AFRL-RH-WP-TR-2017- 0093

INTEGRATED SPEECH AND LANGUAGE TECHNOLOGY FOR INTELLIGENCE, SURVEILLANCE, AND RECONNAISSANCE (ISR)

David M. Hoeferlin
Michael R. Hutt
Stephen A. Thorn

SRA International, Inc., a CRSA Company
5000 Springfield Street, Suite 200
Dayton, OH, 45431

Katherine M. Young

N-Space Analysis, LLC
305 Winding Trail
Xenia, OH 45385

Timothy R. Anderson

711 HPW/RHXS
Wright-Patterson AFB, OH 45433

JULY 2017

Final Report for 26 April 2012 to 8 July 2017

AIR FORCE RESEARCH LABORATORY
711TH HUMAN PERFORMANCE WING
AIRMAN SYSTEMS DIRECTORATE
HUMAN-CENTERED ISR DIVISION
HUMAN TRUST AND INTERACTION BRANCH
WRIGHT-PATTERSON AFB OH 45433
AIR FORCE MATERIAL COMMAND
UNITED STATES AIR FORCE

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th Air Base Wing Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RH-WP-TR-2017- 0093 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

// signed //	// signed //
Timothy R. Anderson, Ph.D., WUM Human-Centered ISR Division Human Trust and Interaction Branch 711th Human Performance Wing Air Force Research Laboratory	Louise A. Carter, Ph.D., DR-IV Human-Centered ISR Division Airman Systems Directorate 711th Human Performance Wing Air Force Research Laboratory

This report is published in the interest of scientific and technical information exchange and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
8-July-2017	Final	26 April 2012 – 8 July 2017

4. TITLE AND SUBTITLE

Integrated Speech and Language Technology for Intelligence, Surveillance, and Reconnaissance (ISR)

5a. CONTRACT NUMBER
FA8650-09-D-6939

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
63456F

6. AUTHOR(S)
[1]David M. Hoeferlin (CSRA)
[1]Michael R. Hutt (CSRA)
[1]Stephen A. Thorn (CSRA)
[2]Katherine M. Young (N-Space Analysis, LLC)
[3]Timothy R. Anderson (711 HPW/RHXS)

5d. PROJECT NUMBER

5e. TASK NUMBER
0029

5f. WORK UNIT NUMBER
H0A3

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
1. SRA International, Inc., a CRSA Company
5000 Springfield Street, Suite 200
Dayton, OH 45431

2. N-Space Analysis, LLC
305 Winding Trail
Xenia, OH 45385

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
3. Air Force Materiel Command
Air Force Research Laboratory
711 Human Performance Wing
Airman Systems Directorate
Human-Centered ISR Division
Wright-Patterson AFB, OH 45433

10. SPONSOR/MONITOR'S ACRONYM(S)
711HPW/RHXS

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RH-WP-TR-2017- 0093

12. DISTRIBUTION AVAILABILITY STATEMENT
Distribution A: Approved for public release. Distribution is unlimited.

13. SUPPLEMENTARY NOTES
88ABW-2018-1473, cleared 26 Mar 2018
Contract FA8650-09-D-6939 TO 0029 conducted inhouse R&D in support of inhouse work unit H0A3.

14. ABSTRACT
This report provides research results to serve DoD and NASIC interests in the areas of Automatic Speech Recognition (ASR), Machine Translation (MT), and Natural Language Processing (NLP).

15. KEYWORDS
natural language processing, machine translation, language modeling, automatic speech recognition (ASR), information retrieval (AR), speech synthesis

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
Unclassified					Timothy R. Anderson
a. REPORT	b. ABSTRACT	c. THIS PAGE	SAR	215	19b. TELEPONE NUMBER (*Include area code*)
U	U	U			N/A

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18

Table of Contents

List of Figures

List of Tables

SUMMARY

This document provides a detailed presentation of the work completed by CSRA under the Information Operations Cyber Exploitation Research (ICER) Task Order 29, titled *Integrated Speech and Language Technology for Intelligence, Surveillance, and Reconnaissance (ISR)*. This work was performed for the Human Trust and Interaction Branch of the Air Force Research Laboratory (AFRL) 711th Human Performance Wing, Aiman Aiding Directorate, over the period 26 April 2012 to 8 July 2017 under contract FA8650-09-D-6939. Primary research and lab support activities were conducted at the AFRL Speech and Communication Research, Engineering, Analysis, and Modeling (SCREAM) Laboratory at Wright Patterson Air Force Base, OH.

The accomplishements under ICER Task Order 29 include: (1) Research and experimentation in machine translation (MT), automatic speech recognition (ASR), and natural language processing (NLP) methodologies; (2) tool optimization and development; (3) research into applications for military use; (4) foundational work in the preparation of corpora and datasets; and (5) Information Technology (IT) support to laboratory operations and maintenance. There was an exploratory analysis into ways of applying submodularity techniques to address computing challenges posed by large datasets in speech and language processing. MT and speech tools were examined for use, including systematic testing and code optimizations to improve performance, and better integrate them into the lab environment. Many purpose-built scripts and tools were developed to facilitate data manipulation, data input/output (I/O), data processing, and, as with *Experiment Reader*, *Qahira*, and *Reverse Palladius*, to contribute specialized enhancements to the craft. Variations in pre-and-post processing methodologies were explored for the purpose of finding advances to MT state-of-the-art. New sources of corpora were procured, existing corpora datasets were groomed to eradicate errors and inconsistencies, and examinations were done in pursuits of performing grammatical annotation.

In addition to the aforementioned research-oriented activities, the IT system administration team provided necessary support to laboratory computing and network operations. General activities included important behind-the-scenes operations and maintenance work, such as software patching, troubleshooting, repairs, user support, performing regular backups, etc. Several major hardware upgrades were also conducted to modernize and improve the lab's data storage and processing capabilities.

1.0 INTRODUCTION

This document provides a summary of the work completed by CSRA under the Information Operations Cyber Exploitation Research (ICER) Task Order 29, titled *Integrated Speech and Language Technology for Intelligence, Surveillance, and Reconnaissance (ISR)*. This work was performed for the Human Trust and Interaction Branch of the Air Force Research Laboratory (AFRL) 711th Human Performance Wing, Airman Systems Directorate, over the period 26 April 2012 to 8 July 2017 under contract FA8650-09-D-6939. Primary research and lab support activities were conducted at the AFRL Speech and Communication Research, Engineering, Analysis, and Modeling (SCREAM) Laboratory at Wright Patterson Air Force Base, OH.

The core of the work accomplished under Task Order 29 is detailed in section 2.0 "EXPERIMENTS & ACCOMPLISHMENTS", organized under three primary categories. The first category, 2.1 "ASR, MT, and NLP", covers research, evaluation, and development activities directed at technologies and methodologies associated with automatic speech recognition (ASR), machine translation (MT), and natural language processing (NLP). The content in this section is organized in accordance with three predominant research subtasks; applications in submodularity techniques for training data, applications tailored to AFRICOM missions, and prototype tools/methodology. The second category, 2.2 "Laboratory Corpora Support", describes corpora-related work on new and existing corpora. This involved acquiring new corpora, performing translations on existing corpora, grooming corpora to eliminate errors and inconsistencies, and annotating corpora. The third category, 2.3 "Laboratory System Admin Support", is the Information Technology (IT) administration and support work dealing with the day-to-day maintenance and operations of SCREAM Lab computer systems and networks. Other miscellaneous activities in relation to Task Order 29 are presented in an additional fourth section, 2.4 "Additional Activity".

A recap of the activities, observations, and results are presented in section 3.0 "CONCLUSIONS".

A complete list of the references cited within section 2.0 is provided in 4.0 "REFERENCES".

Supplementary content is presented at the end of this document in five appendices. This includes several full-text manuscripts of pertinent conference papers, manuals, and reports.

Finally, a complete list of acronyms and definitions is provided at the very end in "LIST OF ACRONYMS & GLOSSARY".

2.0 EXPERIMENTS & ACCOMPLISHMENTS

This section describes, in detail, the work performed pursuant to ICER Task Order 29 covering various aspects of SCREAM Laboratory research and operations, to include language translation and modeling, tool evaluation and development, data retrieval and production, and lab infrastructure support. Subsection 2.1 "ASR, MT, and NLP" encompasses a bevy of activities performed in the pursuit of improving training of MT systems, mission-relevant ASR research for AFRICOM, software tools, and analyses of translation methodologies. 2.2 "Laboratory Corpora Support" covers wide-ranging efforts to acquire, groom, and annotate corpora for current and future tasks. 2.3 "Laboratory System Admin Support" outlines equipment purchases and associated work performed in the maintenance and operation of SCREAM Lab network infrastructure. Finally, 2.4 "Additional Activity" summarizes a multi-day training program on speech synthesis attended by CSRA personnel.

Under the ICER contract, work on ASR, MT, NLP, and related technologies and data is executed under multiple task orders in support of the SCREAM Lab. While individual task orders may serve distinct research objectives and interests, much of the amassed tradecraft knowledge and materials (software utilities, algorithms, and procedures) may be utilized across multiple tasks. Thus, some of the significant experiments and accomplishments described in this report for Task Order 29 may also appear in reports for other task orders, especially if work was apportioned across multiple tasks.

2.1 ASR, MT, and NLP

A large portion of the Task Order 29 effort concentrated on advancing ASR, MT, and NLP through examinations of current and prospective methodologies and software tools. The discussion of this work is categorized and presented below in three subsections. An exploratory study into potential benefits from incorporating *submodularity* techniques into language and speech processing methodology is summarized in subsection 2.1.1 "Submodularity for Language and Speech Processing". A couple of research thrusts directed at applying ASR technology to serve AFRICOM mission tasks are described in subsection 2.1.2 "Applications to AFRICOM". The final subsection, 2.1.3 "Prototype Tools & Methodology", covers a number of distinct activities aimed at analyzing and assessing various software tools and methodologies for MT and speech.

2.1.1 Submodularity for Language and Speech Processing

A series of experiments were conducted to probe the usefulness and practicality of applying principles of submodularity to language and speech processing. The primary focus of the work was put on addressing computational challenges in the training of MT and ASR systems, and in language modeling, with the primary factor being the consumption of large amounts of data. The experiments touched on a range of issues; the scalability of submodular data selection, submodular optimization using an expanded set of features, submodular feature selection and phrase table pruning for SMT, and submodular data selection for language modeling. Also, as an added bonus, the researchers conducted preliminary experimentation on a new theoretical approach aimed at addressing the detrimental effects of two competing criterions inherent in submodular data selection, relevance, and redundancy.

The full embodiment of this work is documented in a separate technical report titled "Submodularity for Speech and Language Applications" (Kirchhoff, K., et al. 2016) [1], which is included in its entirety as an appendix, "APPENDIX E: Submodularity for Speech and Language Applications". The following subsections summarize the experiments performed and outcomes observed from this research effort.

2.1.1.1 Experiments

Data Subset Selection: Scalability to Large Datasets

A major challenge in applying submodular optimization to language processing is the computational overhead required to analyze and process large amounts of training data. Simplistic approaches to this problem have been contemplated, but these generally wind up being infeasible in practice. Therefore, this research effort examines a couple of *approximate* submodular techniques and their ensuing performance in terms of processing speed and BLEU score:

- Two-Stage Optimization (Mirzasoleiman et al., 2013) [2]
- Tailored Algorithm (Wei & Bilmes, 2014) [3]

The two-stage technique was evaluated in the context of language processing, and the tailored algorithm technique was evaluated in the context of statistical MT performance.

In the case of the latter, the primary evaluation criteria were data selection speed and BLEU score.

Submodular Feature Types

Three separate experiments were conducted to examine how incorporating different feature parameters into submodular data selection affects translation performance. In these experiments, the optimization is performed with an augmented feature set, extending the standard set of source language n-grams. The following kinds of features were examined:

- Bilingual n-gram Features based on Translation Hypotheses
- Confidence-Weighted Features
- Structural Syntactic Features

Each experiment compared BLEU score performance between a system using the augmented feature set to a baseline using the standard n-gram features. BLEU scores were assessed at several discrete data points covering a range of data subset sizes.

Feature Subset Selection

This experiment applied submodularity in reducing the dimensionality of the feature set. The researchers modeled the submodular processes after an approach that uses a graph-based method of extracting a subset of sparse features for optimization (Liu et al., 2013) [4]. Three instantiations of this method were tested:

- *Basic Method*: graph-based submodular function, subset pruned to reduce redundancy

- *Test-Set Adaptive Feature Selection* Method: graph-based, with additional relevance factor computed with respect to test set feature occurrences and weighting.
- *Iterative Feature Selection w/ Weight Tuning*: graph-based, with additional relevance factor iteratively tuned with respect to test set feature occurrences and weighting.

Phrase-Table Pruning

Submodularity was examined as a means of addressing shortcomings in conventional phrase-table pruning methods, generally, in their inability to incorporate inter-phrasal dependencies. The researchers initially decided upon two approaches, and formulated the respective objective functions. However, upon assessing the feasibility of these proposed methods, it was decided that the optimization processes would be unsuitable for practical application. The experiment was then conducted using a simpler *alternative* submodular method that would still address the problem of inter-phrasal dependencies. This technique entails submodular selection with a new feature that is a weighted combination comprised of a phrase-pair relevance score factor and significance-based or relative entropy scores.

The experiment compared BLEU score performance two prominent conventional pruning methods, significance-based pruning and relative entropy pruning, and those conventional methods augmented by the aforementioned alternative submodular method.

Data Subset Selection for Language Modeling

The large dataset sizes required by language modeling training prompted the use of an *approximate* submodular method for this experiment. The two-pass method, mentioned earlier in the section, "Data Subset Selection: Scalability to Large Datasets", was chosen to serve in this capacity. Additionally, with an interest in gauging any potential effect from choice of language model, the experiment was performed separately using a back-off model and a RNNLM.

The experiment compared the performance of the approximate submodular method with a conventional data selection method, cross-entropy (Moore et al., 2010) [5]. Performance was assessed in terms of *perplexity*, taken at several discrete data set sizes. In addition, the BLEU score performance was assessed and compared in taking a second-pass rescoring using the RNNLM.

Data Subset Selection: A New Theoretical Approach

The researchers devised a two-stage submodular method with an aim to better accommodate the two competing factors in the standard application of submodularity, relevance and redundancy. This approach isolates the two factors, and optimizes for them separately; the first stage maximizes relevancy, and the second minimizes redundancy. An initial subset is extracted in the first stage, where similarity is evaluated between sentences in the subset and test set on the basis of a threshold referred to as the *hyperparameter*. Then the latent redundancy in the initial subset gets removed in the second stage, factoring-in inter-sentence dependencies from a reduced feature set.

2.1.1.2 Results and Conclusions

Overall, the experiments revealed mixed results in terms of feasibility and benefits realizable through applying submodularity techniques to various aspects of language and speech

processing. It was shown that submodularity could be effective at achieving significant reductions in dataset size without compromising BLEU score performance. The effect on speed-of-operation, however, was not as definitive, as moderate reductions in dataset size were found to impose a speed penalty (slower than with the baseline full dataset). Favorable results were also obtained with phrase-table pruning and in data selection for language modeling, showing significant performance improvements over some conventional state-of-the-art methods. Results from exercises dealing with feature spaces were not as favorable. Attempts to incorporate additional features beyond the standard source language n-grams did not yield any significant performance improvements, and, in certain cases, did worse than the baseline. However, one of the three submodular methods (referred to as the *basic method*) applied towards reducing feature space dimensionality (feature subset selection) showed gains in BLEU score performance.

2.1.1.3 Future Recommendations

A series of avenues for future research were recommended. Preliminary experimentation with the new theoretical approach for data subset selection, which did not yield definitive performance improvements, could be continued for more data points on the *hyperparameter* variable. Exploration into language modeling could be extended to analyze effects of incorporating model characteristics into the submodular data selection process. Finally, it would be prudent to explore the application of submodularity to neural network models; the pruning of large neural networks and dataset partitioning for use in parallelized training.

2.1.2 Applications to AFRICOM

The following two subsections describe research performed in support of AFRICOM mission-related tasks.

2.1.2.1 Research into Somali ASR

Research on Somali ASR made use of previously collected information on Somali grammar and online parallel text resources.

2.1.2.2 Research into Cameroonian French ASR

Researchers noticed that dialectal differences caused problems for ASR of Cameroonian French. A phonological analysis of the Cameroonian French versus standard French pronunciation showed systematic differences that were confirmed by a review of literature on this dialect. Wamba & Noumssi (2003) [6] describe five French dialect regions within Cameroon, with sound changes based on the native languages of each region. Biloa (2003) [7] includes a detailed chapter on the phonology of Cameroonian French. Many of the sound changes in the ASR files matched the descriptions of Wamba & Noumssi, including the pronunciation of a final *–e* which is silent in standard French; the pronunciation of *r* as *k*; and the pronunciation of a nasalized vowel as vowel + velar nasal consonant [ŋ].

2.1.3 Prototype Tools & Methodology

This section covers activities conducted in the advancement of software tools and translation methodology pertaining to MT and speech. Such activities included various assessments and comparisons of MT performance, tool modification and development, and experimentation with new concepts and capabilities. Subsection 2.1.3.1 describes a speech-to-text conversion proof-

of-concept with content in a Microsoft PowerPoint file. Subsections 2.1.3.2, 2.1.3.6, and 2.1.3.18 deal with performance assessments and comparisons for various MT and speech synthesis tools. Subsections 2.1.3.4 and 2.1.3.5 describe new tools, *Reverse Palladius* and *Qahira*, developed in the interest of advancing the state of MT. Subsection 2.1.3.3 describes the *Experiment Reader* tool developed to improve SCREAM Lab MT evaluation workflows. Subsections 2.1.3.7 through 2.1.3.19 describe analyses and assessments of various translation methodologies such as word segmentation, dependency parsing, inflection generation, post-editing, OOV words, domain adaptation, and pivot methods. Subsections 2.1.3.20 through 2.1.3.22 address error analyses of MT system outputs, including in-depth assessments and comparisons of error analysis tools such as *SCLITE*, *Hjerson*, and the SCREAM Lab *Revised Hjerson*.

2.1.3.1 PowerPoint Audio Speech-to-Text

The SCREAM Lab was provided a Microsoft PowerPoint file from a NATO group to use in an ASR proof-of-concept experiment. The objective was to develop a procedure to process embedded slide audio media through ASR, and then re-insert the resultant transcription back into the file as slide *Notes*. The required interactions are facilitated through a little known trick for accessing internal content from within the latest Microsoft Office PowerPoint presentation files (PPTX). The contents are made accessible by renaming the original file as a ZIP formatted file (“.zip” file extension), and then opening it in a file archiving/compression utility. This reveals a folder containing media files and XML files that govern the arrangement of internal slide content.

The appropriate audio was extracted, and then processed through the SCREAM Lab Haystack system to convert speech into text. The first step was to identify the audio files, and associate each of them to their respective parent slide. The process of determining this linkage involved manually locating references to the media in the appropriate XML files. Then the audio files were uploaded into Haystack where they were processed through ASR. The resulting utterance and translation text were then cultivated from the output, and manually pasted into the *Notes* section in the appropriate XML files. After saving the changes to the files, the last step was recreation of the ZIP archive file with the modified content, and then changing the file extension back to “.pptx." The file must use the same name as the directory (and original file), or the PowerPoint slideshow will not run properly.

With the objective attained, the team believes it would be feasible to automate the procedure via future effort. This would generally involve expanding the I/O capabilities in Haystack to accept PPTX files. A script could be created to perform the unmasking of the file content, and the reverse step of re-consolidating it as an archive file. A parsing script could then loop through the XML files looking to synchronize media references to media files, process them through ASR and translation, and then insert the resulting text into the right parts of the XML files.

2.1.3.2 Examinations of Select MT and Speech Tools

CSRA developed, evaluated, and/or improved a number of prototype MT and speech synthesis tools for use in the SCREAM lab, including the following:

- Continuous Space Language Model (CSLM)
- Recurrent Neural Network Language Model (RNNLM)

- Trigger-Based Lexicon Model (TriggerLM)
- MIT Long, Short Term Memory Model (LSTM)
- Neural Probabilistic Language Model (NPLM)
- XenC (for natural language parsing)
- GlottHMM

Speech tools like HTK (HMM Toolkit) and HTS (HTK for Speech Synthesis) were also examined, though not as extensively. In modifying the different software packages to improve performance and fix defects, a pattern of common mistakes arose. These mistakes were documented, along with a list of requirements for successful operation in the SCREAM lab, into a semi-formal porting process. This process helps to catch frustrating bugs before they enter the lab and become a problem.

Continuous Space Language Model (CSLM)

CSLM [8] proposes an approach to language modeling that uses a neural network to project the words onto a continuous space and make an abstract interpolation to estimate a phrase's probability. This is still closely related to the n-gram approach, but it allows for the probabilities of previously unseen word combinations to be more accurately estimated, rather than relying on a backoff weight and a lower order n-gram entry. While this does not completely eliminate the 'unknown word' problem, it could theoretically give accurate results without knowing all of the words in a sequence.

The CSLM tools did not have any documentation, help files, or format specifications. They had a 'help' option in the binaries, but it was often misleading or wrong. We examined the source code, corrected the 'help' output, and created a set of documents detailing the file formats and their syntax and grammar. Additionally, the source code came with critical defects which prevented it from being used from within a larger system. The data would be fine, but the system would crash upon exiting and halt any further progress in the pipeline. All of the tools required modification to work with the SCREAM lab's large scale corpora. We patched these defects, created a more standard build process, and submitted the changes back to the author, where they were merged into the main project.

This work was performed around the time when General Purpose Graphics Processing Units (GPGPUs) were becoming available from NVidia with the CUDA language and runtime. As CSLM consisted largely of vector transforms and matrix-vector operations, it seemed like a good candidate for evaluating the capabilities of the GPU. Execution speed tests were conducted to quantify any performance improvements attributed to the use of GPGPU hardware.

The initial test was to use a drop-in replacement for BLAS (Basic Linear Algebra Subroutines), which resulted in a significant, but not excessive, speedup. A second, more careful replacement of the computationally heavy training sections with GPU code was performed, achieving further increases in throughput. While the increase in speed was beneficial, the memory of the GPU and the main system still needed to be periodically synchronized, which put a drag on performance. A partial conversion would be insufficient, but a comprehensive conversion would take us too far out of sync with the official version, which would become an ongoing maintenance burden. At this time, the developer of the main version announced that there were future plans to support

GPUs, so the effort was put on hold. A graphic showing the notable performance improvement achieved with the GPGPU is in Figure 1.

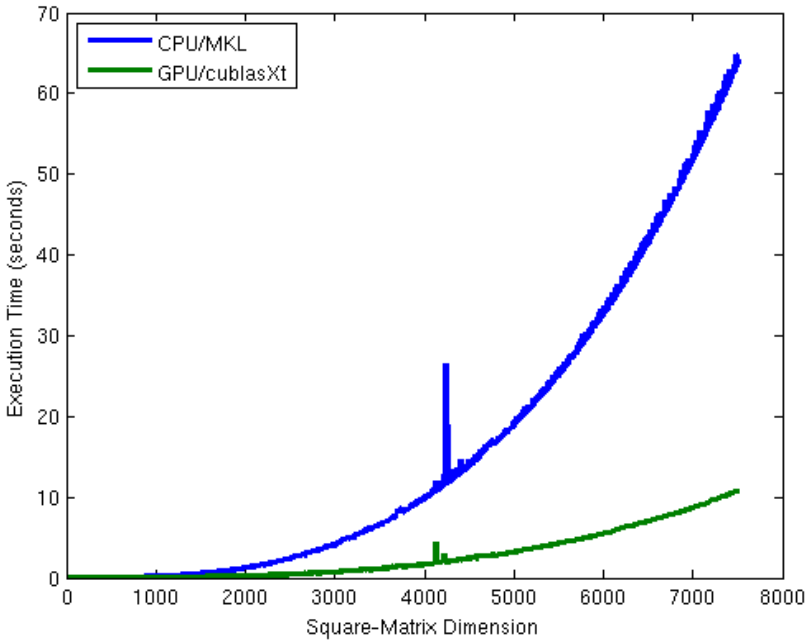


Figure 1: Performance of *CSLM* BLAS DGEMV Call Leveraging GPGPU Hardware

Recurrent Neural Network Language Model (RNNLM)

RNNLM [9] uses a recurrent neural network to efficiently model a language's structure. The motivating idea is that the recurrent structure can capture long range, hidden word dependencies as well as more standard n-gram features by approximating the model of a stream of words as a time series.

As released, RNNLM contained simple implementations of many BLAS functions and general utilities that are available in optimized libraries. It also made some erroneous assumptions about the possible size of the input data. Replacing these implementations with the optimized versions resulted in a 20% increase in throughput. Profiling a typical RNNLM run showed that an excessive amount of execution time was spent reading and writing to disk, so support was added for transparent I/O of compressed files. This also added better buffering support. The added overhead of decompressing the files on the fly was more than offset by the reduced time waiting for the disk.

By far, the largest gains in throughput and execution speed were in analyzing the internal data structures that RNNLM was using. It stored its neurons as a large array, with each element consisting of an activation value and an error value. This is an 'array of structs'. By switching it to use a 'struct of arrays', which stores the activation and error values in separate, cohesive arrays, the total execution time was cut in half (as measured against the already optimized version). Using this style of data organization makes better use of the CPU cache and decreases the number of times that the algorithm has to access main memory. It also leaves it in a much better position to be ported to the GPU.

HTK for Speech Synthesis (HTS)

HTS is a set of utilities for manipulating Hidden Markov Models (HMM) for the purposes of producing synthetic speech. The process for training a model on a speaker, or set of speakers, is highly obtuse and manages to be an involved, manual process while still remaining a mystery to the user. The three main phases of generating a vocal synthesis model are analysis, label generation, and training.

We documented a set of repeatable steps to follow in order to successfully generate a vocal model. These steps were then turned into an interactive script that constrains the user to select valid values and then automates the entirety of the process. Many parts of the training process are highly data parallel, so those steps were extracted and converted to run independently. It also pushes as much error checking as possible to the beginning of the process, so they can be corrected sooner.

For the Wall Street Journal dataset, the analysis and label generation phases were reduced from 173 hours to 9.5 hours – a 94% reduction. However, the actual time that the user is involved is reduced to around 15 minutes. In addition, the analysis phase was converted to work with wav files in addition to raw files, eliminating the need to store both file types, thereby reducing storage requirements by half.

Trigger-Based Lexicon Model (TriggerLM):

We designed and developed a trigger-based lexicon model, as described in “Extending Statistical Machine Translation with Discriminative and Trigger-Based Lexicon Models” [10]. While their model was constrained to triggering a single word in the target language based on the presence of a pair of words in the source model, our software was ultimately configurable to model any grouping of words in either source or target. The goal is to capture long distance effects due to verb splits, translator variance, or simply basic structural differences in the languages. This type of model could be useful in tasks that require relative ranking of a set of candidate translations, like rescoring. The main hurdle is that the size of the model gets very large, very quickly - each combination set in the source language will need to be paired with each combination set in the target language. A moderately sized corpus can easily exceed 50 GB uncompressed. Adding support for compression brings that size down to 10-15 GB on average, which is still a bit heavy. Additionally, generating all of the possible combinations of a certain size for each language is computationally expensive. Splitting the model training phase into two parts (trigger pair counting and recombination/normalization) allowed the computational cost to be spread out over multiple machines.

Further optimizations, like pruning, more aggressive pipelining, and caching, could be applied, but the qualitative performance of the model was determined to be lacking. The results of the paper could not be duplicated on our local dataset. Using a reduced set of data showed a mild improvement over a standard 5-gram model, but the effect is lost when using a larger dataset. Due to the heavy resources necessary to use the TriggerLM, further development was halted.

Recommendations

Software defects are always going to be present, but there are certain postures we can take to lessen their impact and make them more identifiable. The SCREAM lab has a somewhat uncommon set of use cases that surface otherwise hidden bugs:

- Datasets that are larger, in all aspects, than the capacity of a standard signed integer.
- Large numbers of large files being accessed over a network drive.
- Extremely long running processes.
- Pipeline evaluation (sometimes).
- Building power reliability outside the range of a traditional HPC environment.

From these conditions, we have induced the following rules to make a piece of software usable in the lab:

- Use the largest available integer datatype available on the platform for any data driven tabulation value. For C/C++, this is size_t.
- Streaming the input data is vastly preferable to reading it into memory all at once.
- If access is sequential, reading a file into virtual memory (via mmap) produces no gain.
- All input and output files must support compression. Gzip support can be added transparently to most projects.
- During development, compile with clang (LLVM) and use '–fsanitize-address –fno-omit-frame-pointer' to help find buffer/integer overflow bugs.
- Where applicable, use checkpoints so processing can be restarted in the event of a power outage.

Having such large datasets exaggerates slow processes, such that seemingly small improvements make a big difference. Improving the performance by 5% does not seem like much, but if the process runs for two weeks, it will save 16 hours of waiting. This has led to a few simple guidelines for producing more performant code:

- Examine all loops, especially ranged loops. Where applicable:
    - Replace 'copying' and initialization loops with memory operations.
    - Try to remove all branching from the loop body.
- Try to organize data in the same way in which it will be accessed. This is described in the earlier section on RNNLM, "Recurrent Neural Network Language Model (RNNLM)", as using a 'struct of arrays' instead of an 'array of structs'. This will improve cache performance *significantly*.
- All code should be audited and replaced with calls to BLAS where possible. BLAS is highly optimized, vectorized, and correct.
- Compile all code with the Intel compiler and generate a vectorization report.
- Profile a representative run of the software to identify the slow parts.

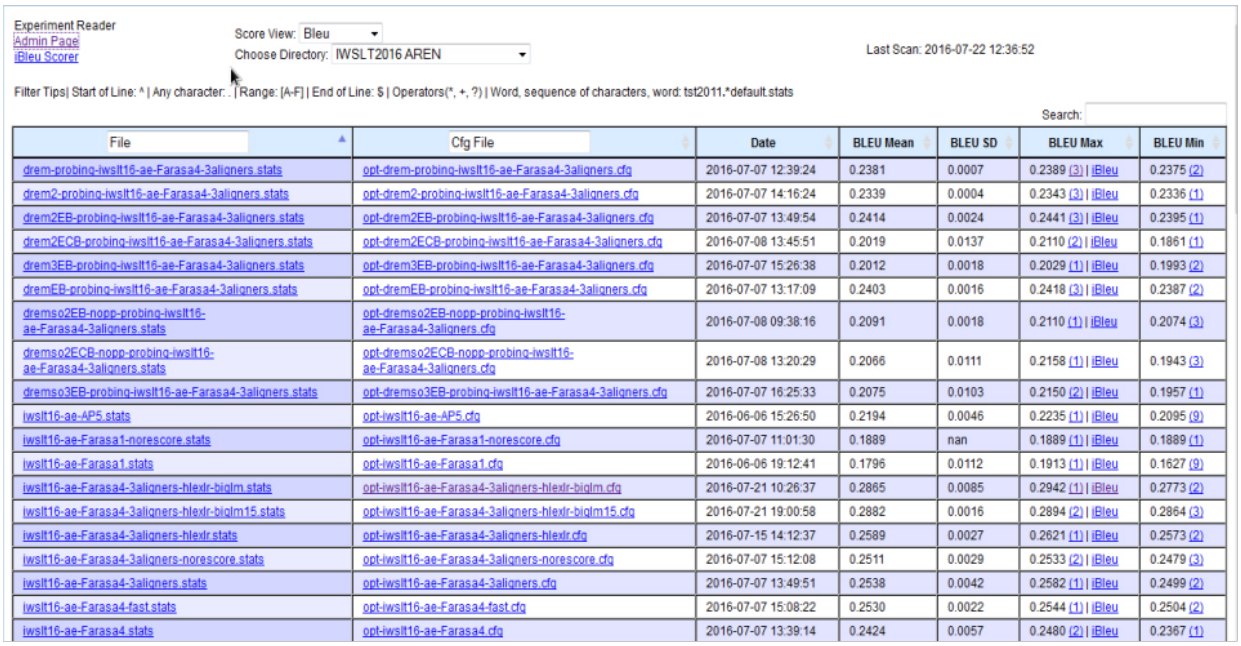
2.1.3.3 Experiment Reader

The *Experiment Reader* is a web application for sorting through enormous amounts of scoring data created during the Moses Machine Translation process. This application creates a viewport

for sorting the scores, statistics, dates and configuration files, and for drilling down into further details of the translation.

Architecture

The Moses MT output is saved out to a folder containing files such as the stats, results, scores, logs and configuration. Code was written to parse out the scoring data from stats files generated by MT processing runs, along with other pertinent metadata, and save it to a MySQL database. This gives users the ability to perform timely queries on MT results from a frontend user interface (Figure 2). Overall, this application design is based on a simple architecture for viewing and administrating, providing for easy upkeep and scoring updates.



Experiment Reader
Admin Page
iBleu Scorer

Score View: Bleu
Choose Directory: IWSLT2016 AREN

Last Scan: 2016-07-22 12:36:52

Filter Tips| Start of Line: ^ | Any character: . | Range: [A-F] | End of Line: $ | Operators(*, +, ?) | Word, sequence of characters, word: tst2011,*default.stats

Search:

File	Cfg File	Date	BLEU Mean	BLEU SD	BLEU Max	BLEU Min
drem-probing-iwslt16-ae-Farasa4-3aligners.stats	opt-drem-probing-iwslt16-ae-Farasa4-3aligners.cfg	2016-07-07 12:39:24	0.2381	0.0007	0.2389 (3) | iBleu	0.2375 (2)
drem2-probing-iwslt16-ae-Farasa4-3aligners.stats	opt-drem2-probing-iwslt16-ae-Farasa4-3aligners.cfg	2016-07-07 14:16:24	0.2339	0.0004	0.2343 (3) | iBleu	0.2336 (1)
drem2EB-probing-iwslt16-ae-Farasa4-3aligners.stats	opt-drem2EB-probing-iwslt16-ae-Farasa4-3aligners.cfg	2016-07-07 13:49:54	0.2414	0.0024	0.2441 (3) | iBleu	0.2395 (1)
drem2ECB-probing-iwslt16-ae-Farasa4-3aligners.stats	opt-drem2ECB-probing-iwslt16-ae-Farasa4-3aligners.cfg	2016-07-08 13:45:51	0.2019	0.0137	0.2110 (2) | iBleu	0.1861 (1)
drem3EB-probing-iwslt16-ae-Farasa4-3aligners.stats	opt-drem3EB-probing-iwslt16-ae-Farasa4-3aligners.cfg	2016-07-07 15:26:38	0.2012	0.0018	0.2029 (1) | iBleu	0.1993 (2)
dremEB-probing-iwslt16-ae-Farasa4-3aligners.stats	opt-dremEB-probing-iwslt16-ae-Farasa4-3aligners.cfg	2016-07-07 13:17:09	0.2403	0.0016	0.2418 (3) | iBleu	0.2387 (2)
dremso2EB-nopp-probing-iwslt16-ae-Farasa4-3aligners.stats	opt-dremso2EB-nopp-probing-iwslt16-ae-Farasa4-3aligners.cfg	2016-07-08 09:38:16	0.2091	0.0018	0.2110 (1) | iBleu	0.2074 (3)
dremso2ECB-nopp-probing-iwslt16-ae-Farasa4-3aligners.stats	opt-dremso2ECB-nopp-probing-iwslt16-ae-Farasa4-3aligners.cfg	2016-07-08 13:20:29	0.2066	0.0111	0.2158 (1) | iBleu	0.1943 (3)
dremso3EB-probing-iwslt16-ae-Farasa4-3aligners.stats	opt-dremso3EB-probing-iwslt16-ae-Farasa4-3aligners.cfg	2016-07-07 16:25:33	0.2075	0.0103	0.2150 (2) | iBleu	0.1957 (1)
iwslt16-ae-AP5.stats	opt-iwslt16-ae-AP5.cfg	2016-06-06 15:26:50	0.2194	0.0046	0.2235 (1) | iBleu	0.2095 (9)
iwslt16-ae-Farasa1-norescore.stats	opt-iwslt16-ae-Farasa1-norescore.cfg	2016-07-07 11:01:30	0.1889	nan	0.1889 (1) | iBleu	0.1889 (1)
iwslt16-ae-Farasa1.stats	opt-iwslt16-ae-Farasa1.cfg	2016-06-06 19:12:41	0.1796	0.0112	0.1913 (1) | iBleu	0.1627 (9)
iwslt16-ae-Farasa4-3aligners-hlexlr-biglm.stats	opt-iwslt16-ae-Farasa4-3aligners-hlexlr-biglm.cfg	2016-07-21 10:26:37	0.2865	0.0085	0.2942 (1) | iBleu	0.2773 (2)
iwslt16-ae-Farasa4-3aligners-hlexlr-biglm15.stats	opt-iwslt16-ae-Farasa4-3aligners-hlexlr-biglm15.cfg	2016-07-21 19:00:58	0.2882	0.0016	0.2894 (2) | iBleu	0.2864 (3)
iwslt16-ae-Farasa4-3aligners-hlexlr.stats	opt-iwslt16-ae-Farasa4-3aligners-hlexlr.cfg	2016-07-15 14:12:37	0.2589	0.0027	0.2621 (1) | iBleu	0.2573 (2)
iwslt16-ae-Farasa4-3aligners-norescore.stats	opt-iwslt16-ae-Farasa4-3aligners-norescore.cfg	2016-07-07 15:12:08	0.2511	0.0029	0.2533 (2) | iBleu	0.2479 (3)
iwslt16-ae-Farasa4-3aligners.stats	opt-iwslt16-ae-Farasa4-3aligners.cfg	2016-07-07 13:49:51	0.2538	0.0042	0.2582 (1) | iBleu	0.2499 (2)
iwslt16-ae-Farasa4-fast.stats	opt-iwslt16-ae-Farasa4-fast.cfg	2016-07-07 15:08:22	0.2530	0.0022	0.2544 (1) | iBleu	0.2504 (2)
iwslt16-ae-Farasa4.stats	opt-iwslt16-ae-Farasa4.cfg	2016-07-07 13:39:14	0.2424	0.0057	0.2480 (2) | iBleu	0.2367 (1)

Figure 2: *Experiment Reader* Application User Interface

The inner-workings of Experiment Reader entail a series of PHP scripts to accomplish the ranking process. *Index.php* is the component used in selecting particular scoring systems to view, and to subsequently sort, search, and filter scores. It is linked to *Admin.php* and to *iBLEU*. *Dir_scanner.php* is for specifying a directory to be rescanned for new scoring data, and it updates the database. The *Admin.php* component provides I/O management for the application. It is used for archiving older test results, creating new storage hierarchies, and detecting newly generated results output. The *MakeCSV.php* component is used in generating comma-separated file output of scoring data for use in third-party systems.

Front End User Interface

The front end user interface was initially developed to control how the data could be displayed and sorted. Each configuration and stats file (Figure 3) displayed is also linked to the actual file so that the user can bring up the complete file for viewing. As the interface was used and the data sets grew larger, users made new requests for capabilities to sort and display the information.

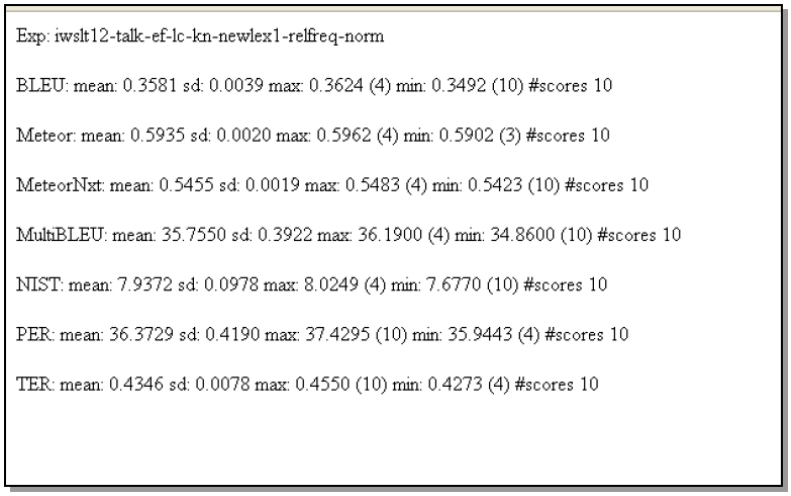


Figure 3: Example Stats File with Various Score Values

For this next phase of development, *jQuery*[1] was adopted to help in the management of the front end. Functionality was added so that each category (directory, file, cfg file, and numerous scores) could be clicked on for sorting the data. Directories and various score categories could be selected through drop-down selectors to allow users full control of the scores they would want to review. At any time, a user can select which directory can be rescanned to detect new scoring data, and then export it all to a comma-separated file for offline viewing or integration into other data manipulation software. Other added functionality includes providing users with easy access to other peripheral information. For example, if a user was browsing the scores and needed more specific information on which configuration file was used, and its contents, they need only click on the cfg file.

Search/Filter Functionality

A filtering system was implemented in jQuery, providing input controls atop columns in the user interface for accepting wildcards and regular expressions. This was done to address frequent requests for an ability to search and filter information, making the presentation to the user easier to manage and read. The new search functionality enables user to filter results from the overall score sets, or by a specific column, such as the file name or config file. This also supports multi-level sorting/filtering, whereby the user can lock-down the initial sort value, and then sort on a secondary parameter. For example, the user locks onto the file with the highest BLEU Score, and then sorts by the Meteor Score to see if similar configurations rise to the top.

Administrative Tools

The next phase of development addressed the means for better I/O management of information. Tools were developed for creating new directories to scan for experiment score results, detecting derelict archive directories, creating downloadable comma-separated files of the results, and detecting changes in any and all directories.

[1] jQuery is a free, open source JavaScript library for dynamic update and control of web pages incorporating various features of client-side scripting.

iBLEU Integration

iBLEU is a JavaScript-based tool created by Nitin Madnani[2] to examine the output from statistical MT and give it a BLEU score down to the segment level. Due to the SCREAM Lab closed network, the iBLEU code was edited to allow translation requests to be sent to the lab's own copy of Systran for translation comparison. Current ongoing integration work is aimed at linking to iBLEU directly from the score sets on the main page.

2.1.3.4 Reverse Palladius

The Reverse Palladius program, *RevP* for short, was developed under ICER Task Order 14 to facilitate the proper translation of Palladius-mapped Chinese Mandarin names from Russian into English. Standard transliterations of such Palladius-mapped names into English are erroneous, and require manual correction. The RevP program provides a user interface from which to pre-translate select names appearing in the source text, thereby reversing the Palladius mapping to revert back to the proper pinyin form before translating (Figure 4). It is also capable of integration with Systran, whereby it will serve-up instant translations of such terms in English. Furthermore, the program allows for continual improvement and efficiency in use through its updatable dictionary of names. RevP was presented at the 2012 conference of the Association for Machine Translation in the Americas (AMTA) [11].

RevP was transitioned to NASIC under Task Order 1. Since that time, there has been only limited interaction with the recipients or users of the program. This activity primarily involved responding to a request from NASIC personnel for technical assistance with the installation and configuration of RevP.

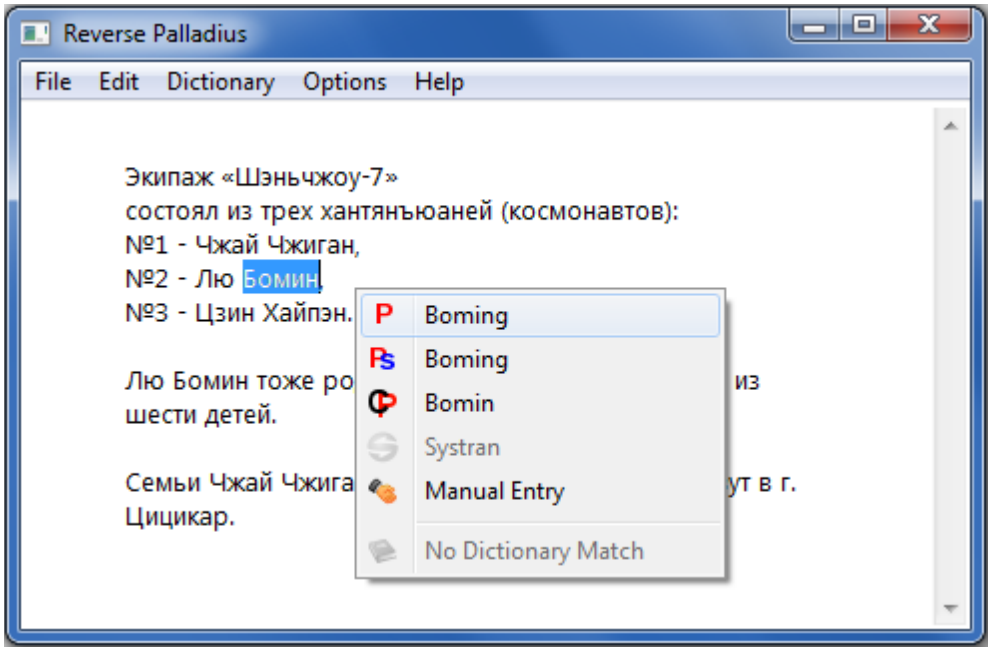


Figure 4: *Reverse Palladius* Application User Interface

[2] http://desilinguist.org

2.1.3.5 Qahira

Qahira is a supervised word alignment editing tool developed and, later, improved upon under other task orders. Thereafter, additional actions were prompted by an expressed interest on the part of the research community in making Qahira available for public use. Then, this effort was initiated, under this task order, by writing a journal article explaining the history of the tool and its functionality. The article (Gwinnup, 2014) [12], which was submitted to 11th International Workshop on Spoken Language Translation (IWSLT) in 2014, is included as an appendix, "APPENDIX C: Qahira: A Word Alignment Viewer and Editor".

The following is a recap of Qahira functionality highlights:

- Graphical user interface with word alignment editing capability (See Figure 5).
- Additional support for display of non-Latin script languages.
- Support for A3Metric Library.
- Word re-ordering via *click-n-drag*.
- Quick translation display (mouseover, *tooltip*); glossary in LDC LCTL Lexicon format.
- Word alignment auditing: grade, probability.
- Sentence sorting by alignment scores.
- Persistent edit/change history.

Additionally, a few enhancements to the program were theorized for future development efforts:

- Adding a preliminary alignment stage to rid Qahira of dependence on an MT system.
- Support for other glossary formats; additional word meaning cues.
- Performance and functionality improvements to increase productivity.

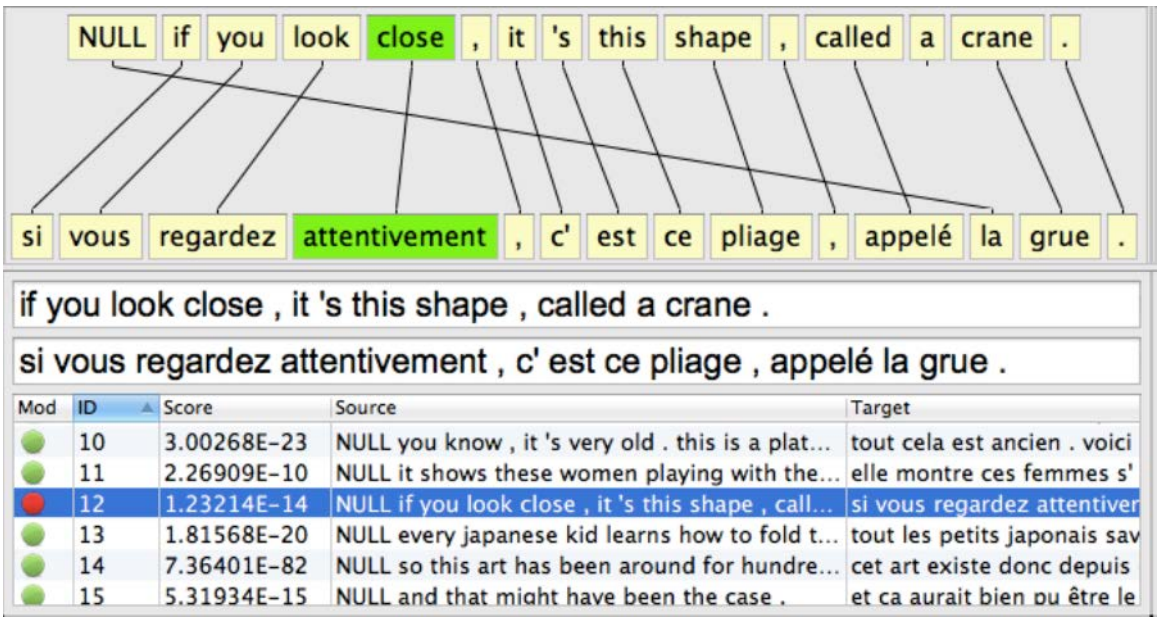


Figure 5: *Qahira* Application User Interface

2.1.3.6 Optical Character Recognition for Chinese MT

Two programs were tested for Chinese optical character recognition (OCR); the Raytheon BBN OCR program and the Google Tesseract OCR program. Difficulties were found for both programs, but the BBN program was nearing the end of its license, and the Tesseract program was more reliable performance-wise. As a result of the evaluation, Tesseract became the exclusive OCR program for use in the SCREAM Lab Haystack system.

Tasks included dataset preparation, and the generation of computer code (scripts, etc.) to post-process and score the OCR output. The test dataset was created from various IWSLT test sets of Chinese PDF documents converted into images for OCR ingestion. Code was developed to rename the file output from the hashed file names created by the OCR server, isolate linkage and text, and re-group into its correct order. Once everything was in its correct order, a Perl script was written to facilitate the parsing of the XHTML output into a text file for sentence alignment and eventual translation. The SCLITE scoring program was used to measure the character error rate and percent correct.

Both OCR programs exhibited a problem in which they break certain characters into their component radicals. For example, the single character, 始, was recognized as the sequence of characters, 女台. A normalization program was written to restore approximately 200 characters that are commonly split during OCR. Some examples of these are shown in Table 1.

Table 1: Example Post-OCR Chinese Character Normalization Rules

Chinese Characters	
OCR	**Correction**
女台	始
石出	□
i炎	□
口合	哈

The two OCR programs also had difficulties with line formats. The Raytheon BBN OCR program identifies zones for processing, which should ideally correspond to individual sentences. However, zoning errors frequently occurred in documents with alternations between long and short lines, with long sentences that wrapped across lines, or with extra line breaks between paragraphs. Documents in newspaper column format also proved to be problematic for zoning. Zoning errors included the creation of overlapping zones, and the clipping of zones to just a portion of long sentences.

Alternate input images were created to confirm the source of zoning errors: The text manipulated to remove wrapped lines, justify margins, and adjust line spacing. This did improve the character error rate, showing the potential performance of the Raytheon BBN OCR program on text in a more uniform layout. The Google Tesseract program, on the other hand, often deleted lines or added blank lines, necessitating a re-alignment of sentences before calculating the error rate. In order to compare the potential usefulness of these programs, a modified version of the IWSLT Chinese test2010 file was created with single-column formatting and justified margins. The character error rates for the two programs on this file were similar; 77.1 for BBN and 79.1 for Tesseract.

2.1.3.7 Chinese Word Segmentation for MT

Segmentation

A statistically-trained Chinese word segmentation program was created, based on data from the ACL Special Interest Group on Chinese Language Processing (SIGHAN) Chinese Word Segmentation bakeoff[3].

Chinese Word Segmentation Lattices

The choice of word segmentation boundaries for Chinese text is sometimes ambiguous. In order to preserve multiple segmentation options during translation, lattice representations were created to represent word segmentation alternatives for input to the Moses system. The Python Lattice

[3] Second International Chinese Word Segmentation Bakeoff

http://sighan.cs.uchicago.edu/bakeoff2005/

Format (PLF) was used with Moses, whereby the lattices were converted to and from the Standard Lattice Format (SLF) in order to apply reweighting.

Various existing word segmentation programs were compared for creating the lattice variants, separately and in combination. This includes the Stanford segmenter trained on either PKU or CTB data, the LTP segmenter, and individual character segmentation.

Initially, equal weights were assigned to various word segmentation lattice arcs. These were then processed to create confusion nets, which were subsequently flattened. Later, language models were used to assign weights to the possible word segmentations.

When using language model weights, the structure of the lattice must be expanded to accommodate different probabilities for the words after the segmentation alternation. Once this expanded structure has been created, the lattice may be re-weighted with a different language model.

For example, in the IWSLT file, dev2010, line104, we see that the characters, 并 "and" and 不 "not", can be segmented together or separately, creating the initial alternation shown below. These characters are highlighted in yellow in the lattice tables and diagrams. The subsequent word, 在于 "lie-in", is highlighted in green.

dev2010-line104:

这个问题并不在于技术本身 "And the problem is not technology itself."

Segmentation Alternatives and Word-for-Word Translations

这个 问题 并不 在于 技术 本身 "this-one issue not lie-in technology itself"

这个 问题 并 不 在于 技术 本身 "this-one issue and not lie-in technology itself"

Equal-Weighted PLF Lattice

(((‘这个',1.0,1),),

((‘问题',1.0,1),),

((‘并',0.5,1),(‘并不',0.5,2),),

((‘不',1.0,1),),

((‘在于',1.0,1),),

((‘技术',1.0,1),),

((‘本身',1.0,1),),)

Equal-Weighted SLF Lattice Table

Table 2: Example Chinese Word Segmentation Lattice Table, Equal-Weighted SLF

Equal-Weighted SLF Lattice Table				
J=0	S=0	E=1	W=这个	a=1.0
J=1	S=1	E=2	W=问题	a=1.0
J=2	S=2	E=3	W=并	a=0.5
J=3	S=2	E=4	W=并不	a=0.5
J=4	S=3	E=4	W=不	a=1.0
J=5	S=4	E=5	W=在于	a=1.0
J=6	S=5	E=6	W=技术	a=1.0
J=7	S=6	E=7	W=本身	a=1.0

Where branches occur in the lattice, each branch has a weight of 0.5, indicating an equal chance for each branch. All the other arcs have a weight of 1.0, indicating that the path must pass through that arc (Figure 6).

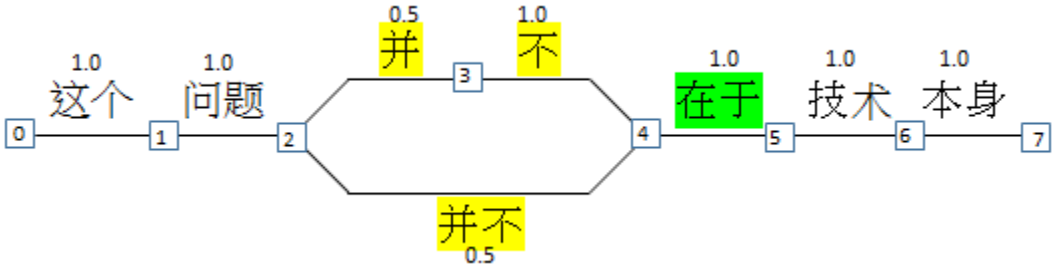


Figure 6: Example Chinese Word Segmentation Lattice

When we look at the word which follows the alternation, 在于 "lie-in", we see that it has a single arc in the equal-weighted lattice. When we apply the language model, however, the probability of this word depends on the preceding two words, which means we need to record a different probability depending on the preceding word segmentation. We therefore derive an expanded lattice representation, with the word, 在于 "lie-in", (highlighted in green in tables and diagrams) occurring on two different paths (Figure 7). The lattice program also inserts a NULL arc for reasons that remain unclear.

Weighted SLF Lattice Table

Table 3: Example Chinese Word Segmentation Lattice Table, Weighted SLF

Weighted SLF Lattice Table					
J=0	S=0	E=2	W=这个	a=2.71828	l=0.0111849
J=1	S=2	E=3	W=问题	a=2.71828	l=0.0352637
J=2	S=3	E=5	W=并	a=1.64872	l=0.00403256
J=3	S=3	E=4	W=并不	a=1.64872	l=0.000473739
J=4	S=4	E=7	W=在于	a=2.71828	l=0.343868
J=5	S=5	E=6	W=不	a=2.71828	l=0.347079
J=6	S=6	E=7	W=在于	a=2.71828	l=0.00280183
J=7	S=7	E=8	W=!NULL		l=1
J=8	S=8	E=9	W=技术	a=2.71828	l=0.000736195
J=9	S=9	E=1	W=本身	a=2.71828	l=4.54135e-05

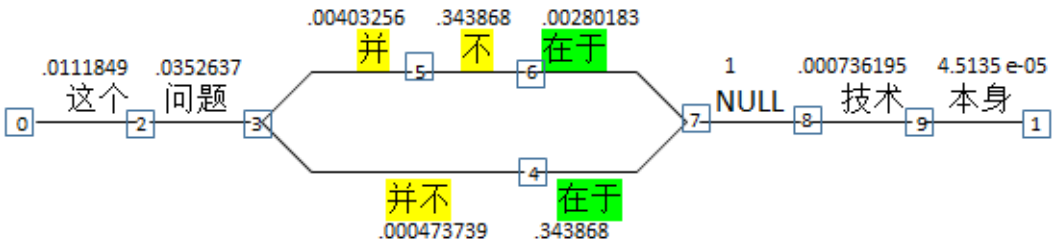


Figure 7: Example Chinese Word Segmentation "Expanded" Lattice

Once this expanded lattice structure has been created, it can be easily re-weighted with a different language model.

2.1.3.8 Dependency Parsing to Change Russian Word Order for MT

Dependency parsing was considered for word re-ordering in Russian. The intent was to test whether Russian sentences could be better translated into English by MT if the Russian sentences were first put in English-like order, subject-verb-object. Russian has this same basic word order, but exhibits many variant orders, using word order for emphasis and relying on morphological agreement to indicate grammatical roles.

Dependency parsing is generally more helpful for free word order languages than constituent parsing, since it identifies semantic relationships instead of strictly structural relationships. An

existing Russian dependency parser, the Malt parser [13], was explored as a resource for dealing with the differing word order of English and Russian. First, the existing POS tagger, TreeTagger, was used to determine Russian POS tags. This output was then given to the Malt parser, which generated dependency parses, as shown in this example (Table 4):

English: “We're all born.”

Russian: Мы все рождаемся.

Malt Parser Output

Table 4: Example Russian Dependency Parses from *Malt Parser*

Russian Dependency Parses from Malt Parser							
Index	**Word**	**Lemma**	**POS**	**POS-Detail**	**Morph**	**Head**	**Dependency**
1	Мы	мы	P	P	P-1-pnn	3	предик
2	все	все	R	R	R	3	опред
3	рождаемся	рождаться	V	V	Vmip1p-m-e	0	ROOT
4	.	.	S	S	SENT	3	PUNC

In this example, the noun, мы “we”, is the subject [предик] of the verb рождаемся “are born”, while the adverb, все “all”, is the modifier [опред] of the verb. The verb is marked as the root; the noun and adverb are marked with number 3, which refers to the verb. We can diagram the relationships this way in Figure 8:

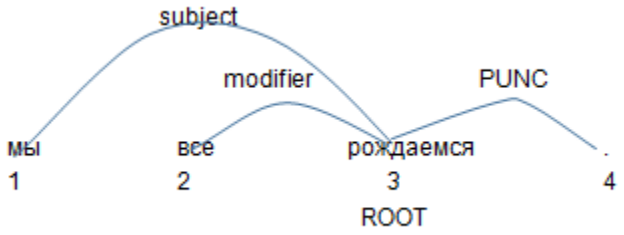


Figure 8: Example Russian Dependency Parsing Diagram

A preliminary program was written to re-order non-SVO Russian sentences. Initially, the program just identifies and outputs the subject, verb, and object of the sentence in SVO order. For example (see Table 5 and Table 6):

Russian Sentence: Этот снимок я сделал в северо-западном районе Амазонки в апреле прошлого года .

Literal Translation: “This picture I took in north-west region Amazon in April last year.”

English Reference: “It's a photograph I took in the Northwest Amazon just last April.”

Output of the Russian Malt Parser

Table 5: Example non-SVO Russian Dependency Parses from *Malt Parser*

Non-SVO Russian Dependency Parses from Malt Parser							
Index	**Word**	**Lemma**	**POS**	**POS-Detail**	**Morph**	**Head**	**Dependency**
1	Этот	этот	P	P	P--msna	2	опред
2	снимок	снимок	N	N	Ncmsan	4	1-компл
3	я	я	P	P	P-1-snn	4	предик
4	сделал	сделать	V	V	Vmis-sma-p	0	ROOT
5	в	в	S	S	Sp-l	4	обст
6	северо-западном	северо-западный	A	A	Afpmslf	7	опред
7	районе	район	N	N	Ncmsln	5	предл
8	Амазонки	Амазонка	N	N	Ncfsgn	7	квазиагент
9	в	в	S	S	Sp-l	8	атриб
10	апреле	апрель	N	N	Ncmsln	9	предл
11	прошлого	прошлый	A	A	Afpmsgf	12	опред
12	года	год	N	N	Ncmsgn	10	атриб
13	.	.	S	S	SENT	12	PUNC

Key Elements of the Dependency Parse

Table 6: Example Russian Dependency Parse Elements

Russian Dependency Parse Elements				
Dependency	**Word**	**POS**	**Meaning**	**Case**
ROOT	сделал	V	took	
предик “predic”	я	P	I	NOM
1-компл “1-compl”	снимок	N	snapshot	NOM/ACC

Output of KAssessMalt.java: records the order of the key elements

OSV\\снимок я сделал

“snapshot I took”

Output of KReorderMalt.java: forces an SVO order on the key elements

я сделал снимок

“I took snapshot"

A database was collected from the online Russian newspaper, Pravda, and analyzed using the Malt parser to determine the predominant word order patterns. A program was written to identify the verb, subject, and object from the Russian Malt parser output, and report the order of these three elements. Previous work had noted a preference for English-like subject-verb-object (SVO) word order in the Russian TED Talk data; however, these documents were originally written in English, which may have influenced the word order of the Russian translations. The Pravda sentences exhibit a variety of word orders, but there is still a preference for an SVO pattern when all three elements are present. Next, the tst2014 Russian dataset was analyzed in the same way. Out of 1183 total lines, 322 had the SVO order, while 61 had one of the other 5 orders of subject, verb, and object. 383 lines had some other ordering, e.g., V or VS or multiple sentences per line.

Further examination scored the previously run MT output for each set of test2014 sentences, to see if the English-like SVO sentences are better translated. The translations of the Russian SVO sentences scored 11.69 BLEU points, while the non-SVO sentences scored 10.33 BLEU points. However, the non-SVO sentences tend to be longer. When accounting for sentence length, there is no clear advantage for translation of the SVO Russian sentences.

Table 7: Example Russian Dependency Parsing on Three Datasets

Russian Dependency Parsing, Multiple Datasets			
	Percent of Lines in Each Order		
Order	**Pravda**	**tst2014**	**newstest2014**
V	34	29	25
SV	19	19	18
VS	12	8	12
VO	8	6	8
OV	1	1	1
SVO	20	30	28
SOV	1	1	1
VSO	1	<1	<1
VOS	1	1	1
OSV	1	1	1
OVS	3	3	4
Total Lines	2843	1418	10,769

Dependency parsing results for the three aforementioned datasets are shown in Table 7. So, while reordering via dependency parsing is possible, it is not clear that this will help MT in this language pair.

2.1.3.9 Techniques in Inflection Generation for MT

One problem for translation into a morphologically complex language is the generation of appropriate inflectional affixes. For example, for English to Russian MT, generating the correct form of a Russian noun requires information about case and gender, which is typically not available from the English source sentence. Some translation systems include a separate step for inflection generation. Others use techniques to analyze and annotate the source sentence with the kind of information that can guide the selection of an appropriate inflected form. This section describes attempts to provide information for inflection via dependency parsing and annotation of source sentences.

Dependency Parsing to Create Factored English Training Data

An attempt was made to test whether dependency parsing of an English sentence could be used to create factored input that could guide the selection of inflected forms. For example, given information about dependencies, we might expect the MT system to relate an English word used as a subject with a Russian word in the nominative case, while the same English word used as an object might be related to a Russian word in the accusative case.

Because Russian also has non-nominative subjects with certain verbs, a decision was made to also include as factors, the head word, and the head word's POS. This would potentially allow the system to capture the pattern of Russian experiencer verbs that take dative subjects. For example, the subject of the verb, нравится "like", occurs in dative case:

> Мне нравится эта книга
>
> me-DAT like this-NOM book-NOM "I like this book" (literally, to-me pleases this book)

Therefore, annotating the English subject with the information that its head is the verb, "like", might provide enough information to support the generation of a dative subject in the Russian translation.

The Stanford parser was applied to generate the English dependency parses, which were then used to guide the creation of factored output (Table 8). Punctuation was a problem because the Stanford dependency parser applies an internal tokenization step, and then omits punctuation from the dependency output. In order to retain the punctuation of the source sentence, the Stanford constituent parser was applied first, retaining punctuation, followed by a Stanford constituent-to-dependency conversion program.

Input

> i was the one who sat down and copied them.

Dependency Parse

Table 8: Example of English Dependency Parsing

English Dependency Parsing from Stanford Parser							
Index	**Word**	**Lemma**	**POS**	**POS-Detail**	**Morph**	**Head**	**Dependency**
1	i	_	FW	FW	_	4	nsubj
2	was	_	VBD	VBD	_	4	cop
3	the	_	DT	DT	_	4	det
4	one	_	NN	NN	_	9	nsubj
5	who	_	WP	WP	_	0	erased
6	sat	_	VBD	VBD	_	4	rcmod
7	down	_	RP	RP	_	6	prt
8	and	_	CC	CC	_	0	erased
9	copied	_	VBN	VBN	_	6	conj_and
10	them	_	PRP	PRP	_	9	dobj
11	.	_	.	.	_	4	punct

Factored Output

i|FW|nsubj|one|NN was|VBD|cop|one|NN the|DT|det|one|NN one|NN|nsubj|copied|VBN
who|WP|erased|null|null sat|VBD|rcmod|one|NN down|RP|prt|sat|VBD
and|CC|erased|null|null copied|VBN|conj_and|sat|VBD them|PRP|dobj|copied|VBN
.|.|punct|one|NN

The factors listed are the word, its part of speech, the dependency relationship, the head word, and the part of speech of the head word. For the word, *one*, which is the subject of the verb, *copied*, in this example, the factored representation is: one|NN|nsubj|copied|VBN.

There was a problem using lowercased training data: Lowercasing causes the pronoun, *i*, to be tagged as a foreign word. Lowercasing the pronoun, *i*, and lowercasing sentence initial words sometimes causes changes in the parse as well. For this effort, the parsing changes were not addressed, but a post-process was applied to correct the factor for *i* from FW "foreign word" to PRP "pronoun".

Dependency Parsing to Create Factored Training Data in Other Languages

Dependency parsing was also used to create factored data with other languages, via the TensorFlow SyntaxNet programs. For English, the SyntaxNet program, Parsey McParseface, creates the dependency parse; for other languages, the SyntaxNet universal parser is applied with language-specific models. A program was written to convert the CoNLL format dependency output to factored form.

The dependency parsing worked best after tokenization. Thus, a calling program was written to apply the Moses programs for punctuation normalization and tokenization, using the language-specific Moses non-breaking prefix lists. After tokenization, the calling program also applies the Moses de-escaper to restore punctuation that has been recorded in the ' or " format.

The factor generation process was tested for English, Chinese, German, Czech, Farsi, Russian, Finnish, Arabic, Latvian, Portuguese, Romanian, Spanish, and Turkish. For Chinese, it was necessary to apply word segmentation before dependency parsing. A Russian example is shown here (Table 9):

Input

И я была не одна. "And I was not alone."

Dependency Parse

Table 9: Example of Russian Dependency Parsing

Russian Dependency Parsing							
Index	**Word**	**Lemma**	**POS**	**POS-Detail**	**Morph**	**Head**	**Dependency**
1	И	_	CONJ	CC	fPOS=CONJ++CC	5	cc:preconj
2	я	_	PRON	PRP	Animacy=Inan|Case=Nom|Gender=Fem|Number=Sing|fPOS=ADJ++JJL	5	nsubj
3	была	_	VERB	VBC	Aspect=Imp|Gender=Fem|Mood=Ind|Number=Sing|Tense=Past|fPOS=VERB++VBC	5	cop
4	не	_	PART	NEG	fPOS=PART++NEG	5	neg
5	одна	_	NUM	CD	Animacy=Inan|Case=Nom|Gender=Fem|Number=Sing|fPOS=NUM++CD	0	ROOT
6	.	_	PUNCT	.	fPOS=PUNCT++.	5	punct

Factored

И|CONJ я|PRON была|VERB не|PART одна|NUM .|PUNCT

Currently, only the generalized POS tag is presented as a factor, but other factors could be added. Such factors include the head and dependency relationship, and the language-specific POS tags and morphological information.

English Verb Annotation to Support Russian Inflection Generation

Kirchhoff et al. (2015) [14] showed that an English dependency parse could be used to annotate the person, number, and gender values of the nouns related to the verb, enabling better inflection generation when translating from English into Arabic. A similar technique was used to annotate English verbs for English-to-Russian translation. A dependency parse was used to identify the subject of the verb, and a program was written to read the dependency parse and annotate the verb with the person and number of the subject (Table 10). The hope is that using annotated

verbs during training will allow the system to learn the correct Russian verb inflections for different subjects.

Original: “Would n't you know it ?”

Annotated: “Would n't you know-2p it ?“

Dependency Parse

Table 10: Example Use of English Dependency Parsing for Verb Annotation

English Dependency Parse							
Index	**Word**	**Lemma**	**POS**	**POS-Detail**	**Morph**	**Head**	**Dependency**
1	Would	_	MD	MD	_	4	aux
2	n't	_	RB	RB	_	4	neg
3	you	_	PRP	PRP	_	4	nsubj
4	know	_	VB	VB	_	0	root
5	it	_	PRP	PRP	_	4	dobj
6	?	_	.	.	_	4	punct

The dependency parse was generated via the Stanford parser. An annotation program was written that identifies verbs and their subjects in the dependency parse, considers the quality of the subject, and annotates the verb with the person, number, and gender of the subject. Annotation was applied to verbs with subjects listed as *nsubj* or *xsubj* in the dependency parse (where *xsubj* is typically the subject of an infinitive). Person, number, and gender were derived from the subject's POS tag or from the characteristics of a pronoun for pronominal subjects. Coordinate subjects were counted as plural.

Adjustments were made to preserve punctuation, to maintain line breaks when a line contains multiple sentences, and to adjust tokenization to the style required by the Stanford parser. Contractions were converted to the Stanford style for parsing (*do n't*), so the sequence *n't* can be analyzed as an adverb), and then back to the original Moses style for output (*don 't*).

Verb annotation gives the MT system additional information with which to select the correctly inflected Russian form, but annotation also increases data sparsity by creating distinct forms of the verb. Furthermore, the MT system already has a good chance of associating the correct verb form with the subject if the subject and verb are adjacent and can be extracted as a phrase, while more distant pairs are less likely to be found in the phrase table, leaving the verb open to translation in the wrong inflected form.

Kirchhoff et al. (2015) address the data sparsity issue by only applying their annotation-trained model when their baseline model translates the subject and verb via separate phrases. The SCREAM Lab program simulates the use of a backoff model by restricting annotation to subjects and verbs that occur at a minimum separation distance. This technique was tested for separation minimums of 0, 1, 2, or 3 words. For example, sentence 1 below has the subject adjacent to the verb, so the separation is 0; sentences 2 and 3 have two intervening words between subject and

verb (separation=2). The verb in sentence 1 will only be annotated if the minimum separation distance is 0. In each sentence, the subject and verb are underlined, and the potential annotation is highlighted in yellow.

1) Would n't you know-2p it ?

2) The country was gradually recovering-3p-sg …

3) The interests of people take-3p-pl precedence …

The final annotation was changed to a factored format, in hopes that this would lessen the problem of data sparsity.

Original Annotation

Would n't you know-2p it ?

Factored Annotation

Would|none n't|none you|none know|2p it|none

The factored verb annotations gave disappointing MT results (Table 11).

Table 11: MT Results on Factored Verb Annotations

Translation of Factored Verb Annotations	
BLEU	**File**
0.2213	enru-pb-lc-baseline
0.2032	enru-pb-lc-facvban0

The Hjerson error analysis program was used to identify inflection errors in the output file for both baseline and annotated data, revealing that verb annotation in the basic condition (no separation limit imposed) failed to reduce the number of inflectional errors.

Table 12: Inflection Errors after MT on Factored Verb Annotations

Inflection Errors on Translation of Factored Verb Annotation		
	Inflectional Errors	**Percent (of hypothesis words)**
Baseline	5823	9.35 %
Annotated	5994	9.35 %

Further analysis with Hjerson showed that inflection errors in the baseline system account for fewer than 10% of the total errors (Table 12). Furthermore, verb inflection errors contribute less than 20% of the total number of inflectional errors. So the potential for gain by verb annotation is limited.

Similar errors were made in both the baseline system and the annotated system. Verb errors for both systems primarily involved either number or gender, as opposed to tense or person. Pronoun errors for both systems showed a tendency for oblique cases in place of nominative.

Surprisingly, the use of annotated data had unintended consequences for the other elements in the sentence. While annotations were only applied to verbs in the training data, changes in inflection were observed for nouns and pronouns as well as verbs.

2.1.3.10 Processing Social Media Text for MT

Social media text was examined and techniques were developed to process the abbreviations, slang, handles, hashtags, and URLs that are common in this type of text. Hashtags are used to label text with topics and can take the form *#alpha* or *#alpha_beta*; handles are used to mention usernames, and take the form *@handle,* or *RT @handle:* when used to retweet a message from another user. Tokenization in preparation for MT may cause additional problems by separating the symbols from the word elements.

One approach is to remove the # @ _ and RT symbols, treating the remaining elements as normal words. This is not appropriate in all cases. MT systems may benefit from treating hashtags differently when they occur in different parts of the sentence. Hashtags are commonly used at the beginning or end of a sentence to label a topic, and do not form part of the words of the main sentence. Hashtags may also be used in the middle of a sentence where they serve as an integral part of the structure of the sentence, while also identifying a topic that describes the sentence. Beginning hashtags may also form part of the sentence, typically as the subject of the verb. Gotti et al. 2014 [15] describe the elements outside the main sentence as the prologue and epilogue, and note that "[their] system is improved by translating epilogues, prologues, and text separately." Shapp 2014 [16] introduces the terms, "syntactic inclusion", and, "syntactic exclusion", to describe the different hashtag functions.

A program was written to remove the hashtag # and _ syntax for syntactically included hashtags, while attaching these symbols to the words when dealing with syntactically excluded hashtags. A hashtag is considered initial if it is preceded only by handles or the RT (retweet) element; hashtags near the end of the sentence are considered to be final if they are followed by nothing else or just URLs.

Dependency parsing was also considered for the processing of social media text. The Stanford parser was compared with the TweeboParser, which was designed for social media. The TweeboParser is able to recognize social media elements, including hashtags, handles, and sentence fragments, leading to improved parses.

A preliminary investigation was conducted to examine whether the translation of social media text could be improved with monolingual human post-editing. A four-step process was created to isolate the actual sentence text for editing, while re-attaching the tokenized # and _ characters to hashtags in the prologue or epilogue of the translated line.

Post-Editing Process for Social Media Text

1. automatically detect initial handles and hashtags, reconstitute initial hashtags, and remove OOV words
2. manually mark the boundaries of the main text with square brackets
3. manually edit the main text to improve English fluency
4. automatically reconstitute hashtags in the epilogue

5. automatically remove hashtag characters # and _ within the main text

When tested on a 100-line sample, this process improved the BLEU score by over two points. Because the human editing is time-consuming, a comparison was made doing just the automatic processing steps, just removing the OOV words, or just removing the hashtag syntax. The output shown (Table 13) includes results from doing just part of the process, for example, just removing Arabic, or just removing hashtag punctuation. Step 3, editing for fluency, is the most time-consuming, so one of the options examined was doing just steps 1, 2, and 4.

Table 13: Results from Monolingual Human Post-Editing on MT of Social Media Text

Translation of Social Media Text for Various Post-Editing Schemes		
	BLEU	**Notes**
MT output	15.79	
1	16.63	
1,2,3	17.17	
1,2,3,4	17.86	
1,2,4	17.10	omit human editing; keep manual marking of main text
no OOV	16.57	
no # _	16.90	
no OOV, no # _	17.11	

The resulting scores suggest that a large part of the improvement is due simply to the removal of hashtag syntax.

2.1.3.11 Addressing Code-Switching Challenges in MT

Code switching refers to the way bilingual speakers may switch between languages, sometimes within a single sentence. Code switching has typically been more prevalent in informal speech, so it is a particular problem when working with social media. The use of multiple languages is also increasing over time with greater globalization of communication. The presence of such intentional multiple-language text creates difficulties for MT, which is typically trained on monolingual text. In addition, the possibility of code switching makes it difficult to screen out wrong-language text errors in data compiled from online material.

For example, the Yandex corpus contains Russian hotel reviews that end in the Ukrainian word, Більше “More". This may represent an untranslated hyperlink, something we would like to exclude from the otherwise Russian data. In the following example, the Ukrainian word and its translation are highlighted in yellow:

> RU: В этом отеле, расположенном на расстоянии короткой прогулки от пляжа МакКензи и в 5 минутах езды от аэропорта Ларнака, Вас ожидают апартаменты, в которых царит Більше...

EN: Just a short distance from MacKenzie Beach and 5 minutes' drive from Larnaca Airport, these apartments provide a relaxed and friendly environment, with an more...

Hotel reviews present particular problems when used as sources of parallel text. An English language site often contains reviews in multiple languages, and these may be collected as English data. We have also seen instances in the Common Crawl data where the Russian site apparently contained some English reviews, and these have been collected as "parallel" data to different reviews on the English side, creating non-matching English-English data.

The examples discussed so far may be treated as errors that should be edited. However, when text contains frequent deliberate language alternations, simply removing the non-majority language is not sufficient.

In order to learn more about code switching issues, a researcher attended an Empirical Methods in Natural Language Processing (EMNLP) workshop, Computational Approaches to Linguistic Code Switching, dealing with the use of multiple languages within a single sentence. The workshop presenters emphasized the increasing prevalence of mixed language text. Some social factors influence language choice (including formality, sentiment, and group membership), but mixed language text is no longer limited to informal contexts.

Various strategies were discussed to address mixed text in NLP, including modeling the mixed language per se, splitting the text into sections and applying models for each language respectively, modeling both languages on all the text, and translating all the text into a single source language before translation. These strategies each have drawbacks, such as losing context when splitting the text, or losing the ability to apply monolingual resources when modeling the language mixture.

2.1.3.12 Translating from an Inflectional Language via Source Text Annotation and Re-Ordering

One of the difficulties for Russian to English MT is the use of case inflection to subsume some of the information that is carried by prepositions in English. Researchers at Yandex [17] describe a technique for Russian-to-English MT, in which they analyze the case of Russian nouns and insert a separate word to represent the case information. By inserting the case element before the noun, they anticipate the order of English preposition + noun, and also give a possible way to motivate the generation of an English determiner. The noun itself is tagged with its cardinality and part of speech. The Yandex researchers also tag adjectives for comparative and superlative, and tag verbs for various elements such as tense and aspect.

SCREAM Lab researchers replicated the Yandex process for nouns and adjectives, creating pre-processed Russian files for use as input to MT training. This improved the BLEU score of the MT output.

A program was written to take the Mystem output and create the Yandex-style form, *CASE lemma.N+cardinality*. This example in Table 14 shows the dative plural form дням of the word, день "day". Note that S is the Mystem tag for nouns (substantives).

Table 14: Example Showing *Yandex-Style* Form of Noun Annotation

Yandex-Style Noun Annotation

Word:	дням
Mystem Output:	дням{ день=S,m,inan=dat,pl}
Program Output:	DAT день.N+pl

For adjectives, the program records whether the adjective is positive, comparative, or superlative, as in the example, темным “dark”. In this example in Table 15, Mystem reports multiple possible morphological analyses, separated by the vertical bar, |.

Table 15: Example Showing *Yandex-Style* Form of Adjective Annotation

Yandex-Style Adjective Annotation	
Word:	темным
Mystem Output:	темным{ темный=A=dat,pl,plen|=A=ins,sg,plen,m|=A=ins,sg,plen,n}
Program Output:	темный.A+pos

In combination, the phrase, "toward dark days," becomes "toward dark.A+pos DAT day.N+pl". Also shown (Table 16) is a variant in which the number of the noun is included in the preposed case element.

Table 16: Example Showing Variant Forms of *Yandex-Style* Annotation

Yandex-Style Phrase Annotation	
English:	for the dark days
Literal:	toward dark days
Russian:	к темным дням
Annotated:	к темный.A+pos DAT день.N+pl
Variant:	к темный.A+pos DATPL день.N+pl

Some formatting problems had to be addressed, including the tokenization of HMTL tags in the input data, the escaping of bracket characters to avoid confusion with the Mystem output, the interpretation of underscore and backslash characters, and the removal of residual Yandex-style annotation from OOV words after MT.

Several variants of the Yandex-style annotation were applied to the train, dev, and test files in MT experiments. Results are summarized below. The best-scoring combination was the annotation of nouns and adjectives without listing alternative case elements. This variant achieved a gain of over one BLEU point over the baseline, and nearly half a BLEU point over a secondary baseline (Table 17).

The Mystem program requires tokenized and lowercased input. As a control, a second baseline was created with just tokenization and lowercasing. This scored better than most of the annotated versions, highlighting the importance of this simple processing step.

The example below shows a noun that is ambiguous for nominative or accusative case. The preposed case element was created using just the first case (NOM), using both cases (NOM-ACC), or just using a generic case element (CASE). Number was marked as a suffix on the lemma along with the POS (N+SG), as part of the preposed case element (NOMSG), or left unmarked.

Adjectives are annotated in just two variants, following the Yandex pattern of lemma with suffix indicating part of speech and positive, superlative, or comparative. These scored better than similar variants without adjective annotation.

Table 17: Translation Results from *Yandex-Style* Noun and Adjective Annotation

Yandex-Style Noun and Adjective Annotation			
Adjectives	**Nouns**		**BLEU**
lemma.A+pos	NOM	lemma.N+SG	21.43 (IWSLT)
--	--	lemma.N+SG	21.39
--	CASE	lemma.N+SG	21.01
--	--	--	20.96 (tok & lc)
lemma.A+pos	NOM-ACC	lemma.N+SG	20.94
--	NOMSG-ACCSG	lemma	20.93
--	NOMSG	lemma	20.91
--	NOM	lemma.N+SG	20.89
--	--	lemma	20.88
--	NOM	lemma	20.74
--	NOM-ACC	lemma	20.69
--	--	--	20.30 (baseline)

Another problem arises when the Russian word has multiple possible morphological analyses. Mystem may report ambiguity at the level of POS tags (e.g., some words can serve as either conjunctions or demonstrative pronouns), or in the specification of morphological features within a POS designation (e.g., some noun forms are the same for both nominative and accusative case).

The Yandex paper describes a restriction on POS tag ambiguity, such that no annotations are created for forms with ambiguous POS designations. The SCREAM Lab version instead adopts the Mystem -d option to have the program report only the most probable POS tag.

The Yandex paper authors do not address morphological feature ambiguity. Russian nouns frequently have syncretic (identical) forms for different cases, so preventing annotation of all ambiguous forms would severely limit the case pre-posing technique. However, it is not clear what annotation should be used when several cases are possible.

For example, what should be the parser output when the noun has multiple possible case and number analyses, as in the example below (Table 18)? Here, the form, хлеба “bread”, has the ending –a which can represent accusative plural, genitive singular, or nominative plural.

Table 18: Example of Morphological Feature Ambiguity Affecting Annotation

Parser Annotations on Russian Word, хлеба "bread"	
Original Word:	хлеба "bread"
Mystem Output:	хлеба {хлеб=S,m,inan=acc,pl|=S,m,inan=gen,sg|=S,m,inan=nom,pl}
Program Output A:	NOM хлеб.N+SG
Program Output B:	NOM_ACC_GEN хлеб.N+SG_PL
Program Output C:	Sentence 1: ACC хлеб.N+PL Sentence 2: GEN хлеб.N+SG Sentence 3: NOM хлеб.N+PL

The original form of the SCREAM Lab program creates Program Output A, above, because the program checks the Mystem output for nominative case first; if it fails to find nominative, it goes on to check for accusative, etc. This creates a bias for reporting nominative case. Similarly, the program checks for singular number before checking for plural number. In this instance, the result is the combination of nominative and singular, which is not a legitimate analysis of the form хлеба.

One alternative is to preserve all cases and numbers reported by Mystem, as shown in Parser Output B, above, at the expense of increasing data sparsity. A different option would be to create alternate sentences for each possible morphological analysis, as shown in Parser Output C, above. This option was not tested in the current effort.

In its morphological annotation, Mystem reports the case variants in alphabetical order (abl, act, ins, dat...). This becomes clear with borrowed words, which Mystem often treats as nouns that are completely ambiguous in case and number, as in the example for the foreign name, Сэнди "Sandy", below:

> Сэнди{сэнди=S,persn,mf,anim=abl,pl|=S,persn,mf,anim=abl,sg|=S,persn,mf,anim=acc,pl|=S,persn,mf,anim=acc,sg|=S,persn,mf,anim=dat,pl|=S,persn,mf,anim=dat,sg|=S,persn,mf,anim=gen,pl|=S,persn,mf,anim=gen,sg|=S,persn,mf,anim=ins,pl|=S,persn,mf,anim=ins,sg|=S,persn,mf,anim=nom,pl|=S,persn,mf,anim=nom,sg}

The SCREAM Lab program creates annotations in which the cases follow the order nominative, accusative, genitive, dative, instrumental, and ablative, as shown below. The rare cases VOC (vocative) and PART (partitive) are reported after these, if present. (This order is morpho-syntactically motivated, grouping possibly syncretic/identical forms together.)

> NOM_ACC_GEN_DAT_INS_ABL сэнди.N+SG_PL

The most important requirement is that the order of cases be uniform in the annotations, to avoid creating separate but equivalent elements like ACC_GEN and GEN_ACC.

A second round of annotation experiments added verb annotation. Following the Yandex paper, an aspect element was generated before the verb, and verb itself was lemmatized with annotations for POS, tense, and mood. For these experiments, the training data and language models were updated to include all 2014 IWSLT data. The addition of verb annotations

provided a slight improvement over the previous best annotation variant for nouns and adjectives (Table 19).

Table 19: Translation Results for *Yandex-Style* Annotation

Translation Results for Yandex-Style Annotation	
BLEU Score	**Annotation**
21.42	Baseline (no annotation; tokenized and lowercased)
22.08	IWSLT2014 Best: lemma.A+pos, NOM lemma.N+SG (no ambiguity)
22.31	IWSLT2014+Verbs: adding ASPECT lemma.V+TENSE+MOOD

An example sentence is shown below in Table 20. (Note that there is a grammatical error in the English reference: It should say “in a typical week”, but the determiner is missing.)

Table 20: Sentence Context for *Yandex-Style* Verb Annotation

Yandex-Style Verb Annotation	
English Reference:	Do you know how many choices you make in typical week?
Baseline = Russian reference, tokenized and lowercased:	знаете ли вы , сколько раз в неделю вы делаете выбор ?
IWSLT2014 Best:	знаете ли NOM вы.PRON+PL , сколько.A+pos NOM раз.N+SG в ACC неделя.N+SG NOM вы.PRON+PL делаете NOM выбор.N+SG ?
IWSLT2014 + Verbs:	знать.V+NONPAST+INDICATIVE ли NOM вы.PRON+PL , сколько.A+pos NOM раз.N+SG в ACC неделя.N+SG NOM вы.PRON+PL делать.V+NONPAST+INDICATIVE NOM выбор.N+SG ?

The work on verb annotation led to the discovery of a typo in the Yandex article: The authors state that a verb with non-past tense and imperfect aspect should be marked as future, when actually it is the perfect aspect of a non-past tense verb that is interpreted as future in Russian. For example, in Table 21:

Table 21: Example Challenging *Yandex* Article Remarks on Verb Annotation

Tense Determination on Russian		
Verb	**Morphological Analysis**	**Meaning**
читаю	чита =read ю=1st person, singular	“I am reading”
прочитаю	про=perfect aspect чита =read ю=1st person, singular	“I will read”

In evaluating the annotation systems, it is important to note that the Yandex researchers' annotations are designed to anticipate English morphology, when translating from Russian into English. However, as shown in the examples below, the English phrases do not always conform to the anticipated forms. This may limit the positive effect of Yandex-style annotation to just those sentences in which the English reference exhibits the expected form.

For nouns, the addition of a preposed case element anticipates the determiner (Table 22, Table 23).

Table 22: Anticipation from *Yandex-Style* Noun Annotation on Russian

Anticipation from Yandex-Style Noun Annotation on Russian		
Russian with Annotation	**English Reference**	**Anticipated**
NOM день.N+SG	a typical day	a day

Table 23: Sentence Context; *Yandex-Style* Noun Annotation on Russian

Yandex-Style Noun Annotation on Russian	
English Reference:	Do you know how many choices you make in a typical day?
Russian Source:	знаете ли вы , сколько раз в день вы делаете выбор ?
Annotated Russian:	знаете ли вы , сколько раз в день вы делаете выбор ? знать.V+NONPAST+INDICATIVE ли NOM вы.PRON+PL , сколько.A+pos NOM раз.N+SG в NOM день.N+SG NOM вы.PRON+PL делать.V+NONPAST+INDICATIVE NOM выбор.N+SG ?

For verbs in the third person singular, the Yandex researchers annotate the verb itself, creating a situation in which there is an English-style morphological alternation between the third person singular and the other forms (Table 24 and

Table 25, and see also the previous example).

Table 24: Tense Anticipation from *Yandex-Style* Verb Annotation on Russian

Tense Anticipation from Yandex-Style Verb Annotation on Russian		
Russian with Annotation	**English Reference**	**Anticipated**
делать.V+NONPAST+INDICATIVE	make	make
делать.V+NONPAST+INDICATIVE+3P+SG	reports making	makes

Table 25: Sentence Context; *Yandex-Style* Verb Annotation on Russian

Yandex-Style Verb Annotation on Russian	
English Reference:	I recently did a survey with over 2,000 Americans, and the average number of choices that the typical American reports making is about 70 in a typical day.
Russian Source:	недавно я провела исследование более 2000 американцев : обычный американец утверждает , что в среднем делает выбор около 70 раз в день .
Annotated Russian:	недавно.A+pos NOM я.PRON+SG PERFECT проводить.V+PAST+INDICATIVE NOM исследование.N+SG более.A+pos 2000 ACC американец.N+PL : обычный.A+pos NOM американец.N+SG утверждать.V+NONPAST+INDICATIVE+3P+SG , что в ABL среднее.N+SG делать.V+NONPAST+INDICATIVE+3P+SG NOM выбор.N+SG около 70 NOM раз.N+SG в NOM день.N+SG .

For some verbs, the Yandex researchers add a preposed ASPECT marker for the past tense, in which English is likely to have a form of the auxiliary verbs, *have*, and, *be* (Table 26 and Table 27, and see also the previous example).

Table 26: Tense Anticipation (Past-Tense) from *Yandex-Style* Verb Annotation on Russian

Tense Anticipation from Yandex-Style Verb Annotation on Russian		
Russian with Annotation	**English reference**	**Anticipated**
PERFECT проводить.V+PAST+INDICATIVE	did	had done
IMPERFECT делать.V+PAST+INDICATIVE	used to go	was doing*
*(я делала покупки = literally, "I was-doing shopping")		

Table 27: Sentence Context; (Past-Tense) *Yandex-Style* Verb Annotation on Russian

Yandex-Style Verb Annotation on Russian	
English Reference:	So when I was a graduate student at Stanford University, I used to go to this very, very upscale grocery store; at least at that time it was truly upscale.
Russian Source:	когда я была аспирантом в стэнфордском университете , я делала покупки в одном гастрономе премиум класса , по крайней мере в то время он был таковым .
Annotated Russian:	когда NOM я.PRON+SG IMPERFECT быть.V+PAST+INDICATIVE INS аспирант.N+SG в стэнфордский.A+pos ABL университет.N+SG , NOM я.PRON+SG IMPERFECT делать.V+PAST+INDICATIVE NOM покупка.N+SG в один.APRO+pos ABL гастроном.N+SG NOM премиум.N+SG GEN класс.N+SG , по крайний.A+pos DAT мера.N+SG в тот.APRO+pos NOM время.N+SG NOM он.PRON+SG IMPERFECT быть.V+PAST+INDICATIVE таковой.APRO+pos .

2.1.3.13 Sources of OOV Words in Russian-to-English MT

The appropriate treatment of OOV words in MT output depends on the reason the word is out of vocabulary. Named entities should be treated differently from inflected forms, for example. An analysis was conducted on the WMT newstest2014 file, looking at OOV words in the Russian-to-English MT output, and considering the capitalization of the OOV word in the original Russian file:

- 1412 Total OOV Words
- 770 Common Words

- 586 NE
- 56 Uppercase OOVs at start of sentence (therefore can't tell if they are NE)

Manual review of a selection of OOV words from the IWSLT tst2010 Russian to English MT output considered how these words should best be handled: passed through unchanged, transliterated, translated via an inflection sensitive process, or omitted.

In making this list, the annotator evaluated all OOVs that occurred 2 or more times (n = 71), and then also annotated a random sample of 29 of the remaining singleton OOVs to bring the total to 100 words. This resulted in the following proportion of OOV types:

- 74 should translate
- 22 should transliterate
- 4 should omit
- 0 should pass through

Words that are already written in Latin characters, such as brand names, should be passed through unchanged; no such Latin spellings were found in this sample.

Transliteration is appropriate for many OOV named entities, providing useful information even when the resulting spelling is not an exact match to the reference.

Specialized translation is needed for inflected Russian words that are not found in the training data. Some inflected forms may be found via lexical approximation which relates unknown words to similar, known words.

Finally, some words are too difficult to recover and should be dropped. In this sample, these included Chinese names which do not transliterate well, and sight spellings, in which an English word has been visually translated into the corresponding Cyrillic characters without regard to its actual pronunciation. For example, transliteration will not succeed for the non-standard borrowing, магическая /magičeskaja/ “magical”, which copies the English g, creating in Russian the sound, [g], instead of the needed sound, [dž].

2.1.3.14 Techniques in Pre-Translation of Russian OOV Words for MT

Pre-Translate via Dictionary

The Mystem program tags Russian NE; these tags can be used to drive a pre-processing step, in which named entities are either tagged or replaced with their English translations via dictionary reference. For the Moses system, NE can be tagged with their favored translation in the input, forcing that translation during sentence processing.

The technique of tagging and pre-translating NE was extended to general OOV words in Arabic-to-English MT, using the VarCon (Variant Conversion) program to substitute the English translations for the Arabic OOV words throughout the data for input to another MT system.

Lexical Approximation

Translating from an inflectional language leads to problems with data sparsity, as the training data typically fail to exhibit instances of all the possible inflected forms. Previous researchers

[18, 19] introduced the technique of lexical approximation, in which a missing word in the test data is approximated by another, more common word that varies in spelling or morphological inflection. Previous work in the SCREAM Lab used an existing lexical approximation method, as well as locally-developed stem-and-inflect program [20]. Current efforts extended the stem-and-inflect process.

The stem-and-inflect process identifies source words that cannot be translated via the current phrase table, and replaces them with morphologically related words that are found in the phrase table. The input file is examined to find words which have no unigram entry in the phrase table; we use the program TreeTagger to identify the part of speech, and then we remove inflectional endings to derive a stem. We apply all possible Russian inflectional endings for the given part of speech, and then check the resulting inflected forms for unigram entries in the phrase table. If an inflected form of the OOV word can be found in the phrase table, that form is used to replace the OOV word in the original Russian file. If multiple candidates are found, we use the one with the highest frequency of occurrence in the training data. This process replaces words that we know we cannot translate with semantically similar words that we can translate.

For example, the word, услышало "heard", (past neuter) is not found in the phrase table. TreeTagger identifies this as a verb. We stem it to услыша, and apply the verbal inflectional endings. We find four of the inflected forms in the phrase table: услышали, услышала, услышал, услышать. Of these, the infinitive, услышать "to hear", occurs most frequently in the training data, so we replace услышало with услышать in the original sentence. When translated, this creates a sentence which includes the previously missing English word, "hear". The words for "hear" are highlighted in yellow in the translations appearing in Table 28.

Table 28: Example of Stem-and-Inflect for Russian-to-English Translation

Stem-and-Inflect for Russian-to-English Translation		
Reference:	We want the leaders of NATO to hear our voice, the voice of the Ukrainian people.	
Translating without stem-inflect, and dropping unknowns:	Мы хотим, чтобы руководство НАТО услышало наш голос, голос украинского народа.	We want the NATO leadership our voice, the voice of the Ukrainian people.
Translating with stem-inflect:	Мы хотим, чтобы руководство НАТО услышать наш голос, голос украинского народа.	We want NATO to hear our voice, the voice of the Ukrainian people.

In some instances the stem-and-inflect technique may derive an exact match to the target form, especially when translating into an inflectionally poor language like English. In other instances, the technique returns incorrect verb tenses or swaps singular and plural nouns. However, even these substitutions may offer semantic gains versus leaving the word untranslated.

Previous efforts represented all possible inflected forms via lattices or n-best lists; the current effort instead returns the variant form with the highest frequency in the training data. The

current effort also extends the implementation of the stem-and-inflect technique from the Joshua MT system to Moses, taking into account differences in phrase table format.

The revised stem-and-inflect process was run on the test files for the SCREAM Lab submission to WMT 2014.

2.1.3.15 Techniques in Post-Process Translation of Russian OOV Words for MT

Improved Rule-Based Transliteration

Previous work on OOV words in the SCREAM Lab introduced a rule-based transliteration program to recover information from Russian OOV words. This program applies a simple letter-mapping from Cyrillic characters to Latin characters representing their typical sounds. Examination of transliteration output shows that this could be improved by stemming Russian inflectional endings before transliteration. For example, we want to remove inflections before transliterating фермионами /fermionami/ "fermion" with instrumental-plural ending, /ami/; we also need to remove inflections before transltierating Бозон Хиггса /Bozon Xiggsa/ "Higgs Boson" with gentivie case ending, -a, on the name, Higgs.

However, some foreign words/names have endings that can be mistaken for Russian inflectional endings, so stemming may remove part of the actual word. For example, we don't want to stem -a from the place name, Массада /Massada/.

The decision of whether to stem before transliteration is complicated by the fact that foreign names in Russian text are sometimes inflected and sometimes left uninflected [21] (the full manuscript of this source is also provided as an appendix, “APPENDIX B: A Taxonomy of Weeds: A Field Guide for Corpus Curators to Winnowing the Parallel Text Harvest”). An examination of names borrowed into Russian from English in the TED Talk data showed this range of behavior: a) first and last name both uninflected, b) first and last name both inflected, c) last name only inflected. Here are three examples of possessive constructions which should exhibit genitive case:

a) песню Уитни Хьюстон /uitni x'yuston/ 'a Whitney Houston song' [neither name inflected]

b) закон Артура Кларка /artur+a klark+a/ 'Arthur Clarke's law' [both names inflected]

c) Книга Эл Гора /ɛl gor+a/ 'The Al Gore book' [last name only inflected]

Some improvement was gained in the transliteration program by restricting the stemming program to only remove noun and adjective inflections, not verb inflections.

Improved Lexicon-based Transliteration

Previous work on transliteration in the SCREAM Lab also introduced a lexicon-based transliteration system for Russian OOV words. This program derives the typical sounds for the Cyrillic characters and attempts to match that pronunciation against the English pronunciations given in the Carnegie Mellon University (CMU) Pronouncing Dictionary[4]. This system was improved by extending the lexicon of English pronunciations to include words from the WMT English training file that are not found in the main CMU dictionary. The pronunciations for these missing words were created using the SONIC spell text-to-speech program.

The lexicon-based transliterator first maps the Cyrillic letters into their typical phonetic values, and compares these values with the entries in the CMU Pronouncing Dictionary. If an exact match is not found, the program considers variations of the original word, either by removing possible inflectional endings, or by accepting mismatches in the vowels. When the target word is present in the dictionary, the transliterator does well, but when the word is absent, the transliterator often choses a similar English word instead. For example, one sentence listed fast food restaurants, and included the unknown borrowed words, уэндис "Wendy's", and чикен "(Kentucky Fried) Chicken". The transliterator mapped the Cyrillic letters into the sounds, and then picked the closest words from the CMU Pronouncing Dictionary, getting "chicken" correct, but mapping "Wendy's" into the word, "indus". In processing another sentence, the word, биткойны "bitcoins", was not found in the CMU dictionary; the stemmed variant, биткойн "bitcoin", was also not found, and the stemmed form was unfortunately matched to "botkin". The transliterations of the three sample words are shown in Table 29.

Table 29: Example of Lexicon-based Transliteration of Russian OOV Words

Lexicon-Based Transliteration of Russian OOV Words			
Cyrillic	чикен	уэндис	биткойны
Stemmed	--	--	биткойн
Sound Map	Cikye_yE_e_En	uEndis	bit_Tkoy_yi_in
Consonant Frame	C@k@n	@nd@s	b@tk@n
CMU Sound	CH IH K AH N	IH N D AH S	B AA T K IH N
CMU Word	chicken	indus	botkin
Target	Chicken	Wendy's	bitcoins

Note: Within the transliterator, alternative sound mappings are shown with an underbar; for example, Cyrillic т can map to English [t] or [θ], which is indicated by t_T. For consonant-mapping, the symbol @ represents any vowel.

After identifying the English out-of-dictionary words, "bitcoin" is added to the CMU dictionary, along with its SONIC-provided pronunciation "B IH T K OY N". Then, when considering the sound sequence b@tk@n, the transliterator is able to select the word "bitcoin" as the output.

[4] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

Translation of Named Entites via Transliteration Mining

Transliteration mining was used to create a large list of NE pairs for the translation of Russian OOV words. The Mystem morphological analysis program was applied to Russian text to detect Russian NE words. Transliteration techniques were then used to identify potential English translations from the parallel English text.

A program was written to take the Mystem NE tags and look for a match for each named entity. First, the Cyrillic characters are transliterated into their typical Russian sounds. Then the program examines the English sentence for capitalized words (excluding the common sentence-initial words *A, The, I, And, This*). Any words which do not match the first sound of the Russian word are discarded.

The program uses a permissive matching for the first sound of the word, due to spelling variations in both Russian and English. Russian lacks certain sounds; to indicate these sounds in foreign words, Russian writers substitute similar sounds (Table 30)

Table 30: Sound Mappings for Missing Sounds in Russian

Sound Mappings for Missing Sounds in Russian	
Sound	**Substitutions**
[θ]	т [t]
[h]	х [x], г [g]
[w]	в [v], у [u]

This sound substitution allows us to find the English word, Honolulu, as a match for the Russian word, Гонолулу /gonolulu/, for example.

English spelling requires us to allow other variations. These include common English alternations like *n/kn*, *s/c*, *c/k*, as well as ways that English accommodates foreign sounds (e.g., the letter *j* typically represents [dž] but may also indicate [y] in Spanish words, and the letters, *gi*, may represent [dž] in Italian words like *Giovanni*).

If more than one word satisfies the matching requirement for the first sound, the program chooses the best candidate based on edit distance from the original word, using a Levenshtein distance normalized for the length of the word. For example, given a sentence with the name, Brigitte Bardot, we attempt to find a transliteration match for the word, Бриджит /bridžit/. We find the potential English matches, Brigitte and Bardot. The word, Brigitte, receives a lower edit distance score and is therefore the preferred match (Table 31).

Table 31: Example of Transliteration Matching for Russian OOV Words

Transliteration Matching for Russian OOV Words

Russian	Latinized	English	Normalized Edit Distance
Бриджит	brijit	brigitte	0.375
		bardot	0.667

The components of hyphenated words were considered independently and in combination. The program records the original Russian and English words along with the distance score. About 20,000 potential pairs were considered. Preliminary examination showed that scores below about 0.66 indicate reasonably good NE pairs.

Sentence-alignment errors can lead to spurious transliteration mining pairs, so a length disparity constraint was added to exclude sentences in which the Russian is more than twice as long as the English, or vice versa. An instance count was also added in order to distinguish accidental matches and misspellings from valid NE transliterations. This counts the instances for each English match for a given Russian word, and records the instance ratio as this count divided by the total count for the Russian word. The following algorithm was used to select the most likely NE pairs:

Keep the NE pair if any of the following conditions are met:

a) edit distance = 0 (record these as the most reliable NE pairs)

b) edit distance < 0.2

c) edit distance falls between 0.2 and 0.5, inclusive, and:

sentence length disparity < 2

instance ratio > 0.01

This transliteration mining process generated 32,559 Mystem-tagged NE pairs. A second round of transliteration mining was conducted considering any capitalized Russian word, not just the words tagged as NE by Mystem. A program was written to identify capitalized Russian words, while excluding acronyms, personal pronouns (which are capitalized in some styles of Russian writing), and sentence-initial words. This step generated an additional 22,044 capitalized-word NE pairs, for a total of 54,603 likely NE pairs.

A second program was written to take OOV words in Russian to English MT output, look them up in the NE pairs list, and replace the OOV word with its English counterpart if a translation pair was found. This transliteration mining technique was applied to NE processing for the SCREAM Lab Russian-to-English MT submission for the WMT2015 competition. The list of 54K likely NE pairs was also provided to colleagues at MIT-LL for training data for a neural net transliterator for Russian to English OOV words.

The edit distance calculation for Russian requires stemming. The addition of Russian inflectional endings can make a Russian word differ from its uninflected English match. The lemma of the Russian word is therefore used when calculating the edit distance. For example, the instrumental form, Африкой “Africa”, is lemmatized to Африка “Africa” before matching the word with English candidates (Table 32).

Table 32: Example of Transliteration Matching, Based on Lemma, for Russian OOV Words

Transliteration Matching for Russian OOV Words

Russian	Lemma	Latinized	English	Normalized Edit Distance
Африка	африка	afrika	africa	0.167
Африкой	африка	afrika	africa	0.167

This creates a problem with stems that happen to end in potential inflectional sequences. For example, the name, Адама "Adama", looks to Mystem like the name, Адам "Adam", with the Russian inflectional ending, - а (gen/acc). Mystem postulates a lemma that is further from the English in this instance, potentially preventing the harvest of this NE pair (Table 33).

Table 33: Potential Lemma Issues in Transliteration Matching for Russian OOV Words

Transliteration Matching for Russian OOV Words				
Russian	**Lemma**	**Latinized**	**English**	**Normalized Edit Distance**
Адама	адама	adam	adama	0.2

Selective Transliteration Based on Capitalization of Source Word

As discussed in an earlier section, the appropriate treatment of OOV words depends on the reason the word is out of vocabulary. Since transliteration is generally helpful for named entities but not for common words, a program was written to analyze Russian OOV words by looking at the form of the word in the original Russian file. Lowercased words are discarded, and uppercase words that are not sentence-initial are considered to be names, and are transliterated via letter-mapping.

For example, in the following sentence, the originally-uppercase OOV names, Кортин /Kortin/ and Патеком /Patekom/, are transliterated, while the lowercase words, 39-летних "39-years" and дочерей-близнецов "daughter-twins", are deleted. The relevant words are highlighted in yellow in the examples (Table 34).

Table 34: Example of Capitalization-Based Selective Transliteration of Russian OOV Words

Capitalization-Based Selective Transliteration of Russian OOV Words	
Original Russian with Uppercase Names:	Это решение усугубило болезненные воспоминания матери из Перта Джун Кортин , которая потеряла своих 39-летних дочерей-близнецов Джейн и Дженни в теракте , осуществленном Патеком и его сообщниками почти десять лет назад .
Initial MT Output with Russian OOV Words:	This decision has painful memories of the mother from Perth June кортин , which lost its 39-летних дочерей-близнецов Jane and Jenny in the terrorist attack , effected патеком and helped him nearly ten years ago .
Selective Transliteration:	This decision has painful memories of the mother from Perth June Kortin , which lost its Jane and Jenny in the terrorist attack , effected Patekom and helped him nearly ten years ago .

In this example, the names are not recovered completely, since the transliteration of Kortin does not match the actual spelling, Corteen, and because the name, Patek, occurs with an instrumental case ending. However, having the transliterated names provides some information and may improve readability.

Looking at a Russian-to-English MT output file of approximately 10,000 lines, the following distribution was found:

- 4407 OOV words
- 1874 regular word OOVs (including 228 sentence-initial uppercase words)
- 2290 NE OOVs

When the OOV word occurs at the start of a sentence, it may be possible to use other instances of the same OOV word to determine its status. In the following examples shown in Table 35, we see that we can distinguish the sentence-initial capitalized OOV words, Расселить “resettle” and Магалуф “Magaluf (name of a resort)”, by their use in other sentences in which they are not sentence-initial.

Table 35: Example of Sentence-Initial-Based Selective Transliteration of Russian OOV Words

Sentence-Initial-Based Selective Transliteration of Russian OOV Words	
Source	**Ref**
Расселить людей из будущего музея Владимир Барабаш обещает в 2015 году.	Vladimir Barabash promises that the resettlement of those people living in the future museum will happen in 2015.
Жильцов в скором времени планируется **расселить** по новым квартирам.	It is planned to resettle the residents in new apartments in the near future.
Магалуф попал в заголовки международной прессы этим летом [...]	Magaluf made international headlines this summer [...]
Туристический курорт **Магалуф**, главным образом, популярный среди молодых британских отдыхающих, [...]	The tourist resort of Magaluf, mainly popular with young British holidaymakers, [...]

Statistical Transliteration as a Post-Process

An English-to-Russian MT system was created using the Moses statistical transliteration capability to recover NE OOV words as a post-process. For example, the NE, *LaForge*, was plausibly transliterated as Лафорж /Laforʒ/ in the following sentence:

> Original English-to-Russian MT Output: " мы хотим , чтобы наши студенты вернуться " , - сказал laforge .
>
> Transliterated: "мы хотим, чтобы наши студенты вернуться", - сказал Лафорж.

2.1.3.16 Techniques in Domain Adaptation for MT

Domain adaptation leverages existing MT resources to adapt a general model to a specific domain.

Using Meta-Data

The meta-data supplied for the TED Talks and for the WMT Russian/English test files provide the potential to train separate language models or translation systems on different domains. These domain adaptation efforts are described here. Other possible sources for domain adaptation include the IWSLT UM (Macau) corpus sections, the HindEnCorp corpus sections, and the various sections and keywords found in online Russian news sites. These resources are described further in a later section, 2.2.4.3 “Meta-Data”, dealing with grammatical annotation of corpora.

TED Talk Annotations for Speaker, Translator, and Date

The TED Talks are provided with meta-data listing URL, the speaker, and the translator if applicable. From the URL it is possible to determine the date each talk was filmed. An examination was made of the distribution of talks by date, in consideration of the possibility that different test sets might benefit from training data taken from similar years. For example, the distribution of Chinese/English talks is shown below in Table 36.

Table 36: Distribution of Chinese-English *TED Talks* Datasets by Year

Chinese-English TED Talks										
File	**2006**	**2007**	**2008**	**2009**	**2010**	**2011**	**2012**	**2013**	**2014**	**2015**
dev2010	1	2	1	4						
tst2010					11					
tst2011						14				
tst2012							12			
tst2013							12	7		
tst2014							1	11		
tst2015									11	1
train2015	49	116	183	211	220	228	243	217	209	41

Next, the talks were analyzed by translator. A program was written to record the BLEU score, total lines, total words, translator ID and name, document ID and URL, and source file for each individual talk. When applied to the IWSLT test files, this displayed a wide range of BLEU scores for individual talks within a single test file. For tst2011, for example, the BLEU scores range from 9.95 to 20.65 (Table 37). Talks by the same translator did not necessarily have similar BLEU scores, suggesting that the inherent difficulty of the talk might have more influence than the person creating the translation.

Table 37: Translation Results for Individual Talks, *TED Talks* (tst2011)

Translation Results, TED Talks (tst2011 Dataset)				
BLEU	**talkid**	**URL**	**Translator ID**	**Name**
20.65	1104	eythor_bender_demos_human_exoskeletons	495543	Felix_Chen
9.95	1096	mark_bezos_a_life_lesson_from_a_volunteer_firefighter	193561	Coco_Shen
16.66	1102	isabel_behncke_evolution_s_gift_ of_play_from...	220760	Angelia_King
12.36	1166	alice_dreger_is_anatomy_destiny	831361	Jiwei_Qu
11.21	1161	jessi_arrington_wearing_nothing_new	925579	Ann_Lee
11.64	1137	carlo_ratti_architecture_that_senses_and_responds	221131	Ralph_Jin
15.57	1171	camille_seaman_haunting_photos_of_ice	220760	Angelia_King
17.01	1115	mick_ebeling_the_invention_that_unlocked_a_locked...	495543	Felix_Chen
17.24	1176	jok_church_a_circle_of_caring	221131	Ralph_Jin
13.53	1107	ralph_langner_cracking_stuxnet_a_21st_century...	495543	Felix_Chen
10.02	1114	morgan_spurlock_the_greatest_ted_talk_ever_sold	354776	Lili_Liang
15.44	1144	amit_sood_building_a_museum_of_museums_on_the...	250727	Jenny_yang
16.80	1160	aaron_o_connell_making_sense_of_a_visible_quantum...	221131	Ralph_Jin
12.15	1165	paul_romer_the_world_s_first_charter_city	221131	Ralph_Jin

Another program was created to collect all the talks by a particular translator and generate a combined score (Table 38).

Table 38: Translation Results Aggregated by Individual Translator, *TED Talks* (tst2011)

Translation Results Aggregated by Translator (tst2011 Dataset)			
BLEU	**Translator ID**	**Name**	**Set of Talk IDs**
13.62	221131	Ralph_Jin	1137, 1176, 1160, 1165
16.33	495543	Felix_Chen	1104, 1115, 1107
16.72	220760	Angelia_King	1102, 1171

An examination of the distribution of talks by translator in the IWSLT training files showed that there are not enough talks to train individual MT systems for different translators (Table 39).

Table 39: Distribution of Talks by Translator, *TED Talks* (IWSLT)

TED Talks (IWSLT Dataset)				
Translator	**Test Docs**	**Lines**	**Training Docs**	**Lines**
Ralph_Jin	4	344	12	1346
Dennis_Guo	2	330	4	429
Felix_Chen	3	235	29	3248
Jenny_Yang	3	192	34	3995
Psycho_Decoder	2	167	26	2232
Lin_Piao	2	167	7	1022
Lee_Li	2	147	21	2046
Emma_Zhao	2	122	14	1806
Angelia_King	2	84	41	4373

Instead, an attempt was made to use a particular translator's training talks as a dev set to tune the MT system to that translator's style. For example, Felix Chen translated 29 of the training documents. A dev set can be created using these 29 documents, and the set of 1688 documents that were not translated by Ralph can serve as the restricted training set.

Results are reported below in Table 40 for two translators. The first line shows the system trained on all the data, and the second line shows the data trained on the restricted training set.

Table 40: MT Results on *TED Talks* After Translator-Specific Tuning

MT Performance on TED Talks Tuned to a Specific Translator				
		dev2010	**dev=Felix**	**dev=Lili**
Translator Felix (495543), test documents 1104, 1115, 1107	Train (all)	15.89	15.29	
	Train minus Felix	13.29	14.08	
Translator Lili (354776), test documents 1104, 1115, 1107	Train (all)	9.89		8.64
	Train minus Lili	8.77		8.99

When training on the restricted training data, there was an improvement for both translators in tuning on the held out data instead of dev2010. However, this tuning improvement is not enough to offset an overall drop in score from the reduction in training data.

These systems were also tested on the complete tst2011 file (Table 41). Here, there was an expected drop in score when restricting the training data (Column 1); and once again there was an improvement in score when tuning on one of the tst2011 translators instead of tuning with dev2010 (Rows 2 and 3).

Table 41: MT Results on *TED Talks* After Translator-Specific Tuning (tst2011)

MT Performance on TED Talks Tuned to a Specific Translator (tst2011 Dataset)				
		dev2010	**dev=Felix**	**dev=Lili**
Translators Felix and Lili, tst2011 file	train(all)	16.70	13.39	11.25
	train minus Felix	13.94	14.90	--
	train minus Lili	14.84	--	15.10

Similar improvements with translator-specific tuning were seen for dev2010, tst2012, and tst2013, even though those test sets do not contain talks by these particular translators, suggesting that the cause of the improvement is not strictly due to matching translator style.

Language of Origin and Word Order for WMT 2015 Russian/English Test Files

The WMT 2015 test files for Russian to English translation are annotated with origlang-"ru" or origlang="en", indicating whether there is a Russian article that was translated into English, or vice versa. A program was written to sort the files into Russian-originated and English-originated files. There were 74 Russian-originated articles, with 1385 lines of text, and 48 English-originated articles, with 1433 lines of text. The vocabulary of each set was examined. The Russian-source vocabulary contained certain words not found in the English-source vocabulary, such as РФ *RF=Russian Federation*, рублей *rubles*, санкции *sanctions* and отношении *against* (found in the phrase, *sanctions against Russia*). The distribution of these

terms makes sense, as English-source articles refer to Russia as opposed to the RF, are not likely to discuss rubles, and are less likely to discuss sanctions than the Russian-source news articles.

Meanwhile, the English-source vocabulary contained some words not found in the Russian-source vocabulary. These included words specific to certain article topics (асбеста *asbestos* and мезотелиома *mesothelioma*). Surprisingly, this list also contained some common words like школе school and ходить *walk/go*. Further investigation showed that these were also accidental gaps, due to the subject matter of particular articles.

Word order was also examined. Because of the relative freedom of Russian word order, there might be a greater variation of word orders when the original language was Russian. The Malt Parser was applied to generate dependency parsers for each Russian sentence, and a program was written to count the resulting instances of subject-verb (SV), subject-verb-object (SVO), etc. There was a larger proportion of VS sentences in the Russian-source articles than the English-source articles, but the overall pattern was that both data sets exhibited more SV and more SVO patterns than other orderings (Table 42).

Table 42: Word Order in English and Russian Sources (WMT 2015)

Word Order (WMT 2015 Dataset)	
origlang-en 1797 sentences	**origlang-ru** 1772 sentences
S 0	S 0
V 612	V 601
O 0	O 0
SV 398	SV 373
VS 115	VS 200
OV 20	OV 25
VO 77	VO 96
SO 0	SO 0
OS 0	OS 0
SVO 481	SVO 378
SOV 5	SOV 6
VSO 4	VSO 5
VOS 15	VOS 19
OSV 6	OSV 10
OVS 64	OVS 59

The genre annotations were also examined, for possible domain adaptation. All 122 articles in the test set were annotated as genre="news", but the test data contained one article with text from a novel, Howard Jacobson's “J”. Most of the other articles were in fact news or economics articles; there were some sports articles, and just a handful of "lifestyle" articles about celebrities, music, food, or gardening.

Similarity Measures Including Tversky Score

A comparison was made between the IWSLT 2013 and 2014 TED Talks, looking for patterns in the new talks added for 2014. The number of unique words was considered as a measure of novelty. Counts are shown here for the Russian/English parallel data (Table 43). Russian words were stemmed to reduce variation.

Table 43: Tallies of Unique Words in Russian-English *TED Talks* (*IWSLT 2013, 2014*)

Unique Words in Russian-English TED Talks (IWSLT 2013, 2014 Datasets)			
Language & File	**Train**	**2013 Total**	**New Talks Total**
English, ruen-en	46170	30033	8359
Russian, ruen-ru	124628	73826	24440
Russian, ruen-ru-stem	59551	36670	10523

The Tversky Similarity Ratio [22, 23] was identified as an easy-to-implement measure for file comparison. The Tversky calculation compares the list of unique words in each file, and generates scores ranging from 0 to 1, with larger scores indicating greater similarity.

Tversky Similarity Ratio:

$$c / (c + a + b)$$

where

c = words common to both files

a = words only in file a

b = words only in file b

Factors influencing the score include the size of the files, and anything that causes a word to be distinct from its other occurrences (e.g., tokenization, casing, morphological inflection, etc.) A program was written to implement the Tversky similarity measure. Files were tokenized and lowercased before comparison. Russian words were considered both stemmed and unstemmed; Chinese sentences were word segmented with the Stanford parser. The resulting Tversky Similarity Scores are shown in Table 44.

.

Table 44: Tversky Similarity on Files in *IWSLT 2013* and *2014* Datasets

Tversky Similarity, By Document (IWSLT 2013, 2014)							
Document		**dev2010**	**tst2010**	**tst2011**	**tst2012**	**tst2013**	**tst2014**
en-fr.en	tst2013	0.240945	0.269075	0.245869	0.26078	---	0.260117
	tst2014	0.246173	0.268038	0.240481	0.255735	0.260117	---
fa-en.fa	tst2013	0.247317	0.269144	0.277926	0.281714	---	0.261051
	tst2014	0.240326	0.256177	0.254111	0.255617	0.261051	---
ar-en.ar	tst2013	0.16088	0.167099	0.16222	0.171454	---	0.164756
	tst2014	0.155571	0.16996	0.161447	0.164815	0.164756	---
ru-en.ru	tst2013	0.156182	0.172778	0.172103	0.178933	---	0.173167
	tst2014	0.167115	0.177225	0.17713	0.178305	0.173167	---
ru-en.ru (stem)	tst2013	0.237016	0.251215	0.251444	0.260194	---	0.257688
	tst2014	0.250654	0.259407	0.257806	0.263692	0.257688	---
zh-en.zh (seg)	tst2013	0.234605	0.260348	0.241389	0.245556	---	0.256373
	tst2014	0.237199	0.249964	0.243914	0.245787	0.256373	---

2.1.3.17 Experiments in Pivot Methods for MT

Ukrainian to Russian to English

Pivot methods refer to the use of MT into and out of an intermediate language; generally, pivot methods are applied when there are limited resources to train a system directly between the two outer languages. In order to build a translation capability from Ukrainian to English, a pivot method was applied to take advantage of, both, the similarity of the Russian and Ukrainian languages, and the strong existing SCREAM Lab Russian-to-English MT resources.

Bilingual online news sites were used to compile parallel text for the Ukrainian-to-Russian step. Preliminary output of the Ukrainian>Russian>English pivot process showed promise (Table 45). The similarity of Russian and Ukrainian yielded a high BLEU score for the initial step. The overall pivot score is good, although not as good as a direct translation from the original Russian into English.

Table 45: Preliminary Translation Results for Ukrainian-to-English Pivot Process

Translation Results, Ukrainian-Russian-English	
	BLEU
Ukrainian>Russian	0.5579
Ukrainian>Russian>English	0.22
Russian>English	0.28

A qualitative analysis of a sample news article showed that the sense of the article was generally preserved, although there were some mis-translations, such as the shift from the word "freed" to the word "fired":

Ukrainian: Полонених військових уже звільняють – Турчинов

Ukrainian>Russian: ленных военных уже увольняют – турчинов

Russian>English: Prisoners of war have already fired - Turchinov

The mis-translation occurred in the first step, when Ukrainian звільняють was translated as Russian увольнять “cashier, dismiss, turn away”. The more appropriate Russian word for this context is освобождать “free, liberate, rescue”.

An interesting effect of the pivot method is that unknown words in the first step persist in their Ukrainian spelling through to the English, so the English output contains both Ukrainian and Russian OOV words. An analysis of the Ukrainian OOV words showed primarily common words, a few NE, and some inflected NE that could potentially be recovered through stemming. Additional parallel text might reduce the number of common word OOVs in the Ukrainian>Russian step, while transliteration might help with the NE.

In support of the pivot translation, several general tools were developed for working with Ukrainian text. The Moses tokenizer was revised to recognize the use of apostrophe in Ukrainian to indicate a non-palatalized consonant. Normalization was applied for punctuation variants. The Russian non-breaking prefix list was adapted to create a Ukrainian list, adding the four characters that are specific to Ukrainian.

The previously created Russian-to-English letter-based transliteration program was adapted to Ukrainian by adding Ukrainian-specific characters and making adjustments for characters which have different pronunciations in Ukrainian than in Russian. Initial testing shows that the transliterator often turns borrowed words into sequences that are not correct English, but are still phonetically understandable, as shown below in Table 46.

Table 46: Initial Results of Letter-Based Transliteration of Ukrainian

Output from Letter-Based Transliterator Operating on Ukrainian		
Ukrainian	**Letter-Map**	**Target**
Джиммі Стюарт	Dzhimmi Styuart	Jimmy Stewart
Сідні Пуатьє	Sidni Puatye	Sidney Poitier
Вашингтона	Vashingtona	Washington

Testing also shows the need to stem Ukrainian forms before applying transliteration. For example, the transliteration of *Vashingtona* "Washington" would be improved if the case ending *–a* was removed before transliteration. An existing program, Stemka, was identified and applied to stem Ukrainian inflected forms.

A separate transliteration program was written to convert Ukrainian characters to their corresponding Russian characters. Much of the alphabet is the same, but there are characters specific to Ukrainian that need to be mapped into Russian characters, and there are shared characters that have different sounds in the two languages. In particular, the languages differ in their use of hard and soft signs to represent the palatalized quality of consonants.

Synthetic Arabic Data

Back translations were used to create additional, synthetic data for MT systems. An existing Systran English-to-Arabic MT system was used to translate monolingual English data in to Arabic, creating parallel English/"Arabic" training data to supplement existing training data for the Arabic-to-English MT system under development. The output of this MT process is described in "Quality Control of Synthetic Data" under section 2.1.3.20 "Error Analysis of MT Output".

2.1.3.18 MT System Comparison Testing

Comparing Joshua Rule Tables and Moses Phrase Tables

The SCREAM Lab research included two main MT systems; Joshua and Moses. In order to compare these systems, a program was written to convert from the Joshua grammar format to the Moses rule-table format. In Joshua, non-terminal elements are indicated with indices (e.g., [X,1]) that relate the position of the non-terminal on the left hand side of the rule with the position of the non-terminal on the right hand side of the rule. The Moses format does not include indices; instead, relationships among the non-terminals are indicated by alignment values (e.g., 2-2).

The Moses runs generate alignments for all of the matching elements, as shown below, while Joshua only aligns non-terminals. In addition, the example rule-table has a third set of numbers, containing optional frequency counts. Here is an example of the conversion of Joshua grammar entries to the Moses rule-table format:

> Joshua Grammar: [X] ||| вся политика ||| all politics ||| 3.60390 6.66101 1 1.00000 0.69315 0.69315

[X] ||| вся политика [X,1] ||| all politics [X,1] ||| 3.60390 6.66101 1 1.00000 0.69315 0.69315

Derived Moses Rule-Table: вся политика [X] ||| all politics [X] ||| 0.0272173674567845 0.00127985306577889 0.367879441171442 0.367879441171442 0.49999859028196 0.49999859028196 |||

вся политика [X][X] [X] ||| all politics [X][X] [X] ||| 0.0272173674567845 0.00127985306577889 0.367879441171442 0.367879441171442 0.49999859028196 0.49999859028196 ||| 2-2

Original Moses Rule-Table: [X][X] всё политика [X] ||| [X][X] all politics [X] ||| 0.164461 0.0154115 0.396439 0.0953937 2.718 ||| 0-0 1-1 2-2 ||| 2.72619 1.13095 1.13095

The Joshua system provides a script to convert a Moses rule-table to the Joshua grammar format.

Work was also done to adapt the Joshua 5 update to the SCREAM Lab systems; the Joshua pipeline script was revised to incorporate Moses tokenization with HTML escaping of special characters, in order to match existing language models, and to allow the use of multiple language models.

Statistical Post-Editing with Different MT Systems

Statistical post-editing applies a second MT system to the output of an initial translation. The second system is trained to translate from one form of the target language to an improved form of the target language. It is trained on held-out training data, in order to build a system that translates from the initial MT output to the reference data.

For Russian-to-English translation, an initial translation was made via Systran, and then a Joshua system was trained improve the Systran output. This increased the BLEU score about 7 points over the Systran baseline, although it did not achieve the same level as a Joshua direct translation baseline (Table 47).

Table 47: Translation Results for Russian-to-English Using Statistical Post-Editing

Translation Results, Russian-to-English	
Joshua baseline, ru>en	0.2992 BLEU
Systran baseline, ru>en	0.1744 BLEU (case-insensitive scoring)
Systran + Joshua	0.2459 BLEU (both files lowercased, Moses tokenized)

Some errors were corrected for the Systran system. The Systran processing step failed to generate a final line-feed error at the end of the file, causing an error for the Joshua system. Systran was also found to generate some control characters in association with accented characters in its translation memory.

A qualitative review of post-editing output showed some improvement in vocabulary choice as well as some improved syntactic constructions, as highlighted in yellow in this example, below (Table 48). One vocabulary choice moved away from the reference (highlighted in blue).

Another syntactic choice seems more idiomatic, but actually reduced the bigram match with the reference (highlighted in green).

Table 48: Examples Showing Effects of Statistical Post-Editing on Russian-to-English Translations

Russian-to-English Translations	
Systran Output:	The goalkeeper “of Chicago” Nikolai Хабибулин emphasized: he understands, that it must play better. According to the 40-year Russian, it will continue zealous to work during the trainings. In the last two encounters “of the hawks” of Хабибулин first passed six washers from “Tampa Bay”, and then four - from “Ottawa”, moreover in the second game it was substituted after the fourth passed washer.
Joshua Post-Editing Output:	the goalkeeper of " chicago " nikolai хабибулин stressed : he understands that he must play better . according to the 40-year-old russian , it will continue to diligently work during trainings . in the last two meetings of the " hawks " of хабибулин first passed six washers from " tampa bay " , and then four - from " ottawa , " moreover in the second game it was replaced after the fourth passed the puck .
English Reference:	The goalkeeper of "Chicago" Nikolai Khabibulin emphasized: he understands that he must play better. According to the 40-year-old Russian, he will continue to work hard at practices. At the last two meetings of the "Hawks" Khabibulin initially let through six pucks from "Tampa Bay", and then four from "Ottawa", though in the second game he was replaced after the fourth missed puck.

An alternative post-editing approach was considered, in which the unknown words are dropped from the Systran output prior to post-editing with Joshua. Preliminary scores here were lower than the version which retains unknown words.

Some problems were found with the way Systran marks unknown words. Systran uses the tag, <nfw>, for "not found words", but does not tag unknown words when they are adjacent to a translated personal name. Systran also does not tag words in other alphabets. Finally, in some cases Systran was able to identify a possessive construction, which it translated into English with an apostrophe-s, but it retained the unknown Russian noun, as shown below (

Table 49). The relevant words are highlighted in yellow in these examples.

Table 49: Example Showing Unknown Word Anomalies in *Systran* Russian-to-English Translation

Systran Russian-to-English Translation	
Russian Input:	Мексика добилась хороших экономических показателей при администрации Фелипе Кальдерона, которая сейчас складывает свои полномочия, однако страна погрязла в войне с наркотиками, которая за шесть лет уже унесла порядка 60.000 жизней.
Systran Output:	Mexico attained good economic indices with Felipe Кальдерон's administration, who now adds her authorities; however, the country of in the war with the narcotics, which in six years already took away order 60.000 lives.
English Reference:	Mexico has been performing well economically under the outgoing administration of Felipe Calderon, but the country is in the grip of a drug war, which has already claimed an estimated 60,000 lives in six years.

Constituent Parsing vs. Dependency Parsing

Constituent parsing can be used to create tree-to-tree or tree-to-string MT training data. For Russian, however, researchers generally create dependency parses instead of constituent parses, since the dependency parse is more tolerant of word order variations (see section 2.1.3.8 "Dependency Parsing to Change Russian Word Order for MT"). A method was developed to adapt dependency parses into the constituent parses that can be used within the Moses system.

First, the Russian sentence is tokenized and POS tagged, and the tagger output is re-written in CoNLL format. The Russian Malt Parser creates the dependency parse from the tagger output. The dependency parses are converted into constituent structures using the following algorithm:

a. For each word, i, in the dependency chart, create a node with the index, i*2. This creates spaces between the nodes in the index.

b. For each word, i, that is listed as a head word in the chart, propagate a non-terminal (X') node with the index, (i*2) +1. This indexes the X' nodes immediately after their terminal nodes. Copy the POS from the terminal node. Record the vertical link between the terminal node X and its X' node.

c. For each terminal node, Y, create a link to its head node, X', following the indices in the dependency chart. For example, if the word, *alpha*, has *beta* listed as its head word, we link the terminal node, Y, for *alpha* to the X' node above the terminal node, X, for *beta*.

d. Finally, consider whether any of the Y nodes in the last step have their own non-terminal Y' nodes. In this case, change the link from (Y,X') to a link from the parent node: (Y',X')

An illustration: Suppose we have the following dependencies shown in Table 50.

Table 50: Constituent Parsing Example; Dependency Parse of Russian Sentence

Dependency Chart		
Index	**Word**	**Head-Index**
1	alpha	2
2	beta	3
3	gamma	0 (ROOT)

First we create nodes with spaces between them in the index (Table 51):

Table 51: Constituent Parsing Example; Primary Node Chart

Node Chart	
Node	**Word**
2	alpha
4	beta
6	gamma

Then, for the words which are heads, we need to create additional tree structure (Table 52):

Table 52: Constituent Parsing Example; Secondary Node Chart

Node Chart	
Node	**Word**
2	alpha
4	beta
5	X' above beta
6	gamma
7	X' above gamma

Now, we can link the nodes this way (Figure 9):

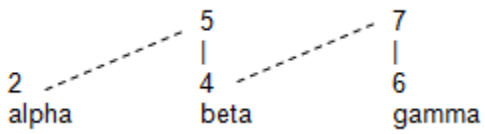


Figure 9: Constituent Parsing Example; Initial Derived Constituent Structure

And finally, we re-assign any links from nodes that have X' nodes themselves, linking instead from the non-terminal node (Figure 10):

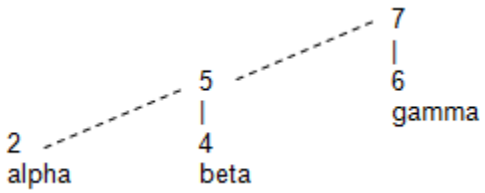


Figure 10: Constituent Parsing Example; Derived Constituent Structure after Link Shifting

The tree structures are then converted into bracketed constituent representations, using this algorithm:

a. Start with the root node.

b. For each node, write the following:

   non-terminal node (POS

   terminal node (POS word

c. Then, take the children of that node in turn and repeat the write-out process.

d. When the node and all its children have been listed, write a closing bracket,).

For the tree diagram above, the resulting constituent bracketing is shown below (using the indices here instead of the POS tags):

(7 (5 (2 alpha) (4 beta))(6 gamma))

Note that the dependency chart may indicate relationships that create crossing lines in a tree structure. Consider the following sentence:

English: Talk about a sweet beginning.

Russian: Более оптимистичное начало трудно представить .

Literal Russian translation: More optimistic beginning difficult to-imagine .

The dependency chart (Table 53 and Figure 11) for this sentence indicates a crossing line where the child of the root, “difficult”, has to be written after the phrase, “more optimistic beginning”.

Table 53: Example Dependency Parse of Russian Sentence

Dependency Parse of Russian							
Index	**Word**	**Lemma**	**POS**	**POS-Detail**	**Morph**	**Head**	**Dependency**
1	Более	более	R	R	R	2	огранич
2	оптимистичное	<unknown>	A	A	Afpnsnf	3	опред
3	начало	начало	N	N	Ncnsan	5	1-компл
4	трудно	трудно	R	R	R	0	ROOT
5	представить	представить	V	V	Vmn----a-p	4	предик
6	.	.	S	S	SENT	5	PUNC

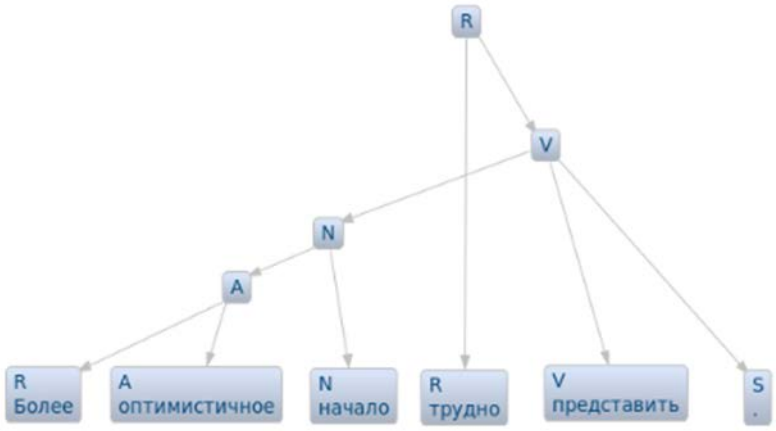


Figure 11: Example Dependency Chart of Russian Sentence

By starting with the root node, the bracket-writing algorithm re-orders the sentence so the root word, трудно “difficult”, comes first:

(R(R трудно)(V(N(A(R Более)(A оптимистичное))(N начало))(V представить)(S .)))

Talk about a sweet beginning.

Literally: Difficult more optimistic beginning to-imagine .

The tree structure that this represents is well-formed, with no crossing lines (Figure 12):

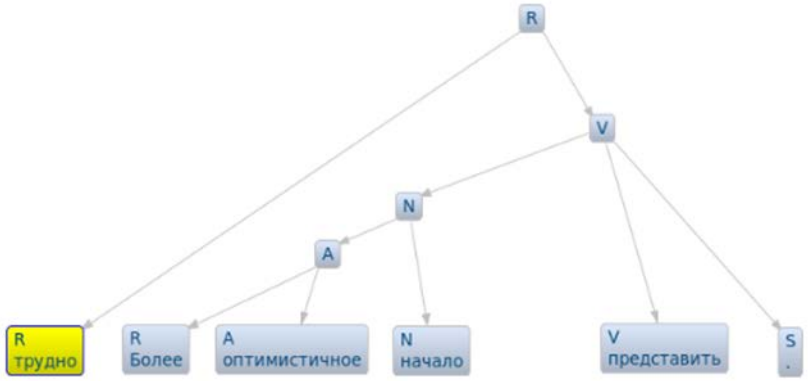


Figure 12 Example Dependency Chart of Russian Sentence After Word Reordering

The write-out algorithm, by considering each child node in order, generally has the effect of re-ordering the dependent words in such a way that the lines no longer cross. This re-ordering may or may not be desirable as input to the MT, but it at least ensures well-formed constituent structures.

Some of the dependency parses contain multiple roots, indicating sentence fragments or other separate pieces of a sentence. In this situation, the conversion program propagates a parent node to link all the root nodes, using the part of speech label of the final root node.

2.1.3.19 Miscellaneous MT Efforts

Other research efforts investigated language modeling, word alignment constraints, sentiment analysis, and recasing. For word alignment, initial exploration was conducted on manually specifying word alignments for known elements, such as seeding alignments in MGIZA++. Also considered was the possibility of annotating source languages in multiple-language text, since the presence of wrong-language alignments and phrase table entries has exhibited a disproportionate effect in our previous MT systems.

An investigation was made of possible data sources for Russian sentiment analysis. The Russian Information Retrieval Evaluation Seminar (ROMIP) provides the 5000-word SentiRus lexicon[5], developed by Chetviorkin and Loukachevitsh (2012) [24] for the product domain. Unfortunately, words in this lexicon are only annotated for their likelihood of being an opinion word, and not for their positive or negative quality. Chetviorkin and Loukachevitch noted in their 2012 article that "no publicly available Russian sentiment lexicon exists", and many resources that are mentioned in the more recent literature are still under development.

A recaser was built to determine the appropriate capitalization of English MT output. The Combined Common Crawl was used as a resource. The existing Russian/English Common Crawl cleanup program was applied to remove non-English sections and correct errors in encodings and mixed-alphabet spellings. A program was written to exclude headlines and other lines containing sequences of all caps that will mislead the recaser.

[5] http://romip.ru/en/collections

2.1.3.20 Error Analysis of MT Output

Phrase Based vs. Hierarchical vs. Neural Systems

Output analysis was conducted on various SCREAM Lab MT Systems. Comparisons were made for different MT systems, including Jane, Joshua, and Moses. Comparisons were made for different types of systems, including phrase-based, hierarchical, and neural systems. The existing programs, Hjerson and MT-ComparEval, were used to compare output files (for more on these programs, see section 2.1.3.21 “Improvements to the Hjerson Error Analysis Program”).

In general, hierarchical systems were found to be more fluent than phrase-based systems. Neural systems tended to be fluent, but had problems with NE.

When a factored phrase-based system was examined, it was found to be more similar to a hierarchical system than to a simple phrase-based system. This similarity showed up in both cross-system BLEU scores and in a measure of sentence similarity (Table 54).

Table 54: Similarity of Factored Phase-Based vs. Hierarchical Systems

Factored Phrased-Based vs. Hierarchical Systems			
WMT2015 System	**BLEU**	**Identical Sentences**	**75% Matching Sentences**
hi9r2 vs. pb6r8 (factored)	0.7018	533	1019
pb11r0 vs. pb6r8 (factored)	0.6047	212	578
hi9r2 vs. pb11r0	0.5709	157	478

The output of neural systems often had excessive repetition, in which an otherwise appropriate word or phrase was repeated many times within a sentence. For example, a neural translation translated one Arabic phrase as "and they would pay it for themselves", which is fairly reasonable compared to the reference, "but they do it for themselves, and they get paid for it". However, the neural translation output repeated this phrase 18 times within the sentence.

Existing SCREAM Lab programs were used to identify sentence-internal repetition (see section 2.2.3.2 “IWSLT 2014 TED Talks”). For a neural system involved in system combination, a program was written to replace such repetitive lines with the output of one of the other contributing systems. For example, a neural system was used in a system combination translation of the file tst2013 from Arabic to English. When flagging phrases of 10 or more words, the repetition correction program replaced 5 out of the 1169 lines. The replacement of just these few lines improved the BLEU score on the test set by 0.38 BLEU points.

The different system types vary in their handling of unknown named entities. For the phrase-based systems, there was some substitution of first and last names, suggesting a possible alignment error in the training data. For example, where the Russian input had just the last name of the composer, Мендельсона (Mendelssohn + genitive case), the English output of the phrase-based system had just the first name, *Felix*.

The phrase-based systems that applied lexical approximation sometimes replaced unknown NE with similarly spelled names; for example, *Downey* was translated as *Duffy*, and *Ayers* was

translated as *Myers*. Neural systems which used byte pair encoding (BPE) also tended to substitute similar sounding names, e.g., *ginger* for *Genentech*, *Beck* for *Peek, Alpha* for *Alvin*.

Neural systems without BPE had a tendency to substitute unrelated names for unknown NE. One source for such errors would be if the NE occurs in similar environments in the training data. In this example, the phrase-based system had better coverage for the NE, translating *laos* appropriately, while the neural system substituted *tehran* and *iran*, as shown by the yellow highlighted words.

> Reference: we made it all the way to the border of laos , but i had to spend almost all my money to bribe the border guards in laos .
>
> 2-mos-pb-base: we cut the road all the way to the border of laos , but i had to spend most of the money a bribe border guards in laos .
>
> 1-nem-farasa-sw: we cut the road down to tehran , but i had to spend most of my money to bribe the border guard in iran .

Combining MT Outputs into an Oracle Document

When comparing two MT systems, a program was written to compare one sentence at a time against the reference and select the better-scoring output to create an oracle document. This program then outputs the document-level BLEU score for the two output files and the oracle file. For an Arabic-to-English data set using one neural network joint model (NNJM) and one non-NNJM model, the oracle document outperformed the original documents by one BLEU point.

The average number of words per line was slightly higher when the NNJM system was chosen. A program was written to create a second output document in which the choice of sentence is made on the basis of sentence length, without reference to the BLEU score. This output did not out-perform the original NNJM output.

Contribution to Error Rates by Word Frequency

A calculation was made to determine how much high frequency words contribute to the error rate in the MT output. Previous work in the SCREAM lab showed that, for ASR, high frequency words contribute the largest proportion of recognition errors; in this situation, techniques that improve results for the small set of high frequency words may be effective in improving overall performance. For the MT analysis, word classifications were calculated from an English language model, yielding 73 high frequency words, 40799 mid frequency words, and 59130 low frequency words. The SCLITE program was used to align the MT output with the reference and count errors of insertion, deletion, and substitution.

The frequency analysis shows that, unlike ASR, the MT output derives an equal amount of error from the high frequency and mid frequency words.

Table 55 reports overall error rate (e.g., instances of high frequency word errors divided by the total number of words). For the file tst2013, the 73 high frequency English words contribute 31.9% of the overall error rate, as do the 40799 middle frequency words, while the low frequency words contribute just 8% of the overall error rate.

Table 55: Effect of Word Frequency on MT Word Error Rate

Word Error Rate (tst2013, English)				
Freq	**Words**	**Instances**	**Error Count**	**Error/Total**
High	73	12,116	8808	31.9
Mid	40,799	10,936	8814	31.9
Low	59,130	4,534	2217	8.0
Total:		27,586		

Comparison of SCREAM Lab Systems with Other Systems in the Workshop on Statistical Machine Translation (WMT) Competitions

After the Workshop on Statistical Machine Translation (WMT) competitions (Bojar et al. 2014, Bojar et al. 2015, and Bojar et al. 2016) [25, 26, 27], the MT outputs of SCREAM Lab systems were compared to that of competitors, with an aim to understanding and improving errors. These comparisons focused on transliteration and uppercased words.

Transliteration

An examination of the WMT 2014 Russian-to-English output showed differences in the transliteration of OOV words and in the treatment of hyphenated words. A review was made of OOV words, comparing the SCREAM Lab system (AFRLv8) to the top two systems, from the University of Edinburgh and the Russian company, Yandex. The AFRLv8 system is a phrase-based system with rule-based transliteration of unknown words. The top systems apparently used inflection stemming, which enables an MT system to relate inflected forms to previously seen forms; this is useful in working with Russian, because the training data may not contain all the various noun and verb forms. All three systems could correctly translate the uninflected name, Блэкберн, as Blackburn, but the AFRLv8 system was unable to translate the inflected form, Блэкберне, returning instead a sound-based transliteration (Table 56, line 1).

The other systems apparently removed hyphens, allowing them to translate the component word, "percent", in 60-процентной (Table 56, line 2). These systems also appeared to have statistical transliteration of unknown words, while the AFRLv8 system used a rule-based mapping, leading to more consistent output, as in this example for the surname, Дитта *Ditta* (Table 56, line 3).

Table 56: *WMT 2014* Systems Comparison on Inflected Words, Hyphenated Words, and Unknown Words, Russian-to-English

Comparison of Russian-to-English Translations from WMT 2014 Systems			
Source	**AFRLv8**	**Edinburgh**	**Yandex**
1. Блэкберне	blekberne	Blackburn	Blackburn
2. 60-процентной	60-protsentnoy	60 percent	60 percent
3. Дитта	Ditta	Ditta, Dita	DITT, the ditch, Dita, Diet, Ditta

Some words showed up as OOV in the SCREAM Lab MT output, despite the presence of sufficient evidence for their translations in the training data. Further examination showed that these reflect a problem in phrase extraction, in which the unknown word had in fact been aligned to the target word, but extraction was prevented by the presence of several additional, spurious alignments.

Words Written in All Capital Letters

Comparisons of Russian-to-English WMT output for 2015 and 2016 revealed differences in the treatment of uppercase and lowercase words. MT systems often process data in lowercased form, and then apply a final truecasing step to restore uppercase to some letters, using a program that has been trained on examples of typical case usage.

For WMT 2015, we compare the systems from Johns Hopkins University (JHU), the University of Edinburgh (EDIN), and the SCREAM Lab system, AFRL-H. System AFRL-H, also known as hi9r2, is a hierarchical system with selective transliteration of unknowns. All-caps training data was identified as a source of incorrect truecasing for the AFRL-H system. For example, in Table 57 below, the phrase, "TEMPORARILY BLOCKS LAW THAT WOULD CLOSE", should not be capitalized; the capitalized translation in AFRL-H derives from all-caps words in the data used to train the truecaser.

Table 57: *WMT 2015* Systems Comparison Showing Spurious All-Caps Output, Russian-to-English

Comparison of Russian-to-English Translations from WMT 2015 Systems	
SRC:	Судья временно блокирует закон, который мог бы закрыть все абортные клиники в Луизиане
REF:	Judge temporarily blocks law that could close all Louisiana abortion clinics
JHU:	The judge temporarily blocks a law that could close all abort clinic in Louisiana
EDIN:	The judge temporarily blocks a law that would close all абортные clinic in Louisiana
AFRL-H:	Judge TEMPORARILY BLOCKS LAW THAT WOULD CLOSE all clinics in Louisiana

An analysis of the JHU and EDIN systems shows that they only output all-caps phrases when there is an all-caps phrase in the Russian source sentence, as shown in Table 58. In these particular examples, the AFRL-H translation chooses better words, but loses the benefit of those choices when scoring is case-sensitive.

Table 58: *WMT 2015* Systems Comparison on All-Caps Phrases, Russian-to-English

Comparison of Russian-to-English Translations from WMT 2015 Systems	
SRC:	НЕБОЛЬШОЕ ИССЛЕДОВАНИЕ: Новые лекарства могут замедлять рак легких и рак яичников
REF:	SMALL STUDY: New drugs may slow lung, ovarian cancer
JHU:	Small RESEARCH: New drugs can slow the lung cancer and ovarian cancer
EDIN:	Небольшое RESEARCH: New drugs can slow the lung cancer and ovarian cancer.
AFRL-H:	A small study: new drugs may slow lung cancer and ovarian cancer
SRC:	Меня беспокоит то, что если мы не будем готовы, мы обнаружим, что повторяем ошибки, которые изначально привели к ТОМУ, ЧТО ПРОИЗОШЛО, ЕСЛИ ВООБЩЕ ПРОИЗОШЛО.
REF:	My concern is that, if we are not forewarned, we will find ourselves repeating the mistakes that led to WHAT HAPPENED, IF IT HAPPENED, in the first place.
JHU:	I am concerned that if we are not ready, we will find that repeat mistakes that initially led to TU THAT A IF A WAGE.
EDIN:	I am concerned that if we are not ready, we will find that repeat mistakes, which initially led to ТОМУ THAT A RESULT OF A IF A RESULT OF A ВООБЩЕ.
AFRL-H:	I am concerned that if we are not ready, we will find that we repeat the mistakes, which initially led to what happened, if it happened at all.

The case-matching between the source and the JHU and EDIN systems could be the result of a rule-based truecasing procedure that looks back at the Russian sentence. The strange word choice for the all-caps sections suggests that these systems may have been trained on cased data, and therefore lacked all-caps training data for the phrases in question.

The WMT 2015 Common Crawl training files were reviewed to determine the source of all-caps words in the AFRL-H output, in order to consider whether all-caps sections could be excluded from the training data. A program was written that detects a sequence of at least two words that are all uppercase. Single word acronyms are excluded from this analysis.

There were many sources of all-caps words, including, but not limited to, headlines, listed below in Table 59. This would make it difficult to effectively exclude all-caps sequences from the training data.

Table 59: Sources of All-Caps Phrases in the *Common Crawl* Training Data

Sources of All-Caps Phrases in the Common Crawl Training Data
headlines
titles and bylines (often all-caps phrases followed by regular sentences on the same line)
emphasis
brand names
hyperlinks
sports scores
Unicode character descriptions
boilerplate (e.g., credits or copyright notices written entirely in capital letters)
lyrics website (song titles and artists' names are listed in all-caps)
sequences of acronyms

The AFRL WMT 2016 systems were trained on lowercased data, with a subsequent re-casing step, and therefore were not susceptible to this type of all-caps error.

For 2016, we compare systems from Edinburgh (EDIN), Johns Hopkins (JHU), and the SCREAM Lab (AFRL-K and AFRL-J). AFRL-K is a phrase-based system with neural rescoring and neural transliteration of OOV words, while AFRL-J is a contrast system, with cleaned data and a combination of neural and statistical transliteration.

Most of the WMT 2016 systems had difficulty with the capitalization of headlines and acronyms. In the example below in Table 60, the Russian sentence has a sentence case pattern, but the English reference has headline case, leading to case mis-matches with the output sentences.

Table 60: *WMT 2016* Systems Comparison Showing Case-Mismatched Output, Russian-to-English

Comparison of Russian-to-English Translations from WMT 2016 Systems	
SRC:	Глава ООН заявляет, что военного решения в Сирии не существует
REF:	UN Chief Says There Is No Military Solution in Syria
EDIN:	UN chief claims military solution in Syria does not exist
JHU:	The head of the UN says that there is no military solution in Syria
AFRL-K:	The UN chief says that there is no military solution in Syria
AFRL-J:	The head of the United Nations claims that there is no military solution in Syria

Acronyms are another source of case mis-matches. In the AFRL-K system, casing of the output was determined by looking back at the casing of the original Russian word. If a Russian acronym translates to an English phrase, this strategy leads to capitalization of each letter in each word of the phrase. In Table 61, for example, the Russian acronym, США "USA", triggers full capitalization of the output phrase, "THE UNITED STATES".

Table 61: Example of an Acronym-Induced Case Mismatch from *WMT 2016*, Russian-to-English

Russian-to-English Translation (WMT 2016 "AFRL-K" System)	
SRC:	В США федеральный судья в воскресенье временно заблокировал введение в действие закона Луизианы, ...
REF:	A U.S. federal judge on Sunday temporarily blocked enforcement of a Louisiana law ....
AFRL-K:	In THE UNITED STATES, a federal judge on Sunday temporarily blocked enforcement of the law in Louisiana, ...

The AFRL-K system had trouble with *camel case* (words with a mixture of lowercase and internal uppercase letters). The words which caused problems were frequently borrowed English words within the Russian source sentence. If passed through directly, they would retain the same camel case as the original. The AFRL-K system applies capitalization to just the first letter of such named entities, while the AFRL-J system leaves them unchanged (Table 62).

Table 62: Effects of Camel Case on AFRL *WMT 2016* Systems, Russian-to-English

Comparison of Russian-to-English Translation on Camel Case (WMT 2016 "AFRL-J" and "AFRL-K" Systems)			
SRC	**REF**	**AFRL-J**	**AFRL-K**
МакКи	McKee,	McKee,	Mckee,
FedEx	FedEx	FedEx	Fedex
SABMiller.	SABMiller.	SABMiller.	Sabmiller.
iTunes.	iTunes.	iTunes.	Itunes
iOS	iOS	iOS	Ios
Hewlett-Packard	Hewlett-Packard	Hewlett-Packard	Hewlett-packard

A program was written to count the number of times the camel case words from the reference were expressed in the hypothesis with the correct capitalization, expressed with an incorrect capitalization, or were missing. There were 56 camel case words in the reference. Of these, the AFRL-K system had 26 missing and 30 wrongly capitalized (Table 63).

Table 63: Treatment of Camel Case by *WMT 2016* Systems

Treatment of Camel Case Words by WMT 2016 Systems				
	EDIN	**JHU**	**AFRL-K**	**AFRL-J**
OK	33	31	0	37
wrong	2	3	30	10
missing	21	22	26	9

Truecasing and Word Order

As mentioned in the previous section, "Words Written in All Capital Letters", MT systems typically work with lowercased data to create translations, and then apply a truecasing program to uppercase certain letters. System output can be scored by comparing the lowercased output against a lowercased version of the reference (the uncased condition), or by comparing the truecased output against the original reference file. Truecased scores are typically lower than uncased scores.

Truecasing may reduce the number of matching words, if the first word of the output is not the same as the first word of the reference. For example, Table 64 shows a WMT 2016 sentence in which the reference leads with "The meeting", while the MT systems all follow the literal word order of the Russian source and lead with "According to". When comparing lowercased sentences, the phrase, "according to", matches the phrase found later in the reference; after truecasing, however, these no longer match.

Table 64: Example of Word Order Affecting Case Matching in Truecased Output

Comparison of Truecased Russian-to-English Translations from WMT 2016 Systems	
SRC:	Согласно выпущенному после встречи официальному отчету, на ней также присутствовали премьер-министр Ли Кецян и министры Лиу Юньшань и Чжан Гаоли.
REF:	The meeting was also attended by Premier Li Keqiang, and senior leaders Liu Yunshan and Zhang Gaoli, **according** to a statement released after the meeting.
EDIN:	**According** to the official report released after the meeting, Prime Minister Li Keqiang and ministers Liu Yunshan and Zhang Gaoli were also present.
JHU:	**According** to an official report released after the meeting, it was also attended by Prime Minister Li Keqiang and ministers Yunshan Liu and Zhang Gaoli.
AFRL-K:	**According** to an official report released after the meeting, it was also attended by Prime Minister Li Keqiang and Liu and Zhang Ministers.
AFRL-J:	**According** to an official report issued after the meeting, it was also attended by Prime Minister Li Keqiang and Liu and Zhang Yunshan Gaoli ministers.

The truecasing step used by the SCREAM Lab WMT 2016 Russian-to-English systems causes greater score loss for these systems than for other submitted systems, suggesting that there is some problem in the way the SCREAM Lab truecasing is applied (Table 65).

Table 65: Uncased and Truecased BLEU Scores for *WMT 2016* Systems, Russian-to-English

Uncased and Truecased BLEU Scores for WMT 2016 Systems			
System	**Uncased**	**Truecased**	**delta**
EDIN:	28.8	28.0	-0.8
JHU:	28.8	27.9	-0.9
AFRL-K:	28.8	27.6	-1.2
AFRL-J:	28.4	27.0	-1.4

A program was written that records the number of matching first words, the number of additional case-insensitive matches, the number of times a non-matching first word from the reference was found elsewhere in the output file, and the number of times the first word from the output file was found elsewhere in the reference. Tallies for the case where the first word matches the reference are provided in Table 66. Tallies for the case where the first word in the reference is found elsewhere in the output are in Table 67.

Table 66: Case-Specific Word Matching by *WMT 2016* Systems; First Word Matches Ref

Case-Specific Word Matching by WMT 2016 Systems

(First Word Matching the Reference)				
System	**EDIN**	**JHU**	**AFRL-K**	**AFRL-J**
Case-sensitive matches	1771	1676	1673	1596
Additional case-insensitive matches	35	30	40	**109**
TOTAL case-insensitive matches	1806	1706	1713	1705
Percent matching in 2998 lines	60.2	56.9	57.1	56.9

Table 67: Case-Specific Word Matching by *WMT 2016* Systems; First Word in Ref Found Elsewhere in Output

Case-Specific Word Matching by WMT 2016 Systems (First Word in Reference Found Elsewhere in Output)				
System	**EDIN**	**JHU**	**AFRL-K**	**AFRL-J**
sentences with non-matching word1	1192	1292	1285	1293
ref word1 found later in hyp	414	501	487	516
hyp word1 found later in ref	492	523	536	532
TOTAL expected losses if truecasing	906	1024	1023	1048

The program counts the first words found later in the sentence for both reference and hypothesis, because truecasing can result in a deficit each way. Consider these sentence pairs comparing the reference to the JHU output, shown in Table 68. In the first pair, truecasing will cause a loss of match for both first words, *After*, and, *Police*, since both are found lowercased later in the alternate sentence. In the second pair, however, truecasing only decreases the match for the word, *He*. The first word in the JHU output, *In*, is not found in the reference in either uppercase or lowercase, so its case status is irrelevant.

Table 68: Example of Word Order in Reducing Case Matches in Truecased Output

Truecased Russian-to-English Translation from WMT 2016 "JHU" System	
REF:	**After** fleeing the campus, **police** later picked up Lamb's trail when he crossed back into Mississippi from Arkansas.
JHU:	**Police** attacked the trail lambda **after** his escape from the campus when he was heading back to Mississippi from Arkansas.
REF:	**He** was quite the heartthrob back then.
JHU:	In those years, **he** enjoyed success with women.

The AFRL-J system has the largest number of sentences where the first word is found elsewhere, suggesting that this system has the most to lose when truecasing is applied. This does not fully explain the scoring deficit however. The AFRL-K system and the JHU system have similar expected losses for truecasing, yet the JHU system shows less decrease in BLEU score than the AFRL-K system in Table 65.

Truecasing and Punctuation

Generally, all the systems uppercase the first word when truecasing. However, the word-initial position can be obscured when the sentence begins with punctuation (usually a quotation mark). The WMT 2016 systems were evaluated for their handling of such occurrences (Table 69). The AFRL-J system failed to uppercase in this situation.

Table 69: Comparison of *WMT 2016* Systems Case Matching Performance on Sentences with Leading Punctuation

Comparison of Truecased Output from WMT 2016 Systems (Sentences with Leading Punctuation Mark)					
	REF	**JHU**	**EDIN**	**AFRL-K**	**AFRL-J**
"Alpha	143	170	152	170	40
"alpha			2	7	**124**

The 40 sentences in which AFRL-J shows uppercase after the quotation mark include named entities and the personal pronoun, *I*, which are capitalized regardless of sentence position. The AFRL-K system uses uppercase appropriately after leading punctuation; the 7 exceptions are sentences in which the first word was subsequently dropped as an unknown.

A program was written to examine the potential improvement for the AFRL-J system if truecasing were modified to use uppercase after a leading quotation mark (Table 70). Uppercasing will only improve the output if the first word matches the reference. There were 113 lines with leading quotation marks in both the reference and the AFRL-J system output. Of these, 65 have the same word, and could be improved, while 48 have a different word.

Table 70: Potential for Improving Case Matching Performance on Sentences with Leading Punctuation

Truecased Output from WMT 2016 "AFRL-J" System (Sentence with Leading Punctuation Mark)		
	REF	**AFRL-J**
Can Improve:	"It would be a surprise if the Fed hiked rates ...	"it will be a surprise if the Fed will raise rates ...
Can't Improve:	"Obviously the labor market ...	"it is clear that the labor market ...

Truecasing can also be affected by the presence of colon punctuation introducing a clause (Table 71). The WMT 2015 JHU and EDIN systems always capitalize a word following a colon. This suggests that these systems are using a rule-based, context-sensitive casing decision. In contrast, the AFRL-H system sometimes capitalizes in this situation and sometimes does not, based on the instances of uppercase and lowercase versions of the word in the training data. The English reference also shows variation in capitalization after a colon, giving the AFRL-H system an advantage in this situation.

Table 71: Comparison of *WMT 2015* Systems Case Matching Performance on Clauses with Capitalization Following a Colon

Comparison of Truecased Output from WMT 2015 Systems (Clauses with Capitalization after a Colon)				
	REF	**JHU**	**EDIN**	**AFRL-H**
alpha : Beta	35	99	101	19
alpha : beta	62	0	0	81

A program was written to investigate the correspondence between the AFRL-H output and the reference file. This shows that the AFRL-H system matches the reference program in 62 out of 97 instances, giving a better result than the always-uppercase strategy used by the other systems (Table 72).

Table 72: Case Matching Performance of *WMT 2015 AFRL-H* System on Clauses with Capitalization Following a Colon

Output from WMT 2015 "AFRL-H" System (Clauses with Capitalization after a Colon)			
REF	**AFRL-H**	**Matching**	**Not Matching**
alpha : Beta	alpha : Beta	15	
alpha : Beta	alpha : beta		18
alpha : Beta	no colon		2
alpha : beta	alpha : beta	47	
alpha : beta	alpha : Beta		2
alpha : beta	no colon		13
Totals		62	35

The same effect was observed in the 2016 WMT output (Table 73).

Table 73: Comparison of *WMT 2016* Systems Case Matching Performance on Clauses with Capitalization Following a Colon

Comparison of Truecased Output from WMT 2016 Systems (Clauses with Capitalization after a Colon)					
	REF	**EDIN**	**JHU**	**AFRL-K**	**AFRL-J**
alpha : Beta	48	101	100	34	16
alpha : beta	95	0	0	66	86

When both the reference and the output have colon punctuation, the AFRL-K system matches the reference in 77 instances, and fails to match in 18 instances. The AFRL-J system matches the reference in 68 instances, and fails to match in 27 instances.

Quality Control of Synthetic Data

An earlier section dealing with pivot methods, "Synthetic Arabic Data" under 2.1.3.17 "Experiments in Pivot Methods for MT", describes the use of Systran to translate English into Arabic for use as additional, synthetic data to train the Arabic-to-English MT system. Problems in this backtranslation step diminish the usefulness of the synthetic data. The Systran output was examined, using length disparity as an indication of translation problems. Problems were noted with tokenization and special characters in the English file, which then led to problems in the synthetic Arabic file. Other problems derive from Systran's treatment of Arabic morphology.

An English contraction in apostrophe-s is translated into Arabic words, but some of the English data contains backslashes before the apostrophe, causing the backslash and letter s to persist into the Arabic translation:

> Ready or Not, It's Time to Go Mobile.

يتأهّب أو لا، هو وقت أن يذهب موبايل.

Driver\'s Power Reclining Seat

قوة يرقد سيات 's \سائق

All the King\\\'s Men

all the ملك\ \ \ 's رجل

Other English sentences contained sequences of the unknown character symbol,  FFFD; short sequences were carried over into the Arabic translation, while some extremely long sequences generated an error instead of a translation.

en: 30

Table 74: Comparison of Arabic-to-English MT Output; Morphological vs. non-Morphological System

Occurrences of Comma Characters in Arabic-to-English MT Output		
	Latin Commas	Arabic Commas
plain	587	19
morph	11	577

The plain system output has more OOV words, including, surprisingly, 471 instances of the first person pronoun, *i*. The plain system output also contains escape characters for punctuation. These problems are illustrated in the following line in its plain and morph versions:

Plain: i fiddled مع شخص من الدرجات' s , قليلا فقط, فقطdemoted بعض هذه صورة' s .

Morph: كنت منافسي المليءه بالدرجات ، قليلا فقط ، تم عرض بعض من هذه الاله.

The two systems also differ in the use of diacritics. In theory, standard Arabic writing omits short vowels, although these may be specified with diacritics in religious text or in text written for language learners. There are some diacritics present in the synthetic Arabic training data, and these also appear in the plain system output. These short vowel diacritics are not found in the morphologically processed system output. Similarly, the diacritic for the morphological element, fathatan 064b, which indicates accusative case, is found in the plain system output but not in the morph system output. The Arabic reference file does contain diacritics, so the score of the plain system may benefit from the presence of diacritics.

Quality Analysis of Human Post-Editing

Human post-editing was applied for the WMT Russian-to-English translation. Monolingual English speakers edited the English MT output for fluency. A researcher with some Russian knowledge reviewed the quality of the human post-editing. The Russian letter-map transliterator was also considered as a tool to help the monolingual post-editors identify some of the missing words.

2.1.3.21 Improvements to the Hjerson Error Analysis Program

Error analysis of MT output was conducted using the existing programs SCLITE, Hjerson, and MT-ComparEval, and initial evaluations were made of the error analysis programs, Qualitative (Avramidis, 2016) [28] and Addicter.

The MT-ComparEval tool (Klejch, et al. 2015) [29] annotates the words found in the reference for two different MT outputs, and provides summary information about the correct words and phrases that are found in one output document but not the other. SCLITE[6] counts instances of insertions, deletions, and substitutions between the reference and the hypothesis. Hjerson (Popović 2011) [30] improves on this word error rate (WER) analysis by applying position-independent error rate (PER) to identify re-ordered words, and by using stemming to identify inflectional changes.

[6] <https://www.nist.gov/itl/iad/mig/tools>

Addicter (Berka et al. 2012) [31] allows the user to specify different programs for the initial word alignment, which might improve the quality of the error analysis. Addicter also uses training data for additional error analysis, and includes Hjerson analysis as part of its output.

A series of improvements were made to the Hjerson program. The revised program was applied to both English output and Russian output.

Improvements to Hjerson

SCLITE vs. Hjerson

SCLITE, designed for evaluating ASR, works best on output that is close to the reference standard. SCLITE derives a word alignment between two sentences and identifies insertions, deletions, and substitutions, and calculates an overall WER. This is not always informative for MT output, in which reasonable translations may have words in a different order from the reference; these show up in SCLITE as insertions and deletions.

Hjerson uses a combination of WER and PER to identify re-ordered words. When given a stemmed version of the reference and hypothesis files, Hjerson can also identify inflectional variants, which are considered to be errors in SCLITE and in the calculation of BLEU scores. Hjerson provides error counts as well as color-coded output: green=reordered, blue=insertion/deletion, red=lexical error, pink=inflection.

A testfile was created with example sentences from tst2013 that exhibit inflection and re-ordering. This testfile was evaluated in both Hjerson and SCLITE in order to examine the sensitivity of the BLEU score to inflectional and reordering errors. The examples here show the Hjerson color-coded output (Table 75 and Table 77), the SCLITE alignment (Table 76 and Table 78), and the BLEU score.

Table 75: Word Reordering Example; *Hjerson* Color-Coded Output

Hjerson Program Color-Coded Output (Word Reordering)	
REF:	a morning that i will never forget .
HYP:	i will never forget that morning .

Table 76: Word Reordering Example; *SCLITE* Output

SCLITE Program Output ***(Word Reordering)***										
REF:	A	MORNING	THAT	i	will	never	forget	****	*******	.
HYP:	*	*******	****	i	will	never	forget	THAT	MORNING	.
Eval:	D	D	D					I	I	
caps = changed, D=deletion, I=insertion, S=substitution										

BLEU = 40.99

Table 77: Inflectional Change Example; *Hjerson* Color-Coded Output

Hjerson Color-Coded Output ***(Inflectional Change)***	
REF:	there is a problem with community meetings .
HYP:	there was a problem with the community meeting .

Table 78: Inflectional Change Example; *SCLITE* Output

SCLITE Program Output ***(Inflectional Change)***									
REF:	there	IS	a	problem	with	***	community	MEETINGS	.
HYP:	there	WAS	a	problem	with	THE	community	MEETING	.
Eval:		S				I		S	
caps = changed, D=deletion, I=insertion, S=substitution									

BLEU = 21.11

Modified Error Reporting with Hjerson

In its numerical output, Hjerson reports error types with respect to reference and hypothesis, and also reports a SUMerr value that combines insertions, deletions, substitutions, inflections, and reorderings. A variant of the program was created for SCREAM Lab use, giving a "Word Choice" error measure that reports the sum of deletions + insertions + substitutions. Within Hjerson, deletions are measured as new words in the hypothesis and insertions are measured as unmatched words in the reference, so these can be summed without overlap. However, Hjerson records substitutions as lexical errors for both the reference and the hypothesis. Therefore, the revised program (*Revised Hjerson)* must include just the hypothesis measure for lexical errors in

the Word Choice total. An example demonstrating differences in output from Hjerson and the SCREAM Lab variant is shown below in Table 79, Table 80, and Table 81.

Table 79: Modified Error Reporting Example; *Hjerson* Color-Coded Output

Hjerson Program Color-Coded Output	
REF:	a morning that i will never forget .
HYP:	i will never forget that morning .

Table 80: Modified Error Reporting Example; *Hjerson* Numerical Output

Hjerson Program Numerical Output			
Type	**Count**	**Rate**	**Comments**
Wer:	5	62.50	
Rper:	1	12.50	PER for the reference
Hper:	0	0.00	PER for the hypothesis
SUMerr:	3	41.07	SUM = MISer + EXTer + hLEXer + hINFer + hRer
rINFer:	0	0.00	inflectional error in the reference
hINFer:	0	0.00	inflectional error in the hypothesis
rRer:	2	25.00	reordering error in the reference
hRer:	2	28.57	reordering error in the hypothesis
MISer:	1	12.50	missing (deletion)
EXTer:	0	0.00	extra (insertion)
rLEXer:	0	0.00	lexical error (any remaining errors) in the reference
hLEXer:	0	0.00	lexical error (any remaining errors)

Table 81: Modified Error Reporting Example; *Revised Hjerson* Numerical Output

"Revised Hjerson" Program Numerical Output (SCREAM Lab Variant)			
Type	**Count**	**Rate**	**Comments**
Inflection:	0	0.00	hINFer
Reordering:	2	28.57	hRer
Word Choice:	1	7.00	MISer + EXTer + hLEXer
WER:	5	0.62	
Rper:	1	0.12	
Hper:	0	0.00	

Hjerson Problems with Reordering Classification

There are two types of error possible in the Hjerson classification when the sentence involves reordering and multiple instances of the same word. The first error occurs when the WER calculation assigns an error to one word, but the PER calculation assigns that error to a different word. For example, in the following diagram, the symbols, *a*, *b*, *c*, and *d*, represent words, and the reference has two instances of word, *c*, while the hypothesis has two instances of the word, *d*. Hjerson's first step is to assign a word alignment (Figure 13):

```
REF   a   b   c   c   d
      |   |   |   |   |
HYP   a   b   d   c   d
```

Figure 13: *Hjerson* Classification Error Example #1; Word Alignment

Since the words, c, and, d, don't match for word, 3, this counts as a substitution error in the WER (Figure 14).

```
WER               SUB
REF   a   b   c   c   d
      |   |   |   |   |
HYP   a   b   d   c   d
WER               SUB
```

Figure 14: *Hjerson* Classification Error Example #1; WER Determination

Next, Hjerson calculates the PER (Figure 15). Working from left to right in the reference sentence, Hjerson looks for each reference word anywhere in the hypothesis. Word, *a*, is found, word, *b*, is found, and then word, *c*, is found, although in a different location. Then we come to another, *c*. Since each hypothesis word can only be used once, this second *c* has no match, and is classified as a position-independent error. Finally, word, *d*, is found. A similar matching takes

effect for the hypothesis, where the first *d* in the hypothesis is matched to the final *d* in the reference, leaving no match for the final *d* in the hypothesis.

PER				ERR	
WER			SUB		
REF	a	b	c	c	d
	|	|	|	|	|
HYP	a	b	d	c	d
WER			SUB		
PER					ERR

Figure 15: *Hjerson* Classification Error Example #1; PER Determination

Logically, the PER error should be assigned to the word with the WER error, like this (Figure 16):

PER			ERR		
WER			SUB		
REF	a	b	c	c	d
	|	|	|	|	|
HYP	a	b	d	c	d
WER			SUB		
PER			ERR		

Figure 16: *Hjerson* Classification Error Example #1; Expected PER Determination

Hjerson next uses the combination of WER and PER to classify errors. A combination of WER=SUB and PER=ERR is classified as a lexical error (Table 82 and Figure 17):

Table 82: *Hjerson* Classification Error Example #1; Error Mapping

Hjerson Error Classification Mapping	
CLASS	**LEX**
PER	ERR
WER	SUB

REF	a	b	c	c	d
	|	|	|	|	|
HYP	a	b	d	c	d
WER			SUB		
PER			ERR		
CLASS			**LEX**		

Figure 17: *Hjerson* Classification Error Example #1; Classifications

But in the situation described here, we have a word where WER=no error and PER=ERR. Hjerson has no classification for that error, which therefore goes uncounted. A revision was

made to Hjerson to re-assign lone PER errors to the word that has the WER so Hjerson can classify them correctly.

A second type of error occurs when Hjerson detects re-ordering or inflectional errors. A re-ordering is detected when one sentence has a word error but not a position-independent error (i.e., the word is present, but in the wrong position). At the initial WER level, Hjerson classifies both instances of the word as errors, either insertions, deletions, or substitutions, depending on the alignment. The PER calculation detects the presence of reordering, so no error is assigned to word, *c*, at this level. These classification outcomes are shown in Figure 18.

PER				[OK]	
WER				DEL	
REF		a	b	c	d
		|	|		|
HYP	c	a	b		d
WER	INS				
PER	[OK]				

Figure 18: *Hjerson* Classification Error Example #2; WER and PER Determination

After reordering detection, the words in question are classified as reordering errors, which overrides the insertion or deletion specification from the WER step (Figure 19).

CLASS				**REORD**	
PER				[OK]	
WER				DEL	
REF		a	b	c	d
		|	|		|
HYP	c	a	b		d
WER	INS				
PER	[OK]				
CLASS	**REORD**				

Figure 19: *Hjerson* Classification Error Example #2; Classifications

This is fine when the original classification was insertion or deletion, as in Figure 19 above. But, if the matching word was originally considered a substitution, that means it was aligned to another word that was also called a substitution. In this example, word, *c*, is aligned to word, *e*, and both are initially classified as substitution errors (Figure 20).

WER				SUB	
REF		a	b	c	d
		|	|	|	|
HYP	c	a	b	e	d
WER	INS			SUB	

Figure 20: *Hjerson* Classification Error Example #3; WER Determination

Position-independent error analysis accepts word, *c*, and error classification marks it as a reordered word (Figure 21). The originally aligned word, *e*, retains the substitution classification after re-ordering detection, even though it is no longer considered the counterpart of word, *c*.

CLASS				**REORD**	
PER				[OK]	
WER				SUB	
REF		a	b	c	d
		|	|	|	|
HYP	c	a	b	e	d
WER	INS			SUB	
PER	[OK]			ERR	
CLASS	**REORD**				

Figure 21: *Hjerson* Classification Error Example #3; PER Determination and Classifications

Instead, the leftover word needs to be re-classified as either an insertion or deletion (Figure 22):

CLASS				REORD	
PER				[OK]	
WER				SUB	
REF		a	b	c	d
		|	|	|	|
HYP	c	a	b	e	d
WER	INS			SUB	
PER	[OK]			ERR	
CLASS	REORD			**INS**	

Figure 22: *Hjerson* Classification Error Example #3; Expected Classifications

A similar problem can occur when a word that is originally considered part of a substitution pair is detected to be an inflected form of a word elsewhere in the sentence. The revised Hjerson program re-classifies the leftover words in both situations.

Improving Hjerson by Reporting Lists of Reordered Words

The Hjerson program was modified (*Revised Hjerson*) to report a list of the reordered words, and also to report the distance between re-ordered words, based on the index of the word in the reference and the hypothesis. Further analysis showed that it is not always meaningful to calculate the distance by the indices, and an algorithm was added to report the reordering distance based on distance from matching words which serve as anchors.

For example, in the sentence fragment in Figure 23 below, a longer reference sentence has led to the index, 27, for the word, *in*, versus an index of 24 for the same word in the hypothesis. Revised Hjerson correctly identifies the subsequent place name as a reordered word, relative to the aligned words, but the indices for the place name happen to be the same (28) in the reference and the hypothesis. So a distance calculation based on word indices gives this place name a reordering distance of zero.

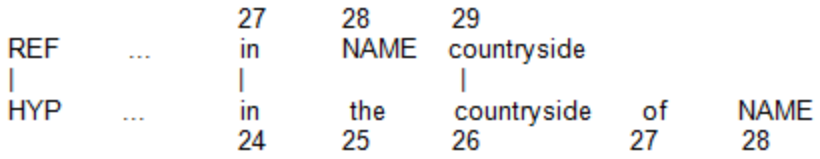


Figure 23: *Revised Hjerson* Word Reordering Example; Word Alignment

A more meaningful measure is the distance from the expected placement of the word with respect to the aligned words. This was calculated using an interpolated word index, making a slot for each word and leaving a blank slot in the other sentence if it lacks that word in that position. This method maintains matching indices for the words that were matched in the initial alignment. Now the distance between the original and reordered place name is well defined as (48-45)=3, as shown in Figure 24 below.

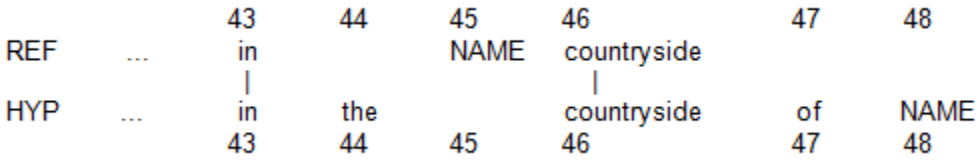


Figure 24: *Revised Hjerson* Word Reordering Example; Modified Word Alignment

The revised distance calculation was used to generate a list of the most frequently reordered words in an Arabic to English MT system. These tended to be closed class words such as pronouns, prepositions, determiners, and conjunctions. This led to the observation that long reordering distances can be misleading, particularly for commas and closed class words, which may be repeated in different parts of a sentence. For example, the hypothesis in the example above has the preposition, *of*, near the end of the sentence, while the reference expresses the phrase without using a preposition. This extra *of* was unfortunately paired by the revised Hjerson program with a semantically unrelated *of* in the reference, near the beginning of the sentence.

The program was revised to only report reordering distances of 10 words or less. In a data set of 1500 sentences, *Revised Hjerson* counted 498 instances of reordering with the word, *the*. When capped at a reordering distance of < 10 words, the count becomes 159 instances of reordering with the word, *the*. These are more likely to be actual instances of reordering.

Examining the Effect of Inserted Words

An examination of the frequency of the words, *the*, and, *of*, in the reference and the hypothesis of the Arabic to English MT system showed that the MT output tends to generate more instances of these words than were present in the reference, contributing to the possible mis-assignment of distant words as reordering pairs (Table 83).

Table 83: Examination of Overzealous Word Insertion by an Arabic-to-English MT

Instances of Closed-Class Words in a 1500 Line Arabic-to-English MT Test File			
Words Per Line	**Total**	***the***	***of***
Reference	24.1	1.38	0.77
MT Output	26.62	2.05	1.21

A program was written to take Hjerson's error listing and create a new file omitting any words marked as inserted. Scoring this file against the reference gives an indication of the way over-generation degrades the translation, showing a potential 2.5 point difference in BLEU score.

The analysis in this case is complicated by the fact that this file contains social media hashtags of the form, *#alpha*, and, *#alpha_beta*. Tokenization unfortunately adds spaces to the hashtags, creating isolated # and _ tokens, which are then counted as inserted words. These elements were removed separately for comparison. The difference attributable to regular inserted words is still nearly 2 BLEU points (Table 84).

Table 84: Effect of MT Inserted Words on BLEU Score

MT BLEU Scores	
	BLEU
MT Output	13.46
no # or _ tokens	14.06
no inserted words	15.96

Tokenization and Lowercasing

The error analysis generated by Hjerson will differ depending on the tokenization and lowercasing of the input files. This can be particularly important with the specialized syntax of URLs and hashtags. In general, Hjerson's analysis will be more informative after tokenization and lowercasing, but URLs and hashtags should be protected from those processes. For example, in a URL like www.WebAddress.com, it is not meaningful to consider www and com as independent words. Therefore, tokenization variants were created to protect URLs and hashtags when conducting error analysis in Hjerson.

Improving Hjerson by Reporting Lists of Inflected Words

The Hjerson program was modified to report a list of the inflected word pairs, and code was added to classify the inflected English words by suffix type, although the amount of information available in English is limited. The endings, *-ed*, and, *-ing*, indicate verbs, while the suffix, *-s*, could be either a noun plural or a third singular verb. The alternation, *it/its*, was recorded as an instance of a possessive. The ability to list the inflected word pairs was also used in an analysis of Russian inflectional errors, which is addressed in a subsequent section titled “Applying “Revised Hjerson” to Russian”.

Choosing a Stemmer for the Detection of Inflected Words in Hjerson

Hjerson uses stemmed input to calculate inflectional errors. Both the Porter Stemmer[7] and the TreeTagger stemmer[8] were considered for use with Hjerson to assist in the detection of inflected word pairs. In general TreeTagger produces a better result, but it has some limitations.

The Porter stemmer will remove an isolated "*s*" from a phrase like, "*the s curve*". The English possessive suffix in apostrophe-s is also vulnerable after tokenization separates the *s*. The removal an isolated "*s*" causes Hjerson to fail because the word count now differs for the stemmed and unstemmed sentence.

The Porter stemmer also removes a final *–s* from words like *is*, *as*, *has*, and removes a final *–e* from words like *one* and *use*. This causes Hjerson to assert spurious inflectional alternations like *i*/*is*, *a*/*as*, *on*/*one*, *ha*/*has*. TreeTagger stems less aggressively, and is also able to identify suppletive forms which do not share character sequences, returning the lemma (typical form) for words like *is*>*be*, *was*>*be*, *fell*>*fall*.

TreeTagger is statistically trained, which means context affects the decision to remove certain suffixes. TreeTagger also tends to preserve derivational endings (those that change the part of speech). These characteristics make TreeTagger a more careful stemmer, but the MT output is not typical well-formed English, and for error analysis we may want to relate words that differ in part of speech, or words that occur in an unusual context. For example, we need to compare phrases like these:

REF: to force foreign students to leave

HYP: to force the students of the flag of foreigners to leave

The Porter stemmer creates **foreign|ers**, allowing Hjerson to match *foreign* and *foreigners*. The TreeTagger program only stems the plural ending, creating **foreigner|s**. This preserves the part of speech of the noun, but obscures the relationship between these words in the stemmed files.

Improving Hjerson by Reporting Differences in Uppercased and Lowercased Words

Machine translation is often conducted on lowercased text, and a final step of truecasing restores capitalization in typical situations, such as the first word in a sentence. However, truecasing may actually lower BLEU scores. For example, if the MT output has reordered the first words of the sentence, capitalizing the first word will cause it to no longer match the reference. The Hjerson program was extended to distinguish errors which are due only to capitalization. A change also had to be made in calling the TreeTagger stemmer, to prevent it from changing capitalization.

In a file of 1500 lines, the program identified about 600 words in which capitalization created errors. This initial analysis was restricted to aligned words, and does not consider position-independent word pairs.

Improving Hjerson by Reporting Word Alignment

The Hjerson program was extended to output A3 files, for use in the Qahira alignment viewer. There are two points in the program at which alignment data can be reported, the initial

[7] http://snowball.tartarus.org/algorithms/english/stemmer.html

[8] http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger

alignment and the final alignment. The initial alignment is used to calculate WER, and this A3 output is useful for debugging and tracing how Hjerson combines WER and PER. The final alignment includes the good links (matches, reorderings, and inflected word pairs). A modified version of Qahira was created to enable a Hjerson mode for this final alignment, with each link type printed in the color used by Hjerson in the sentence output.

Smoothing and Tokenization in the Calculation of BLEU Scores

Hjerson does not generate a BLEU score, so existing SCREAM Lab programs were used to calculate BLEU scores for the sentences being evaluated in Hjerson. Scores were calculated by either of two methods. The program mteval-v13a.pl simulates competition scoring, but requires sgml-wrapped text. For plain text, scoring was conducted by first applying the Moses tokenizer, lowercasing, and then using the program multi-bleu.pl. Unfortunately, the mteval and multi-bleu processes produced slightly different BLEU scores, because these methods differ in tokenization and smoothing.

The mteval program conducts its own internal tokenization, which differs from the Moses tokenizer in its treatment of apostrophes, hyphens, and HTML fragments. A new tokenizer program was therefore written to match the mteval tokenization when providing data for multi-bleu.pl.

The mteval program, but not the multi-bleu program, implements BLEU score smoothing to prevent zero values in short sentences. The BLEU score is calculated as the geometric mean of the matching unigrams, bigrams, trigrams, and 4-grams, adjusted by a brevity penalty. This can generate a zero score if one of the components fails to have any matching ngrams. This is typically not a problem when scoring a document, because it is rare for there to be no 4-gram matches within a whole document; but it can easily occur when using the program to score individual sentences. The mteval-v13a.pl program prevents this via smoothing; assigning a small value to any component that would otherwise have a zero entry. A revised version of the multi-bleu program was written to implement the same smoothing algorithm.

Applying "Revised Hjerson" to Russian

The revised Hjerson program was used to examine the number of inflectional errors in the Russian output of English-to-Russian MT. The existing Mystem[9] morphological analyzer was used to derive lemmas for the inflectional analysis. For example, in the sentence below inTable 85, the revised Hjerson identifies both reordering in green (of *prime-ministers* and the word, *and*), and inflection errors in pink (for *India*, *Japan*, and the verb, *meet*). The translation and morphological annotations have been added by hand for the sake of illustration. While the reference uses a possessive, genitive case construction analogous to "The prime ministers of India and Japan", the hypothesis instead uses nominative case, creating "India and Japan prime ministers".

[9] <https://api.yandex.ru/mystem/>

Table 85: *Revised Hjerson* Analysis of Reordering and Inflectional Errors in English-to-Russian MT

"Revised Hjerson" Output of an English-to-Russian MT (Reordering and Inflectional Errors)							
REF:	премьер-министры	индии	и	японии	встречаются	в	токио
	prime-ministers	India	and	Japan	are-meeting	in	Tokyo
		GEN		GEN	present		
HYP:	индия	и	япония	премьер-министры	встретятся	в	токио
	India	and	Japan	prime-ministers	will-meet	in	Tokyo
	NOM		NOM		future		

Overall, the revised Hjerson program reported a rate of inflectional errors of about 9% on the file, newstest2015 (that is, about 9% of the words in the hypothesis file were incorrectly inflected). This provides a baseline for judging the effectiveness of the inflection generation techniques under development for the WMT competition.

For the newstest2015 file with 2818 lines, the revised Hjerson recorded 3947 reordered words overall, but only 2306 reordered words when the reordering distance was capped at 10 words. As expected, punctuation and closed class words that are often repeated constituted a large number of the words that were reduced when capping reordering distances (Table 86).

Table 86: Effects of Reordering Distance on Word Reordering from *Revised Hjerson*

Word Reorder Instances ("Revised Hjerson" Program, newstest2015 file)		
Word	**Original Distance**	**Cap of 10 Words**
,	465	175
в "in"	322	161
на "on"	114	59
и "and"	125	70

The inflected word lists were examined for insights into the problem of inflection generation for Russian translations. The revised Hjerson program outputs lists of inflected word trios, showing the reference, hypothesis, and stem which is common to both. There were 5572 instances of Russian words where the stems matched, but the inflected hypothesis word did not match the inflected reference word. For the example in Table 87, a manual examination of item a) below shows an error in tense, while item b) shows an error in number.

Table 87: Inflection Analysis of Russian Translations from *Revised Hjerson*

Inflection Analysis of Russian from "Revised Hjerson" Program (Inflected Word Trios)		
a)	был , будет , быть	he was, he will be, to be
b)	будут , будет , быть	they will be, he will be, to be

The inflection lists can be submitted to Mystem to automatically classify the specific error type. For example, Table 88 shows the Mystem entries for the reference and hypothesis in error b), showing the difference of plural (pl) and singular (sg).

Table 88: Inflection List as Input to *Mystem* for Error Type Classification

Mystem Entries for Item "b)"		
REF:	будут	{быть=V, intr=inpraes,pl,indic,3p}
HYP:	будет	{быть=V,intr=inpraes,sg,indic,3p}

These Mystem entries were also used to identify the distribution of inflectional errors across parts of speech. (These classifications sum to more than 5572 total words because Mystem sometimes reports an ambiguous POS.)

Table 89: Analyzing the Distribution of Inflection Errors by Parts-of-Speech

Inflection Errors	
1076	Verbs
2852	Nouns
287	Pronouns
1118	Adjectives
350	Adjectival Pronouns
92	Numerals

2.1.3.22 Manual Evaluation of MT Output for Competitions

Manual evaluation of WMT output sentences was required of all workshop research groups. This involved rank ordering of four or five different MT outputs, based on the reference sentence.

2.2 Laboratory Corpora Support

Activities were conducted in the acquisition, grooming, and annotation of corpora. Much of the work concentrated on Russian, Ukrainian, Chinese, and Hindi corpora. Some attention was also given to Vietnamese, Farsi, and Arabic corpora, albeit to a lesser extent. A small amount of French-to-English translation work was also conducted, and is described in subsection 2.2.1. Subsection 2.2.2 focuses on efforts locating and harvesting parallel corpora, mainly from internet

sources. Grooming efforts are discussed in subsection 2.2.3, whereby a variety of datasets were scrutinized for the purposes of identifying and correcting corpora errors. Among the errors addressed are word and sentence alignment problems, word and sentence segmentation problems, untranslated text, wrong language text, sentence-internal repetitions, duplicate lines, mixed alphabet spellings, punctuation and spelling normalization, and miscellaneous challenges presented by mismatched characters and marked-up/stylized text. Finally, subsection 2.2.4 goes into grammatical annotation work performed on Russian, Ukrainian, Chinese, and Hindi corpora.

2.2.1 Translation Work

Work was started on a project to extend a Pashto/French news database. The existing database has Pashto news audio, Pashto transcription of the news audio, Pashto text derived from the transcription, and French text, previously translated from the Pashto text. The SCREAM lab project will extend this database by providing English translations from the French text. A sample text was prepared for translation; the translator's work was reviewed and feedback was provided on some stylistic and technical issues.

2.2.2 Discovery and Harvest of Parallel Corpora

In general, parallel text resources can be found online by searching for menu items or hyperlinks using the local spelling of the name of the language. For example, for Ukrainian/Russian parallel text websites, look for the words, Українською, and, По-русски, or the abbreviations, УКР, and, РУС.

2.2.2.1 Online News: Russian/Ukrainian, Russian/Ukrainian/English

A survey was conducted of online resources for Ukrainian and Russian corpora, dictionaries, and news articles, as well as sources for Ukrainian/English and for trilingual Ukrainian/Russian/English text. These resources are detailed in a document included as an appendix, “APPENDIX A: Ukrainian Parallel Text Resources”. Since that document was compiled, an additional website was identified; korrespondent.net (Russian) and ua.korrespondent.net (Ukrainian).

A secondary search was conducted for parallel text resources in the science domain, using the keyword, nauka "science", to identify websites. Possibilities include www.slovoidilo.ua "Word and Deed”, which has a science news section, http://science.ua "Science News" (with various science sections, e.g., archeology, astronomy, biology, robotics, etc.), and the science and technology sections of the previously identified websites, http://zn.ua "Mirror Weekly" and http://gazeta.ua "News".

2.2.2.2 Slovnyk Dictionaries: Russian and Ukrainian

The Slovnyk website, www.slovnyk.org, was identified as a resource for bilingual dictionaries in over 30 languages, including Russian and Ukrainian. (The word slovnyk means *dictionary* in Ukrainian.) Languages include: Belarusian, Bulgarian, Croatian, Czech, Danish, Dutch, English (UK), English (USA), Esperanto, Estonian, Finnish, French, German, Greek, Hungarian, Icelandic, Italian, Latin, Latvian, Lithuanian, Macedonian, Norwegian, Polish, Portuguese, Romanian, Russian, Serbian, Slovak, Slovenian, Spanish, Swedish, and Ukrainian.

2.2.2.3 YeeYan Website: Chinese/English

The website, http://yeeyan.org, was identified as possible source of parallel text for Chinese/English. The Yeeyan website provides crowd-sourced translations into Chinese on various topics. A proofreading view displays English and Chinese text side-by-side in parallel paragraphs, which might be useful for guiding sentence alignment. However, an announcement on the website notes that the topics are now limited due to Chinese government censorship, with current events no longer being translated.

2.2.2.4 Experimenting with Common Crawl

To obtain parallel data for low resource languages, we executed the ideas put forth in "Dirt Cheap Web-Scale Parallel Text from the Common Crawl" [32]. The Common Crawl is an enormous dataset of crawled websites that is available through Amazon Web Services (AWS). The basic idea is to find multiple web pages with Uniform Resource Locators (URL) that differ only by ISO language codes. The dataset is free to access, but it must be accessed from within Amazon's AWS infrastructure.

While the available source code was mostly compatible with older versions of the Common Crawl dataset, it could not read more recent data and suffered from considerable performance problems. Removing unnecessary abstractions from the Java code, using standard libraries for dealing with HTTP parsing, and bundling together website metadata reduced processing time by 50%. In addition to lowering the cost for each run, it reduced the number of deployments due to failures.

When cleaned of false positives and other invalid data, the resulting parallel dataset for English and Somali is around 1,750 sentences, which seems to conflict with the paper's results. While these results were disappointing, the final cost-per-run of approximately $150 is low enough that this approach is not out of the question in the future.

2.2.3 Grooming Parallel Text Corpora

Several datasets were analyzed to discover and fix errors and inconsistencies in the corpora. TED Talks, HindEnCorp and HindMonoCorp, and several IWSLT and WMT datasets were among those addressed.

2.2.3.1 TED Talks Translationese

The nature of the TED Talk data causes some problems for word and sentence alignments. Because the original talks are live presentations given in English with visual aids, the following problems may be present:

1. English idioms and cultural references
2. false starts and other disfluencies
3. stage action such as gestures and interaction with the audience
4. material presented in English on the slides that is not included in the talk transcript

The translators often try to remediate these issues, for example, by:

1. adding parenthetical explanations and expanding acronyms

2. omitting false starts or re-wording awkward sentences

3. describing what is happening on stage

4. adding the slide text, sometimes in square brackets

Some example sentence pairs in which the translator has provided extra information:

i go to jpl .

я приближаюсь к jpl [лаборатория реактивных двигателей наса] .

(literally: I approach at jpl [laboratory jet engines NASA] .

hope for the best .

надеюсь , не подведёт . [пишет: " 2 + 2 "; на экране - 4];

(literally: I-hope , not it-fail . [he-writes: " 2 + 2 "; on screen – 4];

Adjusting for these translator innovations might improve word alignments, but since the translations are crowd-sourced, there is substantial variation from talk to talk, making it difficult to automate the corrections. One talk was identified for which this explanatory material changes the wording enough to warrant exclusion from the training data.

There are 861 Russian lines that include bracketed information; this appears to be mostly text from the slides, expansion of acronyms, and other explanatory information. There are 582 English lines with bracketed information, mostly omitted words and the notation, [*unclear]*. Translation might be improved by automatically removing bracketed information from the Russian side.

2.2.3.2 IWSLT 2014 TED Talks

A check for duplicate lines in the IWSLT 2014 TED Talk data turned up a handful of talks that were left untranslated (i.e., repeated as English in the other language file). There were 5 untranslated talks in the Russian file, 5 in the Chinese file, and 7 in the French file. This error was reported to the IWSLT competition organizers, who issued corrected data.

The IWSLT TED Talk files also contain sentence-internal repetition errors that appear to be the result of a processing error on the website. For example, the following English/French sentence pair has a duplication of the yellow highlighted clause in the French translation, indicated by the blue highlight:

Last year I showed these two slides so that demonstrate that the arctic ice cap, which for most of the last three million years has been the size of the lower 48 states, has shrunk by 40 percent.

L'année dernière, je vous ai présenté ces deux diapositives qui montraient que la calotte glacière arctique, *qui pendant ces 3 derniers millions d'année avait la taille des Etats-Unis sans l'Alaska, qui pendant ces 3 derniers millions d'année avait la taille des Etats-Unis sans*

l'Alaska, avait diminué de 40%.

This kind of repetition error has to be distinguished from legitimate, rhetorical repetition, such as music lyrics. A program was written to detect repeated phrases above a specified length; when

parallel text is available, the program only removes repeated phrases if there is no corresponding repetition in the English source sentence. The phrase length for repetition detection was set to 11 or more words (or in the case of Chinese, 16 or more characters). A spelling normalization step is required for Farsi, using the Moses program, remove non-printing characters. This removes the zero-width non-joiner character, which gives the appearance of a word space in Arabic script languages, but does not establish separate words computationally.

Applying the repetition detection program identified a substantial amount of repetition in the Farsi test sets (more than 20% of the lines for tst2012 and tst2013). Lesser amounts of repetition were found in the Chinese dev and test data, and in the French dev data. Training data contained a fair amount of repetition in the Farsi (up to 7% of lines), with minor amounts in the Chinese, French, and Russian.

When translations containing such repetitions are extracted for training data, they cause sentence alignment problems; when they are used as test data, they degrade the MT. Removing the repetitions from the IWSLT Farsi tst2014 file improved the Farsi-to-English BLEU score by +1.53 points.

The TED Talks were re-aligned by the competition organizers for the 2014 training data. A manual review of selected talks suggests that the new alignments are sometimes better, and sometimes worse. There were some systematic changes: most of the in-line speaker annotations have been removed, and there were also changes in the meta-data tags.

The Russian/English TED Talk files were examined more closely. Sentence alignment was checked by looking for length disparities across parallel text. There were 57 Russian lines that were much longer than the English, and 65 English lines that were much longer than the Russian. For example, the length disparity in this pair derives from a mis-alignment, in which the Russian line contains an additional sentence:

> English: so , we did another experiment .
>
> Russian: тогда мы провели ещё один эксперимент . для этого эксперимента мы набрали большую группу студентов
>
> Russian literal translation: so we conducted another one experiment . for this experiment we took large group of-students

The pairs with longer English lines must be examined manually, since Russian sentences may be legitimately shorter than their English counterparts due to the possible omission of subject pronouns, articles, and the verb-to-be. The tokenizer may also contribute to this length disparity by separating English apostrophes. In this example, both verb omission and apostrophe separation contribute to make the Russian sentence shorter than the English sentence:

> English: now , here 's another one .
>
> Russian: ещё одно .
>
> Russian literal translation: another one .

There were some instances of mixed alphabet spellings, which occur when a Russian writer uses a Latin character in place of a visually similar Cyrillic character. There were 365 mixed alphabet words in the Russian file, of which 347 were mostly Cyrillic, 24 were mostly Latin, and 138

involved accented characters or characters from other scripts. A program was applied to automatically convert 328 of these mixed-alphabet words to either all-Cyrillic or all-Latin.

A few unusual characters were found in the English file, such as the accent mark, ´ 055B, instead of the more typical ´ 0301, the use of the dotless "i" ı 0131 in some borrowed English words like *hy-wıre*, and some occasional bad encoding of accented characters in borrowed words or names like *charlotte brontăt'* "charlotte brontë". There were also some usage differences with number phrases: The Russian file tends to use × 00D7 where English spells out the word, *times* (*8 × 13* vs. *8 times 13*). Also, the English file uses the degree sign, ˚ 02DA, while the Russian file contains both ˚ 02DA and ° 00B0.

2.2.3.3 IWSLT TED Talks: Chinese/English

The IWSLT organizers provided different Chinese/English TED Talk data for 2011 and 2013. The 2011 and 2013 data were compared to compile a dataset of just the new talks in 2013. The 2013 talks contain meta-data tags that identify the start and finish of each talk. A program was written to use the first line of each 2013 talk, and seek that line in the 2011 file. A talk was identified if the first lines match, or if the first lines constitute a partial match and the four subsequent lines match, not counting short lines such as "thank you" which may co-occur accidentally. The result of the comparison was a list of 377 new talks, with about 45,000 lines of new data.

The meta-data tags were removed before training on the 2013 datasets. This led to a sentence-alignment problem if the </transcript> tag was attached to preceding material on one side, but used in isolation on the other side. Attached transcript tags were noted in the Chinese/English, French/English, and Russian/English datasets.

The IWSLT 2014 Chinese/English training file exhibits a different sentence alignment problem, in which a Chinese sentence contains the same material as two distinct English sentences, and the Chinese sentence is repeated to align with each English sentence, as shown in Table 90.

Table 90: Sentence Alignment Problem in *IWSLT 2014* Chinese-English Training File

Sentence Alignment	
English	**Chinese**
abc	ABC DEF
def	ABC DEF

There were 64 lines with this kind of duplication, and 74 lines with partial duplication, out of a total of 186,972 total training file lines. Similar alignment problems had been previously found and corrected for the dev2010 and tst2010 files.

2.2.3.4 Assembly of Traditional Chinese Data from Harvested TED Talks

The IWSLT 2015 competition uses simplified Chinese translations of TED Talk files. Many of these talks have also been translated into traditional Chinese characters, typically by different translators. These talks provide a paraphrase of the simplified Chinese that could be used as language model data or as additional references for English-to-Chinese translation. For this purpose, we need to perform character conversion from traditional to simplified Chinese.

All but 18 of the 1718 talks used in the dev, test, and training data were available in traditional Chinese. These talks had been previously harvested in the SCREAM Lab, and stored with timestamps corresponding to the original audio-video files. An existing synchronization script provides the ability to collect parallel lines from a foreign translation and the English transcript. This was adapted to collect parallel traditional and simplified Chinese talks.

First, an error was corrected in the use of the SCREAM Lab harvest. Data from multiple languages is stored as time-stamped text segments that coordinate with the TED Talk video, and a SCREAM Lab extraction script is used to assemble parallel text for the desired language pair. The extraction program monitors the timestamps in both languages. If the difference exceeds a threshold, the program collects additional lines from one side to bring the timestamps back in line. For example, in the talk extract below in Table 91, the English line, “that becomes the rain that feeds the plants”, has been translated into two lines in Ukrainian. The timestamps match for the first line, but do not match for the second line. The extraction program notices that the timestamps are off when comparing the second English line with the second Ukrainian line, and goes on to collect the third Ukrainian line as well.

Table 91: Parallel Text Extraction Error from Ukrainian-English *TED Talk* Transcripts

Parallel Text Extraction Output (SCREAM Lab Script)			
	Timestamp	**Phrase**	**Literally**
English:	41861	that becomes the rain that feeds the plants	
	46134	that feeds the animals."	
Ukrainian:	41861	що стають дощем,	that becomes rain
	44038	який живить рослини	that plants grows
	46215	і тварин".	and animals".

An error was found and corrected in the extraction script, which was failing to increment the index on the English side after this catch-up step.

The IWSLT competition uses sentence-aligned files, but the harvested talks consist of sentence fragments. The English version of the IWSLT file was used as a template to assemble the fragments of the traditional Chinese file. The English and traditional Chinese fragments are collected in parallel by the synchronization program. Then each English fragment is compared to the next English sentence in the template. If a match is made, both the English and the traditional Chinese fragments are assigned to that sentence.

Some variation occurs, due to the fact that the TED Talk website sometimes updates its English transcriptions. Variations between the SCREAM lab harvested English files and the IWSLT English files include the annotation of laughter, applause, or speaker initials; the representation of punctuation with escape sequences like " and minor re-wording. Therefore, during the matching process, a fragment is counted as a match if the template sentence contains 90% of the fragment words. For remaining unmatched fragments, a second-chance match is done using context. For example, if the preceding fragment is assigned to sentence 3, and the following fragment is assigned to sentence 5, then the current fragment must be assigned to sentence 4. Finally, hand editing may be applied to assign any remaining fragments.

After assembly of all the fragments, the traditional characters are then converted to simplified characters. Various programs were used to normalize certain traditional Chinese characters to their simplified Chinese counterparts. For example, the traditional character, 龜 "turtle", becomes the simplified character, □ . The simplified and traditional characters are found in the same codepoint ranges, so it is difficult to diagnose them. In addition, the mapping is not 1-to-1: Multiple traditional characters may map to the same simplified character. In order to detect traditional characters, a preliminary program was created using a set of 1-to-1 mappings specified by the Internet Engineering Task Force (ietf.org). This was supplemented by analyzing the characters in the cedict Chinese-English dictionary, which lists each entry with both the traditional spelling and the simplified spelling. This normalization program, while lacking a full mapping, was still useful as a diagnostic to identify files containing traditional Chinese characters.

Other resources were identified for Chinese spelling conversion, including a CPAN module and an icu4j conversion program. The ietf mapping appears to be more complete than the CPAN module. Finally, a program, *trad2simp.py*, was written to convert the characters.

A final quality-control step checks for derived talks that are too similar to the simplified Chinese talks. This can occur if the same translator creates both the simplified and traditional versions, or if a different translator works off the other Chinese talk, instead of starting with the English talk. While there may be minor variations in punctuation or the treatment of foreign names, any talks having more than 80% of the lines identical after spelling conversion are considered too similar for use as paraphrases.

2.2.3.5 WMT HindEnCorp and HindMonoCorp Corpora

The WMT Hindi training and test data contained some overlap. A program was applied to identify overlapping lines; 552 lines of the 1500-line test file were found to overlap with the training data.

The HindEnCorp parallel Hindi/English corpus and the monolingual HindMonoCorp corpus required review. Programs were written to extract parallel text from these tab-formatted coropora, excluding mal-formed lines. In the HindEnCorp data, characters from Bengali and Dravidian were examined, and found to derive from quotations or names. However, Latin characters derived from lines of Spanish were included by mistake, and these lines were removed. A check for duplicate lines found 59,292 duplicate lines that needed to be removed from the 280,883-line HindEnCorp file.

Control characters, diacritics, and stress marks were identified and removed. Punctuation normalization was also needed. Sentence punctuation included the Hindi danda, । 0964, the Hindi double danda, ॥ 0965, and the Latin period. Abbreviations and initials may be punctuated with the Latin period, the Hindi abbreviation marker, ॰ 0970, or they may be left unpunctuated. Additionally, what would be a single character initial in English is sounded out in Hindi characters, so that TV, for example, is spelled □□□ □□□ /ti vi/. These variations make it difficult to specify a non-breaking prefix list to guide the Moses tokenizer. The HindEnCorp dataset also shows variation in the expression of numbers with Arabic digits (123), Indic digits (१ २ ३), Hindi numeral words, and the combination of Arabic digits with Hindi unit words.

There is a general spelling normalization program provided by the Unicode consortium, charlint.pl, that creates a canonical form for sequences that can be represented by different combinations of codepoints. For example, Hindi represents foreign sounds with dotted forms that can be written as either one or two characters (Table 92):

Table 92: Example of *Hindi* Dotted Forms

Example of Hindi Dotted Forms	
one-character	**two-character**
095E फ़	092B फ+ 093C □ = □□

The charlint program can be used to convert text to either fully composed (NFC) or fully decomposed (NFD) forms. While NFC is standard for many uses, NFD is more useful as preparation for spelling normalization, since the separate elements can be modified or removed.

For Hindi, charlint.pl was used to create the fully decomposed NFD form. This allows for normalization of diacritics, such as dotted forms and nasalization marks. A program was written to implement the Hindi normalization scheme suggested in Larkey et al. [33], and to also normalize digits to the Arabic forms.

2.2.3.6 IWSLT Vietnamese Spelling Normalization

A procedure was established for handling the IWSLT Vietnamese data. Since Vietnamese uses spaces to distinguish syllables, values must be adjusted for maximum sentence length and translation fertility. No problems were found with out-of-range characters or repetition errors, but normalization was required to deal with tone marks. The existing program, charlint, was used to convert data into the composed form (one Unicode codepoint per accented character), as opposed to a sequence of character plus accent mark, as illustrated below in Table 93.

Table 93: Comparison of Composed and Non-Composed Vietnamese Forms

Vietnamese Forms			
Composed:	đến	đ ế n	0111 1ebf 006e
Non-Composed:	đến	đ ê ́ n	0111 00ea 0301 006e

2.2.3.7 IWSLT Farsi Spelling Normalization

The IWSLT-provided Farsi normalizer was revised to add Arabic digit conversion, and to detach digits from any Arabic script characters. A review of the Farsi data shows that, both, Farsi digits (۰ ۱ ۲ ۳ ۴ ۵ ۷ ۸ ۹ , Unicode 06F0-06F9), and Arabic digits (٠ ١ ٢ ٣ ٤ ٥ ٦ ٧ ٨ ٩ , Unicode 0660-0669), are present, with 17,859 Farsi digits and 1,789 Arabic digits in the training file. The IWSLT-provided Farsi spelling normalizer converts Farsi digits to Latin digits (0-9); this was revised to include the Arabic digits.

Some instances of digits attached to words were also observed. Latin sequences like MP3 are generally intentional and should be preserved, but digits attached to Arabic characters, such as روز5 "5 days", should be split off. This affects about 400 words in the Farsi training file. An examination of the IWSLT Arabic files also shows sequences of digits attached to Arabic characters.

2.2.3.8 IWSLT 2016 QED Arabic/English Corpus

The QED Arabic-English corpus that was provided for the IWSLT 2016 competition exhibits problems in sentence alignment, sentence and word segmentation, and spelling.

It was apparent from the spelling errors that many of these crowd-sourced translations were created by non-native speakers of English. Spelling errors were addressed with the Aspell English spelling correction program, using both American and British dictionaries; some files with excessive misspellings were discarded. Manual editing was required to distinguish misspelled words from technical terms and named entities that were not present in the Aspell dictionary. About 400 words were corrected.

Initial sentence alignment for the corpus was thrown off by the presence of an extra blank line in one Arabic file; this was removed. Three empty files were also removed. There was also a segmentation error in the corpus in which some Arabic lines contained just a period, usually

corresponding to a blank line in the English file. During concatenation, 868 of these defective lines were removed.

Other sentence alignment and sentence and word segmentation errors were apparently due to the way the corpus was collected. The QED Corpus derives from the AMARA website, which enables crowd-sourced transcription of video; the AMARA interface presents the worker with 4-second segments of video to transcribe, and these are subsequently assembled into a larger text (Zukerman, 2013) [34]. This way of assembling transcription segments led to sentences being split across lines, or multiple sentences within a single line. In addition, sometimes fragments were combined without spaces, leading to files in which each line has run-together words in the middle of the line. These errors were detected primarily in the English side of the corpus.

Sentences that were split across lines sometimes left the matching English and Arabic words on different lines. An attempt was made to reassemble split sentences using line-final punctuation to identify the end of the sentence. This method failed for some files which lacked punctuation; this also failed to detect lines that contained sentence-final punctuation in the middle of a line. A restriction was therefore placed on the result of reassembling sentences, to discard files which had a high average words-per-line after concatenation, and to discard files in which more than 5 lines and 500 characters were concatenated.

Files with run-together words were analyzed and processed with Aspell. A human looking at these files can analyze the problem easily, based on what is reasonable to expect in the sentence, but automatic, rule-based correction faces some difficulties.

An Example of Run-Together Words in the QED Corpus

> *It's the difference between divergent **thinkingand** convergent thinking. You have to separate the two so that you can diverge your **thoughtsand** come up with this great collection of ideas, and then once you have this great **collectionof** ideas, you focus on the convergent thinking.*

The Aspell spell checker was used to report the proportion of run-together words in a file; in order to protect technical terms and named entities, the program only reported lowercase words in the middle section of the line. Out of 19K total English files, 57 were identified as having a high proportion of medial, lowercase run-together words. These were further processed using Aspell to split the problem words.

Initially the program split the unknown word into progressively longer sections of first word vs. second word, until two known words were detected. This led to unfortunate splits like *thoughtsand* > *thought sand* instead of *thoughts and* and *monkeysin* > *monkey sin* instead of *monkeys in*. Manual editing was used to correct this type of problem. In the future, a word frequency list could be applied to select the best split, or language modeling could be applied to determine which split creates the most reasonable sentence.

Some problems remain. The restriction to lowercase words protected names from being split, but unfortunately also meant that some run-together words were left unrepaired, when the first word was capitalized as the start of the sentence, or when there was a name run together with the following word. The corpus also contains unusual punctuation, truncated words where a speaker changed words mid-sentence, and HTML residue, so normalization and tokenization is needed.

Altogether, the QED corpus cleanup process yielded 889 useful files, comprising 72,475 lines, which was about 70% of the original 1223 lines.

2.2.3.9 WMT News Commentaries: Sentence Alignment Errors from Windows Carriage Return Characters

The WMT 2014 Russian news-commentary data appeared to have sentence alignment problems. Length disparity across parallel text sentences is one way to detect mis-alignments. A program was written to flag lines where the different language versions vary by a factor of n; setting n to 3 disclosed that 21.5% of the Russian news-commentary lines might be mis-aligned. This compared to amounts of less than 3% for the other WMT files. The mis-alignment was caused by the presence of Windows carriage return characters, (CR) 000D, instead of Linux-style line feed (LF) characters, 000A. A program was written to diagnose and remove CR, and the files were re-aligned. In order to create an improved seed lexicon for alignment, programs were written to collect names from the WMT wiki titles and names files into lexicon format, and to harvest unigram, bigram, and trigram phrase table entries as additional lexical entries. After re-alignment, only 0.45% of the news-commentary sentences exhibited length disparity > 3.

For the WMT 2015 competition, a check of the news-commentary files showed the carriage-return issue for multiple languages, as shown below in Table 94. These had to be removed before sentence alignment.

Table 94: Carriage Return Characters in *WMT 2015* News Commentary Data

Carriage Return (CR) Characters in WMT 2015		
Language	**CR Chars**	**Total Lines**
ru-en.ru	2991	222164
ru-en.en	3072	222164
cs-en.cs	3026	191963
cs-en.en	3141	191963
fr-en.fr	2526	253041
fr-en.en	2566	253041
de-en.de	3535	272807
de-en.en	3615	272807
es-en.es	3171	266334
es-en.en	3279	266334
ar-fr.ar	0	144033
ar-fr.fr	0	144033

2.2.3.10 WMT Untranslated Parallel Text

Parallel text sometimes contains untranslated lines, which then leads to same-language phrase table entries that can cause problems for MT. For example, in previous SCREAM Lab English>French MT, the presence in the French file of the borrowed name, "Nine Inch Nails", led to phrase table entries that matched "nine" to "nine", instead of the French translation, "neuf".

Training Data:

EN: So, this is a project for Nine Inch Nails.

FR: Ceci est un projet pour Nine Inch Nails

Phrase Table Entries:

0 nine nine 1.0 0.004847 1

0 nine_inch nine_inch 1.0 0.00230809293

In subsequent MT, the entry, *nine*|*nine*, was sometimes chosen over the correct English-French translation, *nine*|*neuf*. Such source|source phrase table entries appeared to have a disproportionate effect on translation, so removing same language entries from the phrase table is a high priority.

The WMT 2014 Russian news-commentary files had problems with wrong-language text, including one Chinese article and one Russian article in the English file. The Russian article is of particular concern, since this creates Russian/Russian parallel text.

The WMT 2014 Russian/English news-commentary files included a Russian article in the English file. Duplicate detection was applied to remove these and any other identical lines from the parallel text before the creation of the phrase table.

Additional Russian text in the WMT 2014 English training files came from quotations, borrowed words, and apparent translation omissions; these can also lead to same-language phrase table entries. A program was written to remove phrase table entries that contain Russian in the English; this removed 13,619 entries from a total of 47,360,260 entries.

The WMT 2015 news-commentary files contained repeated boilerplate text, in which the web-scraping program apparently picked up the same standard text at the end of each news article. A set of ten English lines occurred 4782 times in both the English and the Russian file:

> *Your message has been sent successfully, thank you for contacting us. Secure rights Your First name*
>
> *Your Last name*
>
> *Your Email*
>
> *....*
>
> *Cancel*

A duplicate detection program was used to remove the boilerplate sections, as well as several lines of untranslated English headlines.

For the subsequent 2016 WMT competition, a decision was made to remove not just duplicate lines, but, lines which share 10% or more of the words across parallel text in all of the training data files. Because the amount of training data is large, removing a large amount of lines is acceptable if it prevents the creation of same-language phrase table entries. When using the more aggressive 10% matching threshold, it was necessary to exclude digits and punctuation from the word count, since these may match across languages. The comparison of matching lines was repeated following lowercasing, to pick up matching phrases that differ only in capitalization.

Altogether, the cleanup process removed about 8.6% of the 2016 training data (Table 95).

Table 95: Cleanup of *WMT 2016* Russian/English Training Data

WMT 2016 Russian/English Training Data		
Lines	**Clean Lines**	**File**
878386	723256	commoncrawl
196245	189293	news-commentary
1000000	983259	Yandex
2074631	1895808	TOTAL

2.2.3.11 WMT Russian/English Data

Described here are corrections made to the WMT training data in 2014, 2015, and 2016, in addition to the carriage return and untranslated text issues discussed above. The WMT training data for these years includes the Common Crawl, the Yandex corpus, the wiki titles and names files, and the news-commentary files.

SCREAM Lab's work during the earlier WMT competitions had identified and corrected a large number of problems in the Common Crawl, and these corrections were also used in subsequent WMT competitions.

The Common Crawl clean-up program was put to additional use starting in 2014: The original program worked with parallel text, and a new version was written to work on a single Russian input file. This new version was applied to the monolingual Russian data used for language modeling. Also, the part of the program that detects Ukrainian in the Russian text was used to create a separate program for language identification, using the Ukrainian-specific characters, I Ї Ґ Є, and i ї ґ є.

The Russian Yandex corpus was processed to remove control characters and to normalize words with mixed Latin and Cyrillic characters. In 2014 a total of 4,505 words with mixed characters were normalized to either all Cyrillic or all Latin characters. Some additional cleanup was applied in 2016 to the English side of the Yandex data, to deal with some mixed spellings including the alternation of Latin I 0049 and Ukrainian I 0406 characters in Roman numerals, and encoding errors such as an apostrophe being encoded by either the Ukrainian letter, ґ 0491, or the Russian letter, т .

A program was written to harvest the parallel Russian and English text from tab-delimited data in the wiki names and titles files. A second program was written to remove duplicate lines. Altogether, 6415 duplicate lines were removed from the titles file (about 1.4%), as well as 94 lines of the names files that were already present in the titles (about 0.17%). Lines in which the same Russian name is listed with a different English translation were retained.

The WMT 2014 English news-commentary file had some wrong language text, including one Chinese article and one Russian article. In 2015, wrong language included 763 lines of Hindi in various places in the English file. Wrong language text in the Russian file included one French and two Spanish articles, as well as one article with bad encoding.

The WMT 2015 Russian file news-commentary file included HTML fragments, soft-hyphen, 00AD, and a large number of non-breaking space characters, 00A0. A program was written to

replace non-breaking spaces with ordinary spaces except within digits, to allow for the European format for thousands, such as 10 000.

A previous program was used to identify and correct mixed-alphabet spelling in the Russian, such as the word, проявление "display", which has a Latin o 006F in place of a Cyrillic o 043E. This program also identified mixed-alphabet sequences like вNewsweek "in-Newsweek" and CO2будет "CO2-will-be", which should be handled by splitting up the run-together words instead of converting characters.

Some sections of the news-commentary files had alignment errors in which multiple sentences in one line were matched to a single sentence in the parallel line in the other language. There were also multiple instances in which one side had a blank line where the other line had text. General sentence alignment problems were not further addressed, but the blank lines and their corresponding text lines were simply removed. This affected about 2,000 lines.

The WMT 2016 Russian test file, newstest2016, was processed to correct non-breaking spaces and some mixed spellings. The WMT English newstest2016 file was processed to remove some sections in Hindi.

The English file was also noted to have British spelling. For example, there were 6 examples of the word, *favourite*, vs. one example of the word, *favorite*. An Aspell-based program was used to confirm that the 2016 file has a slightly more British tone than the 2015 file, by measuring the percent of words found in Aspell's American and British English dictionaries. British spellings like *hospitalised* and *stabilises* did in fact show up as OOV words in one of the SCREAM Lab baseline English-to-Russian MT systems. In future translations, it might be useful to first normalize the training and test files to all American or all British spelling using the Varcon variant conversion program.

2.2.3.12 Gazeta.ua Parallel Russian/Ukrainian Dataset

The Gazeta.ua online newspaper was used as a source of parallel Russian and Ukrainian text. The articles were collected in their HTML-tagged form; a program was written to extract the text from the tagged format. This program also screened for errors that occurred during the text harvest, such as "404 File Not Found" and "503 Service Temporarily Unavailable" when only one of the languages was available for an article. A list of these files was kept for possible future attempts to retrieve the missing articles.

The extraction program was also used to record the category, language, and talk id information for each file, along with the article title and Russian and Ukrainian source URLs. The Gazeta website designates a category for each article, such as politics, economics, sports, science, etc. These could be used in the future to sort articles for domain adaptation or for the creation of specialized language models (see "Russian Online News Meta-Data" in section 2.2.4.3 "Meta-Data").

The extracted text was checked to confirm that only Russian was present in the Russian articles, and only Ukrainian was present in the Ukrainian articles. Small amounts of wrong-language text were accepted based on the presence of names and borrowed words.

The use of online articles leads to the collection of some undesirable material, such as boilerplate text that is repeated on each webpage. This type of text may also remain untranslated in the different language sections of the website. In the Gazeta dataset, a copyright statement was

repeated of 23,000 times. Section headings may also generate repeated text that should be removed. There was also an issue with articles that have a "teaser" section containing the title and first few lines of the article, ending in an ellipsis. These lines are then repeated in full in the main body of the article.

Some difficulty was caused by the use of different HTML mark-up styles over time, including the use of <p>, , <br/ > and <dl><dt> formats. There were 4,712 instances of run-together sentences in the extracted text, many of which had been separated by <br> break tags in the original text.

Another issue was the presence of hyperlinks to related articles, usually in the form, "<READ MORE: other article title here>". These should be removed, but other links may be present that form part of the article text, such as "<website abc> reports that...", and these should be preserved. The extraction program was modified to tag all hyperlinks for future processing.

During extraction, some sentence alignment errors occurred. One source for alignment problems was the presence of reader comments, which typically were posted on only one of the two language versions of an article. When extracted, these comments were paired with just the word, "Comments", in the other language. Another source of alignment error was the erroneous splitting of sentences at abbreviations such as м."meters" or руб. "rubles". The extraction program should be modified to be sensitive to the non-breaking prefix list for each language.

2.2.4 Grammatical Annotation of Corpora

Three forms of grammatical annotation were conducted on corpora as follows. Subsection 2.2.4.1 addresses morphological annotation performed on several languages through stemming and lemmatization with modified tools, plus, a newly developed Russian stemmer tool. Subsection 2.2.4.2 covers NE tagging work performed on Russian and Chinese corpora. Subsection 2.2.4.3 discusses annotation with metadata for Russian, Ukrainian, Chinese, and HindEnCorp data.

2.2.4.1 Morphological Annotation

Morphologically complex languages introduce variation for MT, since nouns or verbs may have many different inflectional endings. Removing inflectional endings can be helpful to reduce data sparsity and facilitate phrase table creation. Some programs remove the inflections and leave the stem, which may or may not correspond to an independent word. Other programs replace the inflected word with its lemma, which is a canonical form, such as the infinitive of a verb or the nominative singular form of a noun. Often, stemming is combined with annotation of the morphological properties of the word.

Stemming and lemmatization were used for several languages, including English, Russian, Ukrainian, and Hindi. Existing programs were adapted for all of these, and a new stemmer was created for Russian.

English Lemmatizer

The existing TreeTagger program was adapted to create lemmatized English sentences. TreeTagger returns lemmas and POS, as shown in Table 96.

Table 96: Example Lemmatized English Sentence from *TreeTagger*

TreeTagger Output		
Word	**POS**	**Lemma**
It	PP	it
has	VBZ	have
never	RB	never
been	VBN	be
easy	JJ	easy

For use as a sentence lemmatizer, it was necessary to override some of TreeTagger's standard formatting, such as lowercasing the lemmas and replacing digits with a placeholder.

Russian Stemmers and Morphological Analyzers

Several existing morphological analysis programs were used to process Russian text, including TreeTagger, Mystem (from Yandex), RFTagger, and Stemka. TreeTagger, Mystem, and RFTagger return lemmas along with the POS and morphological information; Stemka creates output in the form of stem|suffix, or stem|suffix|suffix if there are both derivational and inflectional suffixes on the same word.

TreeTagger and RFTagger are general systems that can be used with a Russian model. Mystem was written for Russian, and Stemka was written for Ukrainian and Russian. In general, Mystem and Stemka appear to be more accurate in analyzing Russian.

A script was written to call Stemka, specifying Russian or Ukrainian input, applying encoding conversion, and removing suffixes to output the bare stem.

The Mystem options {-n -c -d -i --eng-gr} provide morphological annotation, where n=one word per line, c=copy input, d=disamiguate possible POS tags, i=print grammatical information, and eng-gr=use English instead of Russian in the output tags. The Mystem output can then be parsed to return the lemma, identify NE, or retrieve case, person, number, and gender information (see section 2.1.3.12 “Translating from an Inflectional Language via Source Text Annotation and Re-Ordering”, “Pre-Translate via Dictionary” in section 2.1.3.14, and “Translation of Named Entites via Transliteration Mining” in section 2.1.3.15).

Mystem escapes certain punctuation characters, as shown below in Table 97. These must be converted back to their original forms when working with Mystem output.

Table 97: *Mystem* Escape Sequences for Punctuation Characters

Mystem Punctuation Character Escape Sequences	
\u2015	―
\u2026	…
\u2082	₂
\u2116	№
\xAB	«
\xB0	°
\xBB	»
\xD7	×

Similarly, Mystem escapes Cyrillic characters to their codepoints when they are attached to numerals (Table 98); these must also be restored.

Table 98: Example of Cyrillic Characters Escaped in *Mystem* Morphological Analysis

Mystem Cyrillic Character Escape Sequences	
Original Sequence:	в 5см от
Mystem Output:	в{в=PR=} _5\u0441\u043C_ от{от=PR=}

A script was written to call TreeTagger. A spelling adjustment is required for TreeTagger, which does not recognize the Russian character, 0451 ё /yo/. The stressed vowel, /yo/, is written as ё in precise Russian text. It can also be replaced by the symbol, e, which technically represents the pronunciation, /yɛ/. The symbol, ё, is traditionally only written in dictionaries, learning materials, and foreign words, but it is starting to be used more frequently, and is often seen in online materials.

The TreeTagger morphological analyzer for Russian does not recognize words in ё, so we need to normalize ё to e for that process. For example, TreeTagger fails to recognize the adverb, ещё /yɛ shtsh yo/ “still”, and attempts to analyze it as a proper noun. Converting the spelling to еще yields the correct tag. The TreeTagger output for both is shown below in Table 99.

Table 99: Effects of Different Russian Spellings on *TreeTagger* Morphological Analysis

TreeTagger Morphological Analyzer Output			
Word	**Tags**	**Lemma**	**Meaning of Tags**
ещё	Npmsny	ещё	[noun, proper, masculine, singular, nominative, locative]
еще	R	еще	[adverb]

Data supplied to TreeTagger should therefore be normalized, converting 0401 Ё to 0415 E and 0451 ё to 0435 e.

There are reasons to retain the ё / е distinction when not using TreeTagger. There are a handful of minimal pairs in which both spellings are meaningful, such as the pronouns, все /vsyɛ/ ("all " nom. pl., "everyone"), and всё /vsyo/ ("all", nom.sg., "everything"). The spelling distinction also remains useful in certain borrowed words and names.

Previous work in the SCREAM Lab led to the development of a simple Russian stemmer, which uses lists of potential noun, adjective, and verb suffixes, and returns the stem that remains after removing the longest potential suffix. Rules are used to restore soft/palatalized stem spellings after the removal of a suffix with a palatalizing vowel.

Stemming, as opposed to lemmatization, is useful for various tasks, and this program can be adjusted to produce output in various formats. A version of this stemmer, *SCREAMStemmer.java*, was shared with other researchers.

Hindi Morphological Analysis

Two possible programs were identified for the morphological analysis of Hindi; the light stemmer by Ramanathan and Rao (2003) [35], and the Hindi plugin for the GATE translation system, which includes a Hindi POS tagger.

2.2.4.2 NE Tagging

Chinese Statistical NE Tagging

A conditional random field (CRF) named entity tagger was created for Chinese, using the SIGHAN 2006 LDC training data[10]. The following NE features were used: n-gram features, segmentation features, part of speech, and list membership. Segmentation features specify whether a character is at the beginning, inside, end, or outside of a word. Test data was created by hand-annotating the occurrence of NE in the IWSLT test files. The NE were tagged as location (loc), organization (org), person (per), and geo-political entity (gpe).

Variants of the NE tagger were created using different combinations of training data, including the IWSLT training data. Table 100 shows the results of NE tagging the IWSLT test2010 file with the different training sets, scored by *f-measure*, which is a combination of precision and recall.

[10] http://sighan.cs.uchicago.edu/bakeoff2006/instructions.html

Table 100: NE Tagging Scores for *IWSLT test2010* Chinese File

F-measure for NE Tagging of IWSLT test2010 Chinese File					
Training Data	**All NE**	**gpe**	**loc**	**org**	**per**
SIGHAN	0.712	0.651	0.136	0.401	0.670
IWSLT	0.821	0.838	0.361	0.622	0.844
SIGHAN+IWSLT	0.830	0.825	0.495	0.693	0.836

Output of the NE tagger was formatted for use in Moses, with the Chinese NE pre-translated into English according to existing NE lists. Word frequency was used to distinguish among multiple candidates. Words tagged as NE that could not be pre-translated by lists were further processed to consider punctuation variations, and to transliterate borrowed words. A problem was identified for geo-political entities in which adjective forms were being translated as noun forms; POS tags were subsequently used to exclude adjectival geo-political entities. Another problem involved words with attached digits, which required a change in the character segmentation program.

Russian Statistical NE Tagging

The Chinese NE tagger from the previous section, “Chinese Statistical NE Tagging”, was adapted to create a Russian NE tagger. Features were added for capitalization, and Mystem was used for the Russian part-of-speech tagging component. The Russian NE tagger was trained on the Russian Wikipedia NE data of Nothman et al. 2013 [36], which includes multi-word entities (MWE). The GIZA++ word alignment program was used to derive additional individual word NE from the MWE.

NE lists for pre-translation were stemmed to reduce variation from Russian inflectional endings. Input text stemming was restricted to nouns and adjectives, to avoid accidental similarities between NE and stemmed verb forms.

Using Mystem for Russian NE Tagging

The Mystem program tags Russian named entities as family name (famn), personal name (persn), patronymic (patrn), and geographical name (geo). See “Pre-Translate via Dictionary” in section 2.1.3.14 and “Translation of Named Entites via Transliteration Mining” in section 2.1.3.15 for applications that pre-translate the NE into English.

An initial analysis of Mystem output showed that it was generating unusual NE tags, including a personal name tag for the preposition, из /iz/ “from”. Mystem can be set to report all possible POS and morphological analyses, which causes it to report иза /iza/ “Isa” as a possible name. Setting the –d “disambiguate” flag to make Mystem report just the most likely POS prevents this problem.

Also, Mystem has difficulty tagging borrowed names with apostrophes, such as О'Нил “O'Neill”, which is treated as two separate words.

Lowercasing has an adverse effect on Mystem's recognition of NE; personal names were more likely to be interpreted as common nouns when lowercased (Table 101).

Table 101: Effects of Lowercasing on *Mystem* NE Tagging of Russian

	Mystem NE Tagging	
	Input	**Mystem Output**
English:	... I went to jail with Dr. King, in 1963.	
Russian:	... я попал в тюрьму с доктором Кингом, 1963.	Кингом{кинг=S,famn,m,anim=ins,sg} “King” surname
Russian-lowercased:	... я попал в тюрьму с доктором кингом, 1963.	кингом{кинг=S,m,inan=ins,sg} “king” common noun

2.2.4.3 Meta-Data

Corpora may be usefully annotated with meta-data such as text origin, speaker, date, topic, etc. The meta-data from the TED Talks and from the WMT Russian/English files were used in domain adaptation, as discussed earlier under section 2.1.3.16 “Techniques in Domain Adaptation for MT”. Other annotations are discussed here for the IWSLT Chinese/English UM Corpus, the HindEnCorp data, and for Russian online news sites.

IWSLT Macau Chinese/English Corpus Meta-Data

The IWSLT Macau (UM) Chinese/English corpus is divided into sections that could be used for domain adaptation (Table 102). The contents of these sections were manually reviewed.

Table 102: Dissection of *IWSLT Macau* Chinese/English Corpus for Domain Adaptation

IWSLT Macau Chinese/English Corpus	
Section	**Characteristics**
subtitles	overlaps with TED Talk data
microblog	slang (at least in the English); truncated messages, which may be truncated at different points in the English vs. the Chinese
science	technical terms; geopolitical entities; some all-caps lines
laws	numbers in dates, headings
spoken	textbook-style short sentences, including questions
education	some sections probably originally Chinese, with odd English phrasing other sections probably originally English, with fluent English phrasing

HindEnCorp Meta-Data

The HindEnCorp dataset is divided according to text sources, such as TED Talks, Wikipedia, a political blog, etc. An initial analysis was made of the nature of text in each section, for possible domain adaptation. Differences in tokenization and lowercasing were noted within the different sections.

In addition, about 50,000 of the 287,000 lines were identified as names or single words which could be used as a preliminary dictionary for word alignment.

Russian Online News Meta-Data

Online news resources for Russian/Ukrainian and Russian/English text provide several types of meta-data that could be used for domain adaptation, including section headings, keywords, and hyperlinks. Several sources were analyzed for quantity of material in various domains.

For the Russian/Ukrainian online newspaper gazeta.ua, the main headings are culture, economics, history, life, politics, science, and sport (Table 103). Also of note are "interview" and "video" sections that might be useful in the future for work with speech recognition.

Table 103: Potential Domain Adaptation Meta-Data from Russian/Ukrainian Online Newspaper, *Gazeta.ua*

Section Headings in Gazeta.ua		
Category	**Russian**	**Ukrainian**
politics	15516	13461
economics	8061	7316
life	4084	3825
np (newspaper)	3473	3838
sport	2233	2704
culture	1837	1121
history	1346	1271
science	1041	1217
avto (auto)	1039	1054
regions	931	1083
politics-newspaper	991	885
real-estate	632	628
kiev-life	105	90
culture-newspaper	191	119

Korrespondent.net (Russian/Ukrainian) has article groupings that include: business, lifestyle, showbiz, sport, tech, Ukraine, and world. Chaskor (Russian only) publishes user-submitted articles in these departments: Society, Economy, Around the world, Culture, Media, Technology, Health, Exotica, Books, and Calendars.

Some online newspapers tag articles with keywords, while others include a "read also" section with links to related articles. For example, an article from Korrespondent on the election of the Greek president contains the word, "tags", followed by tags for president, Greece, parliament, and nomination:

ТЕГИ: президент, Греция, парламент, назначение

The Russian/Ukrainian online newspaper, gazeta.ru, uses Читайте также “read also” links with its lead story (or, in Ukrainian, Читайте також); these links could be used to build a semantic network of related articles. For example, a gazeta.ua article about weather in the Kerch strait in Crimea delaying ferry crossings concludes with a suggestion to read another article titled, “Kerch ferry stands – about 1000 cars in line”.

A compilation of about 300 Russian financial articles was made for use in domain adaptation. The Russian Common Crawl has an annotation file that gives the starting line and the number of lines in the article; with this information individual articles can be extracted. The annotation file was searched for URLs that contain keywords like "financial" and "stock", excluding accidental

matches like "stockholm". Selected articles were reviewed in order to exclude non-financial articles with financial keywords, such as instructions for working with accounting software. To this was added a group of 8 previously identified economic articles from the WMT15 data.

2.3 Laboratory System Admin Support

Due to the computational and data storage intensive Human Language Technology (HLT) research performed by the SCREAM Laboratory, there is a continuous need for System Administration support and IT investment to maintain the computational efficacy of the network.

The SCREAM Laboratory network is comprised of high-performance workstations, high-performance rack-mounted computational nodes, and various servers including numerous high-capacity storage arrays, MT servers, web servers, database servers, backup servers, authentication servers, software and hardware inventory servers, centralized configuration servers, performance monitors, etc.

Comprehensive coverage of system administration tasks, maintenance, and upgrades is beyond the scope of this document. Many frequent and/or routine system administration tasks, such as routine system maintenance, backups, system repair, system troubleshooting, and user support, also accomplished under this task order may not be listed.

Under the ICER contract, system administration, software maintenance and upgrades, and hardware maintenance and upgrades for the SCREAM Laboratory, are shared among multiple task orders. In many cases, non-trivial system administration tasks were split between different task orders. While, some significant or otherwise interesting tasks are described below, they may also appear in reports for other task orders as the work was split between multiple task orders.

- Integrated 22 new high-performance Linux compute/GPU nodes into the SCREAM network. Most of these systems have two Tesla M40 to support DNN processing, etc.
- Performed an upgrade of the SCREAM network backbone from 1 Gbps to 10 Gbps as detailed in “Development and Utility of Automatic Language Processing Technologies, Volume II” [37].
- Migrated backups from LTO-3 tape to SATA hard drives.
- Integrated hardware compression card with 10 Gbps/sec gzip compatible compress to maximize backup capacity and throughput for HDD backups. Hardware compression cards may also have future uses in filesystems with realtime compression (e.g. ZFS) and/or reducing latency when accessing compressed experimental data.
- Analyzed and documented Linux vulnerability scan detection false positives, etc.

2.4 Additional Activity

This section addresses miscellaneous related activities that did not fit within the delineation of Task Order 29 efforts in prior sections 2.1 through 2.3.

2.4.1 Festival Speech Synthesis System Training

Alan Black from Carnegie Mellon University (CMU) came to the SCREAM Laboratory and taught a four-day class on the Festival Speech Synthesis System.[11] Festival was originally developed by Alan Black at the Centre for Speech Technology Research (CSTR) at the University of Edinburgh, and is actively maintained and developed by personnel from CSTR, CMU, and the Nagoya Institute of Technology. As a companion to Festival, the Festvox project[12] includes documentation, scripts, and example databases for creating speech synthesis voices.

First Tutorial: Training and Testing Speech Synthesis Systems

The first tutorial involved training and testing a speech synthesis system to tell the current time.[13] Each participant recorded 24 prompts, derived phone alignments for the recorded audio, extracted pitch marks and Linear Prediction Coefficients (LPCs), and built a unit selection based speech synthesis system.

Second Tutorial: Comparison of Speech Synthesis Systems

The second tutorial compared unit selection and statistical parametric speech synthesis systems. Each system was trained using a single speaker from the CMU Arctic corpus.[14] The statistical parametric system was developed using CLUSTEGEN, which uses a Mel Log Spectrum Approximation (MLSA) filter to synthesize speech from Mel Frequency Cepstrum Coefficients (MFCCs).

Third Tutorial: Hindi CLUSTERGEN

The third tutorial created a Hindi CLUSTERGEN voice using graphemes instead of phonemes. This eliminates the need for letter-to-sound rules, thereby reducing the amount of expert knowledge that is required to develop a system for a new language.

Fourth Tutorial: Voice Format Conversion

The final tutorial converted the previously developed voices from Festival format to Flite format. Flite is a synthesis engine developed at CMU that is designed to run in real-time using a minimal amount of computer resources.[15]

---

[11] Available at: http://www.cstr.ed.ac.uk/projects/festival

[12] http://www.festvox.org

[13] Available at: http://www.festvox.org/docs/festival-1.2/festvox_10.html

[14] Available at: http://www.festvox.org/cmu_arctic

[15] http://www.speech.cs.edu/flite

3.0 CONCLUSIONS

The activities pursuant to ICER Task Order 29 pushed boundaries by exploring new methods and tools for the advancement of MT, ASR, and NLP, as well as tending to vital foundational components like corpora and lab resources. The activities, results, and highlights of this important work are summarized below.

An exploration into the adaptation of various submodularity techniques for language and speech processing provided insights into alternative ways of attacking the computing challenges from working on large datasets:

The results of these experiments were mixed, with some indicating improvement over conventional methods, and other yielding immeasurable change or even negative change. Favorable outcomes were observed in applying submodularity to data subset selection for MT, phrase-table pruning, and data selection for language modeling. Improvements in BLEU scores were shown to be obtainable with some of these methods, but, in some cases, were met with a tradeoff in terms of processing time. Applying submodularity to feature space reduction did not generally show favorable outcomes, nor did attempts to incorporate additional features beyond the standard source language n-grams. However, one of the feature space reduction methods, termed the *basic method*, did show BLEU score gains over the baseline.

Lessons taken from the submodularity research prompted recommendations for further investigation into this realm. The research team's *new theoretical approach* for data subset selection should be revisited with an expanded scope that utilizes a broader range of the *hyperparameter* variable. Applications for language modeling could be explored further by experimenting with the incorporation of model characteristics into the submodular data selection process. Also, experimentation should be attempted for neural network models, particularly in the pruning of large neural networks and partitioning of data for parallelized training.

Several community-sourced language and speech tools were analyzed and modified to improve their computing performance. This included adaptations to leverage GPU processing capabilities, optimizing internal memory data structures, parallelization, pruning, pipelining, and caching. GPU leveraging yielded a significant improvement in processing speed for the *CSLM* tool, albeit at the cost of introducing memory synchronization issues. A major improvement in speed was achieved for the *RNNLM* tool via a series of code optimizations. Modifications to the *HTS* tool's training process achieved a large reduction in training time. An attempt to improve speed for the resource-intensive *TriggerLM* tool fell short of making any gains. The culmination of these efforts working with these tools led to a list of best practices for adapting and integrating tools into the SCREAM Lab environment.

Tools and techniques that were born and cultivated by the SCREAM Lab included a PowerPoint speech-to-text proof-of-concept, *Experiment Reader*, *Reverse Palladius (RevP)*, and *Qahira*. An ability to take audio from a PowerPoint file, convert it into text, and re-insert it back into the original file was successfully achieved. *Experiment Reader*, which provides consolidated MT scoring and evaluation, was undergoing improvements to integrate with *iBLEU*. *RevP*, a tool for

transliterating occurrences of Chinese names within Russian text, was briefly invoked in some correspondence with an end-user. An effort was made to garner approval for public release of word alignment editor, *Qahira*, for the benefit of the global MT research community.

A multitude of translation methodologies and training data were analyzed to find ways of improving MT:

A statistically-trained Chinese word segmentation program was developed, and used for experimenting with word segmentation lattices and weightings.

Dependency parsing was employed to see if Russian-to-English translation could be improved, but the results did not indicate any advantage to this technique when sentence length is accounted for.

Examinations in inflection generation were conducted via dependency parsing and source annotation methods. For this, English verb annotation applied for English-to-Russian translation was ineffective at reducing inflectional errors, and, rather, it induced inflection errors in nouns and pronouns.

An analysis aimed at improving MT of social media text through monolingual human post-editing revealed slight BLEU score improvement, but such outcome is tempered by the significant time imposition attributed to the human intervention.

Source text annotation and re-ordering, in the *Yandex* style, were performed on Russian-English parallel data. This demonstrated improved BLEU scores, the best of which were obtained by annotation of nouns and adjectives without listing alternative case elements.

In addressing OOV words in Russian, A post-process selective transliteration based on word capitalization was applied to Russian OOV words. This technique did not completely recover the names, but it may provide a peripheral benefit by improving the readability of MT output.

An examination of Ukrainian-to-English translation by pivot method, using Russian as an intermediary, yielded interesting results. The BLEU score on the entire pivot process was lower than that of each the stages (Ukrainian-to-Russian, and Russian-to-English). The propagation of OOV words into the target English was compounded by the presence of Russian OOV words in addition to Ukrainian OOV words.

An experiment in statistical post-editing using *Systran* and *Joshua* yielded mixed results, essentially showing a translation performance boost for *Systran*. However, the performance of the combined *Systran-Joshua* is outdone by Joshua run solo.

The topic of error analysis was addressed by utilizing specialized tools, including some purpose-built in the lab, to examine translation errors produced by MT systems:

A performance comparison was conducted among three kinds of MT systems (phrase-based, hierarchal, and neural) using specialized error analysis tools, *Hjerson* and *MT-ComparEval*. The hierarchal system exhibited better fluency than phrase-based system, and the neural system had problems w/ NE and with excessive repetition errors. In another analysis, the specialized error analysis program, *SCLITE*, reported that high frequency words and mid frequency words contribute equal amounts of error.

A series of analyses were conducted for the purpose of comparing MT errors from AFRL systems to that of competing systems from WMT competitions, namely *EDIN* and *Yandex*. These analyses, largely focused on errors attributed to capitalization, revealed that acronyms and headlines are leading causes of mis-matched case in the output. Some issues with AFRL systems came to the forefront during the process; *AFRL-K* had difficulty with camelcase, and *AFRL-K* and *AFRL-J* had difficulty with truecasing.

Improvements were made to the *Hjerson* error analysis program, and the resultant modified version, *Revised Hjersion*, was put to practice on an English-to-Russian MT:

Among the modifications to *Hjerson* were the addition of a word-choice error measure, fixes to error detection and classification faults, reporting reordered words and separation distances, tokenizations to protect URL and hashtag text, reporting inflected word pairings, and extending output capability to include the A3 file format. A modified version of *Qahira* was created to help analyze final word alignment output produced by *Revised Hjerson*.

The *Revised Hjerson* program was used to process English-to-Russian MT output for the purpose of producing an inflection error performance baseline. These results may be used in benchmarking against other inflection generation techniques under development in WMT competitions.

Corpora-related activities included human translation work, discovery and harvest of parallel corpora, grooming to fix errors and inconsistencies, and grammatical annotation:

A number of bilingual and trilingual sources were discovered for Russian-Ukrainian-English corpora. The Slovnyk website was identified as a source for bilingual corpora in over 30 languages. The YeeYan website is a source of crowd-sourced Chinese-English corpora, but concerns had arisen of impending censorship from the Chinese government. Common Crawl was accessed to produce an English-Somali dataset, and some modifications were made to associated extraction code.

Grooming work was conducted on several datasets, particularly TED Talks, IWSLT English-to-Chinese datasets, WMT Hindi datasets, IWSLT 2016 QED Arabic-English, WMT news commentaries, WMT Russian-English datasets, and Gazeta.ua. Many of the datasets were found to suffer from several problems in common; sentence-internal repetition, punctuation normalization, sentence alignment, sentence and word segmentation, spelling, wrong-language text, duplicate lines, and mixed-alphabet spelling. Crowd-sourcing appeared to be a provocation for problems, particularly in TED Talks, IWSLT 2015 English-to-Chinese, and IWSLT 2016 QED Arabic/English.

Morphological annotation was performed on several languages via stemming and lemmatization. A new Russian stemmer, *SCREAMStemmer.java*, was created and shared with others in the community.

A newly created conditional random field (CRF) NE Tagger for Chinese experienced problems with certain geo-political names and words with attached digits from IWSLT test data. An examination of Mystem for use as a Russian NE tagger revealed difficulties with borrowed names with apostrophes, and with lowercasing.

Several data sources were analyzed for their suitability for meta-data annotation, including 300 Russian financial articles for use in domain adaptation.

The aforementioned research activities and achievements were made possible by the laboratory system admin support, which involved the continual upkeep and upgrading of SCREAM Lab computing infrastructure.

4.0 REFERENCES

[1] Kirchhoff, K., "Submodularity for Speech and Language Applications", *University of Washington* in cooperation with SRA International, Inc., 2016.

[2] Mirzasoleiman, B., Karbasi, A., Sarkar, R., and Krause, A., "Distributed Submodular Maximization: Identifying Representative Elements in Massive Data", *Proceedings of Neural Information Processing Systems (NIPS)*, Lake Tahoe, NV, 2013.

[3] Wei, K., Iyer, R., and Bilmes, J., "Fast Multi-Stage Submodular Maximization", *Proceedings of the 31st International Conference on Machine Learning (ICML)*, Beijing, 2014.

[4] Liu, Y., Wei, K., Kirchhoff, K., Song, Y., and Bilmes, J., "Submodular Feature Selection for High-Dimensional Acoustic Score Spaces", *Proceedings of the 2013 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*: pp. 7184-7188, Vancouver, BC, 2013.

[5] Moore, R. C., and Lewis, W., "Intelligent Selection of Language Model Training Data", *Proceedings of the ACL, 2010*: pp. 220-224, Association for Computational Linguistics, Uppsala, Sweden, 2010.

[6] Wamba, S. and Noumssi, G., "Le français au Cameroun contemporain: Statuts, pratiques et problèmes sociolinguistiques", *Sud Langues 2*: 1-20, 2003.

[7] Biloa, E., "La langue française au Cameroun: analyse linguistique et didactique", *ISBN 3-03910-431-4*, Peter Lang, Bern 2003, 2004.

[8] Schwenk, H., "Continuous-Space Language Models for Statistical Machine Translation", *The Prague Bulletin of Mathematical Linguistics: 93*: pages 137–146, 2010.

[9] Mikolov, T., Karafiat, M., Burget, L., Cernocky, J., and Khudanpur, S., "Recurrent Neural Network Based Language Model", *11th Annual Conference of the International Speech Communication Association (ISCA)*: pages 1045-1048, INTERSPEECH 2010, Japan, 2010.

[10] Mauser, A., Hasan, S., and Ney, H., "Extending Statistical Machine Translation with Discriminative and Trigger-Based Lexicon Models", *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*: pages 210–218, Singapore, 2009.

[11] Young, K., Gwinnup, J., and Reinhart, J., "Reversing the Palladius Mapping of Chinese Names in Russian Text", *The Tenth Biennial Conference of the Association for Machine Translation in the Americas (AMTA)*, San Diego, CA, 2012.

[12] Gwinnup, J., "Qahira: A Word Alignment Viewer and Editor", Unpublished Journal Article, *SRA International*, 2014.

[13] Sharoff, S. and Nivre, J., "The Proper Place of Men and Machines in Language Technology: Processing Russian Without any Linguistic Knowledge", *Proc. Dialogue 2011, Russian Conference on Computational Linguistics*, 2011.

[14] Kirchhoff, K., Tam, W., Richey, C., and Wang, W., "Morphological Modeling for Machine Translation of English-Iraqi Arabic Spoken Dialogs ", *Human Language Technologies: The*

2015 Annual Conference of the North American Chapter of the ACL (NAACL): pages 995–100, Denver, CO, 2015.

[15] Gotti, F., Langlais, P., and Farzindar, A., "Hashtag Occurrences, Layout and Translation: A Corpus-driven Analysis of Tweets Published by the Canadian Government", *9th edition of the Language Resources and Evaluation Conference (LREC),* Reykjavik, 2014.

[16] Shapp, A., "Variation in the Use of Twitter Hashtags", Qualifying paper in sociolinguistics, New York University, 2014.

[17] Borisov, A. and Galinskaya, I., "Yandex School of Data Analysis Russian-English Machine Translation System for WMT14", *Proceedings of the Ninth Workshop on Statistical Machine Translation (WMT)*: pages 66-70, Baltimore, MD, 2014.

[18] Mermer, C., Kaya, H., and Dogan, M.U., "The TUBITAK UEKAE Statistical Machine Translation System for IWSLT 2007", *International Workshop on Spoken Language Translation (IWSLT),* Trento, Italy, 2007.

[19] Paul, M., Arora, K., and Sumita, E., "Translation of Untranslatable Words -- Integration of Lexical Approximation and Phrase-Table Extension Techniques into Statistical Machine Translation.", *The Institute of Electronics Information and Communications Engineers (IEICE) Transactions on Information and Systems, Volume E92-D No.12*: pages 2378-2385, 2009.

[20] Ore, B., Gwinnup, J., Thorn, S., Hutt, M., Hoeferlin, D., Coate, W., Snyder, D., and Young, K., "Development and Utility of Automatic Language Processing Technologies, Volume II", *SRA International*, AFRL-RH-WP-TR-2014-0052, 2014.

[21] Young, K., Gwinnup, J., and Schwartz, L., "A Taxonomy of Weeds: A Field Guide for Corpus Curators to Winnowing the Parallel Text Harvest", *The Twelfth Conference of The Association for Machine Translation in the Americas (AMTA), Vol 2 MT User's Track*: pages 355-370, Austin, TX, 2016.

[22] Lee, M.D., Pincombe, B.M., and Welsh, M.B., "An Empirical Evaluation of Models of Text Document Similarity", *Proceedings of the 27th Annual Conference of the Cognitive Science Society*: pages 1254-1259, Mahwah, NJ, 2005.

[23] Tversky, A., "Features of Similarity", *Psychological Review*, 84(4): pages 327–352, 1977.

[24] Chetviorkin, I. and Loukachevitch, N., "Extraction of Russian Sentiment Lexicon for Product Meta-Domain", *Proceedings of COLING 2012: International Conference on Computational Linguistics (COLING) Technical Paper*: pages 593–610, , Mumbai, 2012.

[25] Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., Soricut, R., Specia, L., and Tamchyna, A., "Findings of the 2014 Workshop on Statistical Machine Translation", *Proceedings of the Ninth Workshop on Statistical Machine Translation*: pages 12-58, Association for Computational Linguistics, Baltimore, MD, 2014.

[26] Bojar, O., Chatterjee, R., Federmann, C., Haddow, B., Huck, M., Hokamp, C., Koehn, P., Logacheva, V., Monz, C., Negri, M., Post, M., Scarton, C., Specia, L., and Turchi, M., "Findings of the 2015 Workshop on Statistical Machine Translation", *Proceedings of the Tenth Workshop on Statistical Machine Translation*: pages 1–46, Association for Computational Linguistics, Lisboa, Portugal, 2015.

[27] Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huck, M., Yepes, A. J., Koehn, P., Logacheva, V., Monz, C., Negri, M., Neveol, A., Neves, M., Popel, M., Post, M., Rubino, R., Scarton, C., Specia, L., Turchi, M., Verspoor, K., and Zampieri, M., "Findings of the 2016 Conference on Machine Translation (WMT16)", *Proceedings of the First Conference on Machine Translation, Volume 2: Shared Task Papers*: pages 131–198, Association for Computational Linguistics, Berlin, Germany, 2016.

[28] Avramidis, E., "Qualitative: Python Tool for MT Quality Estimation Supporting Server Mode and Hybrid MT", *The Prague Bulletin of Mathematical Linguistics (PBML), 106*: pages 147-158, 2016.

[29] Klejch, O., Avramidis, E., Burchardt, A., and Popel, M., "MT-ComparEval: Graphical Evaluation Interface for Machine Translation Development", *The Prague Bulletin of Mathematical Linguistics (PBML), 104*: pages 63-74, 2015.

[30] Popović, M., "Hjerson: An Open Source Tool for Automatic Error Classification of Machine Translation Output", *The Prague Bulletin of Mathematical Linguistics (PBML), 96*: pages 59–68, 2011.

[31] Berka, J., Bojar, O., Fishel, M., Popović, M., and Zeman, D., "Automatic MT Error Analysis: Hjerson Helping Addicter", *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*: pages 2158–2163, European Language Resources Association, İstanbul, Turkey, 2012.

[32] Smith, J., Saint-Amand, H., Plamada, M., Koehn, P., Callison-Burch, C., and Lopez, A., "Dirt Cheap Web-Scale Parallel Text from the Common Crawl", *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*: pages 1374–1383, Bulgaria, 2013.

[33] Larkey, L., Connell, M., and Abduljalee, N., "Hindi CLIR in Thirty Days", *ACM Transactions on Asian Language Information Processing, 2(2)*: pages 130-142, 2003.

[34] Zukerman, E., "Review: Amara is a Web-based service that lets anyone transcribe and translate online video.", http://www.pcworld.com/article/2032787/review-amara-is-a-web-based-service-that-lets-anyone-transcribe-and-translate-online-video.html, *PCWorld*, April 15, 2013.

[35] Ramanathan, A. and Rao, D., "A Lightweight Stemmer for Hindi", *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL):, workshop on Computational Linguistics for South Asian Languages*, Budapest, Hungary, 2003.

[36] Nothman, J., Ringland, N., Radford, W., Murphy, T., and Curran, J.R., "Learning Multilingual Named Entity Recognition from Wikipedia", Artificial Intelligence, 194:151–175, 2013.

[37] Ore, B., Gwinnup, J., Thorn, S., Hutt, M., Hoeferlin, D., Coate, W., Snyder, D., and Young, K., "Development and Utility of Automatic Language Processing Technologies, Volume II", *SRA International*, AFRL-RH-WP-TR-2014-0052, 2014.

APPENDIX A: Ukrainian Parallel Text Resources

Ukrainian Parallel Text Resources

Introduction

This document describes the parallel text resources we have identified for translating Ukrainian into English. Most of this material was assessed around June 2014.

The authors of the following article found that pivoting through Russian was better than translating directly from Ukrainian into English:

"Translating from under-resourced languages: Comparing direct transfer against pivot translation" by Bogdan Babych, Anthony Hartley, Serge Sharoff

http://www.mt-archive.info/MTS-2007-Babych-1.pdf

While transliteration could be applied to Ukrainian to create Russian-like text, enough of the words are different to prevent a direct conversion. According to Wikipedia, Ukrainian is closer to Polish than to Russian; Ukrainian shares about 60% of its vocabulary with Russian. So we plan to try translation rather than transliteration into Russian.

Accordingly, we focus on collecting Ukrainian/Russian parallel text for machine translation, for a pivot process of Ukrainian > Russian, followed by (our existing models for) Russian > English. We also collect Ukrainian/English text for possible direct translation, and trilingual text for testing.

Useful search terms:

Russian: РУС or Русский

Ukrainian: УКР or Українська

English: США

The word *download* in Russian: скачать

Also note that the ISO language suffix for Ukrainian is uk but the country suffix is ua

Ukrainian/English

Dictionary

http://www.freelang.net/dictionary/ukrainian.php

This list can also be found at http://www.slovnyk.org.ua.

public domain

Ukrainian > English: 138,054 words

English > Ukrainian: 63,287 words

Ukrainian/Russian

Dictionary

www.slovnyk.org = www.slovnyk.org.ua (slovnyk = dictionary in Ukrainian)

Russian-Ukrainian dictionary: 111,810 entries

Ukrainian-Russian dictionary: 96,368 entries

Has downloadable dictionaries (GNU general public license) to and from each language pair from among: Belarusian Bulgarian Croatian Czech Danish Dutch English (UK) English (USA) Esperanto Estonian Finnish French German Greek Hungarian Icelandic Italian Latin Latvian Lithuanian Macedonian Norwegian Polish Portuguese Romanian Russian Serbian Slovak Slovenian Spanish Swedish Ukrainian

Also has an online search interface.

Corpora

InfoStream: A Russian-Ukrainian corpus, of which a file with 100,000 lines is freely available here: http://infostream.ua/ling/100-tys-win.zip The encoding is CP1251 (Windows).

OPUS: The Open Parallel Corpus

http://opus.lingfil.uu.se/

select Ukrainian and Russian from the drop down menus

News Articles

Summary of news sites:

- pravda -- articles named in parallel
- newsru -- articles named in parallel
- gazeta -- articles have different names, but matching ID numbers
- korrespondent -- articles have different names, but matching ID numbers
- SETimes -- articles named in parallel
- Mirror Weekly -- articles not named in parallel

Main links and examples of article naming

Pravda

www.pravda.com.ua

www.pravda.com.ua/rus

www.pravda.com.ua/news/2014/05/17/7025661/
www.pravda.com.ua/rus/news/2014/05/17/7025661/

Newsru
http://www.newsru.ua
http://rus.newsru.ua
http://www.newsru.ua/finance/20may2014/11_4.html
http://rus.newsru.ua/finance/20may2014/11_4.html

Gazeta
http://gazeta.ua
http://gazeta.ua/ru
http://gazeta.ua/articles/science/_riven-pozhivnih-rechovin-u-zernovih-padat
ime/556714
http://gazeta.ua/ru/articles/science/_uroven-poleznyh-veschestv-u-zernovyh-b
udet-padat/556714
Korrespondent
http://ua.korrespondent.net
http://korrespondent.net
http://ua.korrespondent.net/sport/3815532-selta-shakhtar-01-onlain-matchu-lihy-yevropy
http://korrespondent.net/sport/3815532-selta-shakhter-01-onlain-matcha-lyhy-evropy

SETimes
http://ukraine.setimes.com/
http://ukraine.setimes.com/ru
http://ukraine.setimes.com/uk
http://ukraine.setimes.com/en_GB/articles/uwi/features/2014/05/22/feature-01
http://ukraine.setimes.com/ru/articles/uwi/features/2014/05/22/feature-01
http://ukraine.setimes.com/uk/articles/uwi/features/2014/05/22/feature-01

Mirror Weekly

http://dt.ua/

http://zn.ua/

http://zn.ua/TECHNOLOGIES/ochki-google-glass-pomogut-slepym-slaboslyshaschim-v-povsednevnoy-zhizni-144659_.html

http://dt.ua/TECHNOLOGIES/okulyari-google-glass-dopomozhut-slipim-ta-lyudyam-zi-slabkim-sluhom-u-povsyakdennomu-zhitti-142907_.html

note on acronyms:

Russian: Зеркало недели = Zerkalo Nedeli = mirror weekly

Ukrainian: Дзеркало тижня = Dzerkalo tizhna = mirror weekly

Miscellaneous Sources

TED Talks

http://www.ted.com/translate/languages/uk

TED talks translated into Ukrainian

We have already collected 707 Ukrainian TED Talks in our database.

Checked for parallel uk/ru and uk/en talks in May 2014, and identified problem in extraction script.

World Bank

http://go.worldbank.org/GHF2B8NN40

http://web.worldbank.org/WBSITE/EXTERNAL/NEWS/0,,enableDHL:TRUE~lang:%240~menuPK:64256347~pagePK:34386~piPK:64256390~theSitePK:4607,00.html

Trilingual Ukrainian/Russian/English

SETimes http://ukraine.setimes.com has been publishing trilingual articles in a "Focus on Ukraine" section

SETimes articles are also translated into other languages: Shqip Srpski Bosanski Hrvatski Македонски = Albanian, Serbian, Bosnian, Croatian, Macedonian

The Institute for War and Peace Reporting http://iwpr.net/ has translated about 30 of the Russian/Ukrainian articles from www.pravda.com.ua into English, creating a trilingual resource.

A search on the iwpr website for “This article republished from Ukrainska Pravda” created a list of about 30 trilingual articles.

For example :

http://iwpr.net/report-news/ukrainian-leader-says-captive-officers-crimea-being-freed

http://www.pravda.com.ua/news/2014/03/26/7020430/

http://www.pravda.com.ua/rus/news/2014/03/26/7020430/

Mirror Weekly http://dt.ua was trilingual according to the Babych article, but the English link seems to be a relic site.

http://zn.ua/

http://dt.ua/

www.mirror-weekly.com

Press Services

There are two main sites that translate foreign press articles about Russia into Russian, inoPressa and inoSMI (ino=foreign, SMI=mass media). inoSMI focuses on opinion articles and uses a forum for reader comments, while inoPressa seems to have just articles. An interesting article about inoSMI: http://russiaprofile.org/culture_living/a1224865563.html

Some of the online newspapers with Ukrainian and Russian articles make use of these services. The site www.newsru.ua has a section www.newsru.ua/press which draws on www.inopressa.ru. This includes links to the original articles (in English, German, etc.) Meanwhile, www.pravda.com.ua/ uses inoSMI, with the section www.pravda.com.ua/inozmi/ .

---------- notes on terminology----------

inoSMI (Russian: иноСМИ, a derivation from "foreign mass media")

иностранные	сре́дства	ма́ссовой	информа́ции
foreign	means*	mass	information

*from a stem that means “middle” – parallel to our word “media”

other terms:

масс-медиа mass-media

средств массовой коммуникации (СМК) *media of mass communication*

APPENDIX B: A Taxonomy of Weeds: A Field Guide for Corpus Curators to Winnowing the Parallel Text Harvest

A Taxonomy of Weeds: A Field Guide for Corpus Curators to Winnowing the Parallel Text Harvest

Katherine M. Young[†] katherine.young.ctr.1@us.af.mil
N-Space Analysis, LLC
Jeremy Gwinnup jeremy.gwinnup.1@us.af.mil
Air Force Research Laboratory
Lane O.B. Schwartz lanes@illinois.edu
University of Illinois

Abstract

Modern machine translation techniques rely heavily on parallel corpora, which are commonly harvested from the web. Such harvested corpora commonly exhibit problems in encoding, language identification, sentence alignment, and transliteration. Just as agricultural harvests must be threshed and winnowed to separate grain from chaff, electronic harvests should be carefully processed to ensure the quality and usability of the resulting corpora. In this work, we catalog a taxonomy of problems commonly found in harvested parallel corpora, and outline approaches for detecting and correcting these problems.

This work is motivated by the lack of a standardized field guide outlining best practices for curating parallel corpora, especially those harvested from the web. Even the most-well curated parallel corpus is likely to contain some problems; even Europarl (Koehn, 2005), arguably the most widely examined parallel corpus, has undergone eight distinct revisions since its release in 2005. While this work is by no means comprehensive of all problems extant in corpus creation and curation, we nevertheless believe that a practical taxonomic field guide, laying out likely pitfalls awaiting corpus curators will represent an important contribution to our community.

1 Introduction

Statistical machine translation typically requires large amounts of translated parallel text to serve as training data for statistical translation models. End-users of machine translation may use inhouse data developed from years of prior human translation efforts (Plitt and Masselot, 2010; Hellstern and Marciano, 2014). A perhaps more common practice, developed over the past fifteen years (Resnik, 1998), involves the automatic harvest of parallel corpora from online resources, such as bilingual

[†] This work is sponsored by the Air Force Research Laboratory under Air Force contract FA-8650-09-D-6939-029.

web sites (Smith et al., 2013) or the crowd-sourced translations of the TED Talk transcripts (Cettolo et al., 2012).

Just as agricultural harvests must be threshed and winnowed to separate grain from chaff, electronic harvests may be carefully processed to ensure the quality and usability of the resulting corpora. Simard (2014) suggested the metaphor of weeds choking out cultivated plants to be more apropos than that of cleaning "dirt" from corpora. We adopt this terminology, identifying a broad variety of such *weeds* found growing wild in online data, potentially degrading the quality of harvested corpora. In keeping with this botanic metaphor, we use *zizania*, a Greek term for a type of weed that grows intermixed with wheat,[1] as a basis for our taxonomic nomenclature.

In this work, we present a taxonomy of weeds commonly found in harvested parallel corpora, and outline approaches for detecting and correcting these problems. At the highest rank, the taxa we present are categorized based on provenance: Do the errors originate from problems during automatic processing of the text (*zizania ex machina*) or from human failure (*zizania ex homine*)? We categorize six major types of the former (§2.1–2.6), as well as six major types of the latter (§3.1–3.6). Throughout this work, we consider weeds that have been previously identified in the established literature, as well as weeds that we have encountered that have not heretofore been described in the literature.

This work is motivated by the lack of a standardized field guide outlining best practices for curating parallel corpora, especially those harvested from the web. Even the most-well curated parallel corpus is likely to contain some weeds; even Europarl (Koehn, 2005), arguably the most widely examined parallel corpus, has undergone eight distinct revisions since its release in 2005. We believe that a practical taxonomic field guide, laying out likely pitfalls awaiting corpus curators will represent an important contribution to our community.

2 Zizania ex machina: Weeds of mechanical origin

We now survey various *zizania ex machina*: weeds that originate during automated corpus processing.

2.1 Wrong Language Text

Wrong-language text errors can occur during automatic collection of parallel text from websites. The scraping program may mis-identify similar languages, or the program may fail to notice a section of foreign text within a page produced in the correct language. For example, if the program is scraping an English-language site with hotel reviews, it may pick up some reviews written in French. Alternatively, the program may fail to exclude a section of text that has remained untranslated across pages of a multilingual site. These failures create two types of errors that can be automatically detected, *Source-Source* errors, and *Source-Other* errors.

[1] See, for example, the usage of *zizania* in the Greek New Testament (Matthew 13:25).

2.1.1 Source-Source instead of Source-Target

An example of Source-Source error occurred in the initial release of the IWSLT 2014 data (Cettolo et al., 2012), in which some of the parallel English-French text was provided untranslated, creating English-English data. This was subsequently corrected. Source-Source errors can be detected automatically by searching for sentences that are duplicated across parallel text; these are usually untranslated sections. Short duplicate sentences should be examined separately, since there can be some legitimate duplication if the text contains URLs, named entities, borrowed words, or quotations. Legitimate duplication at the token level can also be caused by cognates (for example, the English word *importance* matches French *importance*).

2.1.2 Source-Other instead of Source-Target

Examples of Source-Other errors can be found in the French side of the 10^9 English-French corpus (Callison-Burch et al., 2009), in which we find paragraphs in Greek, Russian, German, and other languages. Such Source-Other errors can be detected easily if the incorrect language has a different character set than the correct language. For example, a section of Greek within a supposedly French document can be easily filtered out by specifying a desired range of permitted Unicode code points.

For languages with similar alphabets, we apply a simple dictionary-based program to remove sentences with a majority of unknown words. Recent work (Zampieri et al., 2014; Lui et al., 2014) leverages character n-grams, POS sequences, and other features to train language discrimination systems for similar languages.

Depending on the application, thresholding may be desired to allow a specified amount of wrong-language text (for foreign names, borrowed words, quotations, etc.). On the other hand, web-scraped text from multi-lingual sites often contains isolated wrong-language phrases that we may want to remove, such as hyperlinks in multiple languages. Multi-lingual sites can also contain stock phrases like “Click here to login” that may remain untranslated across the site; these might also need to be removed.

2.1.3 An illustration of a specific language identification clean-up process

For languages with similar but not identical alphabets, detection programs can be written that are specific to that language pair. For example, the English-Russian Common Crawl data includes sections which are actually English-Ukrainian. Ukrainian has four characters not found in Russian which can be used to identify unwanted Ukrainian segments: UKRAINIAN I (i I), YI(ї Ї), GHE WITH UPTURN (ґ Ґ) or IE (є Є). We make an exception to allow UKRAINIAN I in Russian segments when it occurs in a potential context for a Roman numeral (adjacent to Latin X, I, V, x, i, v, or their Cyrillic counterparts).

Second, on the English side of the Russian-English Common Crawl, we find sections of text in other languages such as French. Both English and French use the Latin character set, but French uses special characters not typically found in English such as à é ê î ô oe ç; these could be used to identify the presence of French, with some proportion of exceptions allowed for borrowed words like

café. However, for the Common Crawl we we also want to detect other non-English languages like Spanish. Instead of relying on specific accented characters to detect

Experiment	Corpus Size	Filtered Corpus Size	Avg. Cased BLEU	Avg. Uncased BLEU
Baseline	878386	732129	25.39	26.59
Cleaned	772530	642746	25.73	26.95

Table 1: Before and After Common Crawl experiment results reported in BLEU

non-English text, we apply a spell checker to identify English text. We use the aspell[2] spellchecker to determine the proportion of words that are not recognized as English, and compare this to a set threshold to identify the wrong-language sections. We exclude from consideration words of 3 characters or less, because many short words have false friends in other languages (e.g., *die* in English and German, *on* in English and French).

We demonstrate the effectiveness of these techniques by taking a baseline WMT15 MT system and replacing the phrase and lexicalized reordering tables with ones generated from the Common Crawl corpus in both original and cleaned configurations. Table 1 shows the cleaned corpus yields a +0.34 BLEU improvement over the non-processed baseline even with a 12% reduction in corpus size.

2.2 Historical Encoding Errors

Portions of a corpus are sometimes encoded using a different character encoding scheme than the rest of the document. If not detected and corrected, this leads to an encoding cipher, where sentences appear shifted to an incorrect character range. Encoding errors of this type can also occur when extracting text from a PDF document.

In the Russian-English Common Crawl parallel corpus, a number of Russian source sentences are encoded using the 8-bit Windows-1251 character encoding scheme. Most sentences in this corpus are encoded using UTF-8; when Windows-1251 encoded sentences are interpreted as UTF-8, the Cyrillic characters incorrectly appear as characters from the Latin-1 supplement block. This can be corrected by shifting these characters ahead by 350_{hex} code points into the correct Unicode Cyrillic character range. An example of this code point shift is shown in Figure 1 below:

(**a**) Справка по городам России и мира.

(**b**) Ñïðàâêà ïî ãîðîäàì Ðîññèè è ìèðà.

Figure 1: Russian sentence (**a**) originally encoded as Windows-1251, interpreted as UTF-8 (**b**)

Encoding errors may also show up in isolated characters. We see this in some of the Common Crawl data, in which French accented characters have been converted to Cyrillic characters. For example, we find the words équipe and château written as йquipe and chвteau. This is the reverse of the Russian code point shift described above, and these errors can also be corrected automatically if we know that the Cyrillic characters are out of range for our text. The Common

[2] http://www.aspell.net

Crawl exhibits a variety of code point encoding problems in addition to those shown here. Out of range characters should be examined for code point shifts and encoding problems that could possibly be corrected.

Lang.	Set	Sentences w. repeat errors	Total sentences
French	dev2010	11	887
Chinese	dev2010	87	887
	tst2010	81	1570
	tst2014	13	1068
Farsi	tst2010	1	885
	tst2011	22	1132
	tst2012	343	1375
	tst2013	187	923
	tst2014	53	1131

Table 2: Number of sentences containing segment-internal repetition errors in IWSLT dev and test sets

Lang.	Year	Sentences w. repeat errors	Total sentences
Arabic	2013	3	155,047
	2014	5	186,467
Chinese	2014	550	177,901
Farsi	2013	5,749	81,872
	2014	8,987	112,704
French	2013	173	162,681
	2014	373	186,510
Russian	2013	109	135,669
	2014	145	185,205

Table 3: Number of sentences containing segment-internal repetition errors in IWSLT training sets

There can also be encoding problems with individual characters. A confusion between UTF-8 encoding and Windows-1252 encoding can lead to a single character such as 0xE28099 (’) being interpreted as multiple, single-byte characters: 0xE2 (â), 0x80 (€) and 0x99 (™) Notenbloom (2009). These single-byte/multiple-byte encoding errors can be corrected programmatically with existing tools. Finally, we note that the character U+FEFF may appear in some files as the residue of a byte order marker at the start of a file; this should be deleted to avoid confusion with the Arabic script character U+FEFF, which is a zero-width non-breaking space.

2.3 Bidirectional Reversal

Adobe’s Portable Display Format (PDF) is meant as a display format and does not focus on the orderly layout of data in the document’s container. This presents issues when extracting text in an

orderly fashion from PDF documents. Extraction issues are compounded when dealing with custom fonts and historical encoding schemes. Additional issues involve the display of Right-to-Left (RTL) text.

Sometimes, extraction of RTL text from PDF creates text in which the line is reversed, character-by-character. We can detect reversals automatically by checking the words against a dictionary or word-frequency list to derive a percentage of unknown words. We then compare that percent unknown against the typical score for text from that language. If the percent unknown is supsiciously high, we can use a program to character-reverse the line, and repeat the dictionary check; a better score on the reversed line confirms the reversal error. In correcting reversed lines, we need to be careful how we handle digits, which run left-to-right within right-to-left text in many Arabic-script langauges.

2.4 Automatic sentence alignment errors

When parallel sentences are aligned, typically via automated means, mistakes in sentence alignment lead to mis-aligned sentence pairs that do not represent mutual translations. Many parallel corpora are naturally aligned at the document level: A human translator translates a source document into a target language. However, most statistical methods that make use of parallel data require alignment at the sentence level, and automated sentence aligners may make errors.

Various automated techniques have been proposed to minimize the problem of mis-aligned sentences. Gale and Church (1991) proposed an automated length-based sentence alignment technique that compared the number of words in source and target sentences. Proposed extensions to length-based approaches include the use of cognate frequency (Simard et al., 1992) or other lexical cues (Wu, 1994). Structural tags (such as HTML elements) have also been proposed as an aid to guide sentence alignment (Resnik, 1998).

2.5 Segment-Internal Repetition and Chunking Errors

Processing errors may cause a sentence or sub-sentential fragment to be improperly duplicated within a given line. In many cases, such repetition can be automatically detected and corrected; examination of the corresponding parallel sentence can assist in this process.

The IWSLT 2014 data, for example, contain substantial cases of repetition errors, especially for certain language pairs (see Tables 2 and 3 on the preceding page). An example of a repetition error is shown in Figure 2 below:

Last year I showed these two slides so that demonstrate that the arctic ice cap, *which for most of the last three million years has been the size of the lower 48 states*, has shrunk by 40 percent.
L'année dernière, je vous ai présenté ces deux diapositives qui montraient que la calotte glacière arctique, *qui pendant ces 3 derniers millions d'année avait la taille des Etats-Unis sans l'Alaska,* ***qui pendant ces 3 derniers millions d'année avait la taille des Etats-Unis sans l'Alaska***, avait diminué de 40%.

Figure 2: Example of repeated phrase in English-French TED data. Within the French sentence, the words in ***bold italics*** represent an erroneous copy of the words in *italics*.

While some cases of repetition are not errors (the TED Talks in particular may contain repetition for rhetorical effect), the presence of high amounts of repetition errors in training data and development data can degrade machine translation quality; correcting the large number of repetition errors in the IWSLT 2014 Farsi test file improved Farsi-to-English performance by +1.53 BLEU.

Chunking errors occur when sub-sentential segments are automatically combined without the necessary spacing. For example, a small number of files in the QED Corpus provided to the IWSLT 2016 competition (Abdelali et al., 2014) exhibit a chunking error, in which each line has run-together words in the middle of the line (see Figure 3). This is probably an error in assembly. The QED Corpus derives from the AMARA website, which enables crowd-sourced transcription of video; the AMARA interface presents the worker with 4-second segments of video to transcribe, and these are subsequently assembled into a larger text (Zukerman, 2013). We found 57 files with mid-line chunking, out of 19K total English files.

Chunking errors create unknown words for machine translation. A human looking at these files can analyze the problem easily, based on what is reasonable to expect in the sentence, but automatic, rule-based correction faces some difficulties. A spell checker like aspell can be applied to detect and correct run-together words, but we have to protect named entities and technical terms which may not appear in aspell's dictionary. We also have to be careful to split the words in the correct place. Initially, we simply split the unknown word into progressively longer sections of first word vs. second word, until we found two known words. This led to unfortunate splits like *thoughtsand > thought sand* instead of *thoughts and* and *monkeysin > monkey sin* instead of *monkeys in*. A word frequency list could be applied to select the best split. Alternatively, language modeling could determine which split creates the most reasonable sentence.

It’s the difference between divergent **thinkingand** convergent thinking. You have to separate
the two so that you can diverge your **thoughtsand** come up with this great collection of
ideas, and then once you have this great **collectionof** ideas, you focus on the convergent thinking.

Figure 3: An example of chunking errors in the QED Corpus.

2.6 Harvested Machine Translations

When parallel corpora are harvested from the web, there is a danger that some of the parallel content was created by means of machine translation, rather than human translation. Attempts have been made to automatically identify machine-translated content using various machine learning techniques, including decision tree classifiers (Corston-Oliver et al., 2001), SVM classifiers (Gamon et al., 2005), maximum entropy classifiers (Rarrick et al., 2011), watermarking (Venugopal et al., 2011), and identifying the presence of characteristic MT errors (Antonova and Misyurev, 2011). The extent to which the inclusion of machine-translated content in MT training data harms translation quality of the trained system may depend largely on the quality of the harvested machine translations (Simard, 2014).

3 Zizania ex homine: Weeds of human origin

In this section we survey weeds of human origin that show up in translated text from online sources. In general, zizania ex homine are harder to correct than zizania ex machina, but some automatic correction is possible.

3.1 Mixed Alphabets

Words with mixed alphabets visually resemble correctly spelled words, but are treated as separate tokens in the machine translation process. Such words can be automatically detected and corrected, converting characters to the majority alphabet for that word when they have visually similar counterparts.

Word	Latin (L) or Cyrillic (C)	Meaning
она	LCL	she
сейчас	LCCCCC	now
MP3-плеер	LLL-CCCCC	MP3-player
MP3плеер	LLLCCCCC	MP3player
амазон.com	CCCCCC.LLL	amazon.com
іпациент	LCCCCCCC	iPatient

Figure 4: Examples of Mixed-Alphabet words. In the center column, we annotate each character of the corresponding Russian word as either Latin (L) or Cyrillic (C). For example, in the first row, the Russian word она is encoded such that the Latin characters *o* and *a* are used instead of the more appropriate (but visually indistinguishable) Cyrillic equivalents.

We have encountered mixed alphabet words in the Russian sections of the Russian-English Common Crawl and in the Russian transcriptions of TED Talks. This occurs when the translator uses a combination of Latin and Cyrillic characters to write a Russian word. The reason for these mixed spellings is unknown; perhaps it is due to limitations of the translator's input method, or perhaps it is influenced by typing both English and Russian words. For example, although the first letter and last letter in the word сейчас appear visually indistinguishable, in this instance we find that the former is U+0063 LATIN SMALL LETTER C and the latter is U+0441 CYRILLIC SMALL LETTER ES. We even find the Russian word она written with U+006F LATIN SMALL LETTER O and U+0061 LATIN SMALL LETTER A instead of the appropriate Cyrillic counterparts (U+043E and U+0430); this word is harder to correct, since the majority favors the wrong alphabet.

Some mixed alphabet spellings are deliberate, combining a borrowed English word with a Russian word. Figure 4 above shows examples of this behavior. Converting punctuated words on a part-by-part basis can protect some but not all of these deliberate mixed spellings from automatic conversion.

In addition to the mixed alphabet spellings in Russian, we find creative spellings in many languages that borrow from other character sets, or repurpose characters within the source alphabet, particularly for punctuation. Some examples are given in Figure 5 on the next page. Determining how to correct such creative spellings generally requires human intervention.

3.2 Mixed Morphology

When a translator brings in a borrowed word through transliteration, he or she may choose to inflect the borrowed word using target language morphology. For example, in Urdu text we

Language	Character Written		Character Intended	
Urdu	U+002D -	LATIN HYPHEN	U+06D -	URDU FULL STOP
French	U+00A8 ¨	LATIN DIAERESIS	U+0022 "	LATIN QUOTATION MARK
Russian	U+0431 б	CYRILLIC SMALL LETTER BE	U+0036 6	LATIN DIGIT SIX
English	U+006F o	LATIN SMALL LETTER O	U+00B0 °	LATIN DEGREE SIGN

Figure 5: Examples of Creative Spelling.

find the borrowed English word *leader* with the plural suffix /-wn̲/, creating لیڈروں /lyḏrwn̲/, as well as the borrowed word with the original English plural form (leaders), لیڈرز /lyḏrz/. Names in particular are subject to variation in the application of target language morphology. An examination of names borrowed into Russian from English in the TED Talk data showed this range of behavior: a) first and last name both uninflected, b) first and last name both inflected, c) last name only inflected. Examples are shown in Figure 6; all three examples are possessive structures which should occur with genitive case.

Russian Text	Phonemes	English Text	Annotation Type
песню Уитни Хьюстон	/uitni x'yuston/	a Whitney Houston song	a) neither name inflected
закон Артура Кларка	/artur+a klark+a/	Arthur Clarke's law	b) both names in genitive case
Книга Эл Гора	/ɛl gor+a/	The Al Gore book	c) last name in genitive case

Figure 6: Examples of Mixed Morphology.

Inflected borrowed words often show up as out-of-vocabulary (OOV) words in MT output. If OOV words are going to be transliterated (see §3.3), it is useful to first apply a stemmer to remove any inflectional endings. Lexical approximation can sometimes rehabilitate inflected borrowed words and allow them to be translated (Mermer et al., 2007). Alternatively, Schwartz et al. (2014) identify inflected OOV words at the start of the decoding process, and replace them with variant inflected forms from the phrase table.

3.3 Transliteration of Names and Borrowings

Borrowed words and names may occur in transliteration, with the original sounds mapped into the characters of the new language. While such coinages are not errors, they are subject to variation that creates problems when an MT system attempts to relate them to the original forms.

Statistical methods may be applied to deal with this variation in transliteration, as for example in Durrani et al. (2014). Our work focuses instead on improving rule-based transliteration,

which maps characters into their typical sound values. Because there can be variation in the character-to-sound mapping in both languages, the output of rule-based transliteration is often faulty. This output can be improved by constraining the results to actual English spellings. In particular, we address the recovery of named entities that persist as OOV words in the output of machine translation. We describe two ways to constrain the results of named entity (NE) transliteration into English, one using an English pronunciation dictionary, and another using parallel training data to create a transliteration-based map of NE pairs.

3.3.1 Recovering Names via Transliteration in Conjunction with an English Pronunciation Dictionary

Rule-based transliteration can be improved by leveraging a target language pronunciation dictionary. We adapt the CMU English pronunciation dictionary[3] to guide transliteration from Russian into English. Because vowel spellings may be variable, we create a fall-back representation for each word in which all vowels are converted to a placeholder character, @. We derive a word frequency count from the training data and record the frequency count for each dictionary entry. We also supplement the pronunciation dictionary by noting any words in the WMT 2014 Russian data (Bojar et al., 2014) that are not listed in the CMU dictionary, and deriving their phonetic forms via Sonic (Pellom and Hacioglu, 2001).

When we run our transliteration program, we first map the Cyrillic characters into their typical sounds, recording multiple possibilities where appropriate. Next, we compare these phonetic mappings to the phonetic entries in the English pronunciation dictionary. We try to find words which match the sound pattern for both consonants and vowels; failing that, we use the vowel placeholder representations and allow @ to match any vowel or sequence of vowels. If there are multiple candidate words, we select the word with the highest word frequency count. We output the English spelling of the chosen word.

3.3.2 Recovering Names via a List of Transliterated NE Pairs

We apply transliteration and NE tagging to create a list of NE pairs from parallel Russian-English text; this list can subsequently be used to either pre-translate NEs, or to recover OOV names in the MT output. First, we apply the mystem[4] morphological analyzer to tag NE in the Russian text. For each NE, we then use rule-based transliteration to get a phonetic form, from which we identify possible matches in the English sentence. We record the best match along with the Levenshtein edit distance between the phonetic form and the English spelling, normalized for word length. NE pairs with a distance score below 0.66 are stored in a NE list that can be used to translate Russian NEs. When applied to the Russian-English WMT 2014 training data, this method generated a list of 216K potential NE pairs.

3.3.3 Third Language Mappings

[3] http://www.speech.cs.cmu.edu/cgi-bin/cmudict
[4] https://api.yandex.ru/mystem/

Automatic transliteration processes can stumble when dealing with words that derive from languages other than the source or target. In English, for example, the letter *j* usually indicates the affricate sound [dʒ], but in words of Spanish origin, it may represent [h]. This presence of a third-language sound pattern complicates the use of transliteration. Hagiwara and Sekine (2011) and Li et al. (2007) suggest ways to detect alternate languages in statistical transliteration: Li et al. (2007) train with language-tagged word pairs; Hagiwara and Sekine (2011) introduce latent classes to model language origins. For rule-based transliteration, developing programs to detect and correct such third-language spelling differences requires examination of the sound patterns of the various languages; human intervention may be required to decide when to apply the alternate mappings.

Russian provides a particular problem for transliteration due to the presence of third-language sound patterns from Chinese. When referring to Chinese names in Russian texts, Russian writers follow the Palladius mapping (Palladius and Popov, 1888) to transliterate Chinese names into Cyrillic. Many Cyrillic characters generated by this mapping represent different sounds than those Cyrillic characters typically represent in Russian. For example, the Cyrillic character ж typically represents /zh/, but in the Palladius mapping it represents /r/, and the combination of characters Чж is used to represent /zh/. Figure 7 illustrates how applying the typical Russian-to-English transliteration for OOV Russian words will cause errors for Chinese names, unless we first reverse this Palladius mapping (Young et al., 2012).

(a) 翟志□

(b) Чжай Чжиган

(c) Chzhay Chzhigan

(d) Zhai Zhigang

Figure 7: Chinese name (a), with transliterations into Cyrillic (b) and Latin using normal Cyrillic-to-Latin transliteration (c) and reverse Palladius transliteration (d). The output in (d) is correct.

3.4 Under-achieving Translation

We use the phrase *under-achieving translation* to designate weeds that result from a lack of attention by the human translator. Sometimes translators leave a word untranslated; this kind of error can be detected by methods discussed above in §2.1, including the detection of out-of-range characters if the languages have different alphabets. More subtle weeds can occur when the translator chooses transliteration in place of translation, as the appropriateness of transliteration depends on context.

3.4.1 Transliteration in Place of Translation

Sometimes the human translator simply transliterates the source word, even when an appropriate translation exists in the target language. This may represent a translator's decision to preserve the original form in a named entity, or it may reflect a careless translation. For example, the English word *review* has various Russian translations, such as журнал (review, journal) and рецензия (review, critique). However, in the IWSLT 2014 training data we find *review* transliterated in the

phrase, *Harvard Business Review*, Гарвард Бизнес Ревью /garvard biznes rev'ju/. This choice preserves the title of the publication; translating *review* to журнал /ʒurnal/ would have introduced confusion with the English word *journal*. In the Common Crawl, on the other hand, we find an inappropriate transliteration of *review*, in the phrase *Awards and Reviews*, which becomes Награды и Ревью /nagradɨ i rev'ju/. This instance should probably have been translated. For unfamiliar words, a translator may resort to letter-by-letter spelling, as in the Russian spelling опоссум for *opossum*, which reflects the English spelling rather than the pronunciation [pasəm] or [əpasəm]. The coexistence of translation, sound-based transliteration, and letter-based transliteration creates more variation that must be addressed in machine translation.

3.4.2 Code-switching

The use of transliterated foreign words may also be driven by a form of code-switching (Myers-Scotton, 1993; Diab et al., 2014, 2016) in which the writer deliberately uses foreign words. For example, Urdu writers frequently use transliterated English words, instead of their Urdu counterparts, because the use of English exhibits a level of prestige (Upal, 2008). Hence, we may find transliterated English words in Urdu source text, as well as in Urdu text that has been translated from English. In Table 4, the Urdu writer has used English words in transliteration for four words, in place of using the Urdu words. Such transliterations complicate the machine translation of Urdu by creating variations between transliterated English words and the actual Urdu words.

English	In the top ten, India comes in the last
Urdu	اس سرٹیفکیشن کی ٹاپ ٹین میں بھارت آخری نمبر پر ہیں۔
Transliteration	as srṯyfkyšn ky ṯap ṯyn myn bhart Ajry nmbr pr byn
English words	– certification – top ten – – – number – –

Table 4: Urdu transliteration example. In this example, the author of the Urdu sentence used four English words (transliterating certification into srṯyfkyšn, top into ṯap, ten into ṯyn, and number into nmbr) instead of using the corresponding Urdu words.

3.5 Over-achieving Translation (Explicitation)

Human translators intend to communicate meaning, and so may depart from the source text in ways that improve understanding, but degrade the usefulness of the translation as parallel text. Translators may expand acronyms, add explanation of localized vocabulary, or include the actual source-language words. Translators working on informal speech may remove false starts and clean up awkward sentence structure.

We term this type of explicitation (Blum-Kulka, 1986) *over-achieving translation*, in contrast to the *under-achieving translation* of the previous section. This type of extra information is difficult to detect and modify for machine translation. If the translator has set off added material in brackets or parentheses this can be detected, but often the additional material is integrated into the translation.

The TED Talks suffer from particular problems with over-achieving translation, since they are spoken presentations supplemented by visual aids. The English transcriptions tend to follow the

speaker closely, while the translations often clean up disfluency.[5] If text appears on the slides, the translators often include a translation of this material in the transcript.

Similarly, sentence alignment problems can also be caused when human translators summarize (Khadivi and Ney, 2005), engage in one-to-many, many-to-one, or many-to-many sentence translations (Gale and Church, 1991), or engage in non-literal free translations Imamura and Sumita, 2002);[6] the resulting parallel sentences may be less useful from the perspective of machine translation training than other more literal translation pairs. This problem may be mitigated by removing less literal translation pairs from the parallel corpus (Okita, 2009; Jiang et al., 2010), or by flagging sentence pairs which exhibit atypical length ratios for manual inspection (our tools take the latter approach).

3.6 Translation Directionality

Other researchers have noted that translated text differs in crucial ways from native text, in both general simplification (Lembersky et al., 2013) and by influence from the word order and vocabulary choice of the source language text (Fusco, 1990). Koppel and Ordan (2011) show that classifiers can be trained to distinguish the direction of translation. Translation models are typically built from parallel corpora without regard for which language of the pair is the original source language. Changing this paradigm to one where original source language is taken into account has been shown to improve translation quality (Kurokawa et al., 2009).

4 Conclusion

This work is motivated by the lack of a standardized field guide outlining best practices for curating parallel corpora, especially those harvested from the web. Even the most-well curated parallel corpus is likely to contain some problems; even Europarl (Koehn, 2005), arguably the most widely examined parallel corpus, has undergone eight distinct revisions since its release in 2005. In this work, we categorize six major types of problems that originate in automated processing of corpora, as well as six major types of problems that originate in human translator actions. In this work, we establish an initial taxonomy of weeds. While this work is by no means comprehensive of all problems extant in corpus creation, we nevertheless believe that a practical taxonomic field guide, laying out likely pitfalls awaiting corpus curators will represent an important contribution to our community.

The extent to which various types of weeds are harmful in practice is not fully established. Asia Online (2009) and others have claimed substantial positive results from weeding. Likewise, we found substantial improvement in translation quality when major repetition errors are corrected. On the other hand, Goutte et al. (2012) report that statistical MT systems may be robust to sentence alignment errors as high as 30%. In future work we plan a more thorough empirical examination exploring how sensitive various machine translation systems are to various types of weeds.

[5] Cho et al. (2014) suggest handling this issue by tightly integrating disfluency removal into the MT decoding process.

[6] Imamura and Sumita (2002) also identify as problematic to their data-driven rule-based MT technique situations where a given source phrase is translated in multiple different ways throughout the corpus. Modern statistical machine translation techniques tend to be relatively resistant to this variety of weed.

References

Abdelali, A., Guzman, F., Sajjad, H., and Vogel, S. (2014). The amara corpus: Building parallel language resources for the educational domain. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*.

Antonova, A. and Misyurev, A. (2011). Building a web-based parallel corpus and filtering out machine-translated text. In *Proc. Building and Using Comparable Corpora (BUCC'11)*.

Asia Online (2009). Study on the impact of data consolidation and sharing for statistical machine translation. Technical report, TAUS.

Blum-Kulka, S. (1986). Shifts of cohesion and coherence in translation. In House, J. and Blum-Kulka, S., editors, *Interlingual and Intercultural Communication*. Narr, Tübingen.

Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., Soricut, R., Specia, L., and Tamchyna, A. (2014). Findings of the 2014 workshop on statistical machine translation. In *Proc. WMT*.

Callison-Burch, C., Koehn, P., Monz, C., and Schroeder, J. (2009). Findings of the 2009 Workshop on Statistical Machine Translation. In *Proc. WMT*.

Cettolo, M., Girardi, C., and Federico, M. (2012). WIT3: Web inventory of transcribed and translated talks. In *Proc. EAMT'12*, Trento, Italy.

Cho, E., Niehues, J., and Waibel, A. (2014). Tight integration of speech disfluency removal into SMT. In *Proc. EACL'14*, Gothenburg, Sweden.

Corston-Oliver, S., Gamon, M., and Brockett, C. (2001). A machine learning approach to the automatic evaluation of machine translation. In *Proc. ACL'01*, Toulouse, France.

Diab, M., Fung, P., Hirschberg, J., and Solorio, T., editors (2016). *Proceedings of the Second Workshop on Computational Approaches to Code Switching*.

Diab, M., Hirschberg, J., Fung, P., and Solorio, T., editors (2014). *Proceedings of the First Workshop on Computational Approaches to Code Switching*.

Durrani, N., Sajjad, H., Hoang, H., and Koehn, P. (2014). Integrating an unsupervised transliteration model into statistical machine translation. In *Proc. EACL'14*, Gothenburg, Sweden.

Fusco, M. (1990). Quality in conference interpreting between congnate languages: A preliminary approach to the Spanish-Italian case. *Interpreters' Newsletter*, (3).

Gale, W. A. and Church, K. W. (1991). A program for aligning sentences in bilingual corpora. In *Proc. ACL'91*, Berkeley, California.

Gamon, M., Aue, A., and Smets, M. (2005). Sentence-level MT evaluation without reference translations: Beyond language modeling. In *Proc. EAMT'05*, Budapest, Hungary.

Goutte, C., Carpuat, M., and Foster, G. (2012). The impact of sentence alignment errors on phrase-based machine translation performance. In *Proc. AMTA'12*, San Diego, California.

Hagiwara, M. and Sekine, S. (2011). Latent class transliteration based on source language origin. In *Proc. ACL'11*, Portland, Oregon, USA.

Hellstern, A. and Marciano, J. (2014). Two sides of a coin: Machine translation and post-editing projects from the perspectives of the client and language services provider. Proc. ATA'14.

Imamura, K. and Sumita, E. (2002). Bilingual corpus cleaning focusing on translation literality. In *Proc. INTERSPEECH'02*, Denver, Colorado.

Jiang, J., Way, A., and Carson-Berndsen, J. (2010). Lattice score based data cleaning for phrasebased statistical machine translation. In *Proc. EAMT'10*, Saint-Raphaël, France.

Khadivi, S. and Ney, H. (2005). Automatic filtering of bilingual corpora for statistical machine translation. In *Proc. NLDB'05*, volume 3513 of *Lecture Notes in Computer Science*.

Koehn, P. (2005). Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proc. MT Summit X*, Phuket, Thailand.

Koppel, M. and Ordan, N. (2011). Translationese and its dialects. In *Proc. ACL'11*, Portland, Oregon.

Kurokawa, D., Goutte, C., and Isabelle, P. (2009). Automatic detection of translated text and its impact on machine translation. In *Proc. MT Summit XII*, Ontario, Canada.

Lembersky, G., Ordan, N., and Wintner, S. (2013). Improving statistical machine translation by adapting translation models to translationese. *Computational Linguistics*, 39(4).

Li, H., Sim, K. C., Kuo, J.-S., and Dong, M. (2007). Semantic transliteration of personal names. In *Proc. ACL'07*, Prague, Czech Republic.

Lui, M., Letcher, N., Adams, O., Duong, L., Cook, P., and Baldwin, T. (2014). Exploring methods and resources for discriminating similar languages. In *Proc. Workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects*, pages 129–138.

Matthew (1st century). Gospel of Matthew. Greek New Testament.

Mermer, C., Kaya, H., and Doğan, M. U. (2007). The TÜBİTAK-UEKAE statistical machine translation system for IWSLT 2007. In *Proc. IWSLT'07*, Trento, Italy.

Myers-Scotton, C. (1993). *Social Motivations for Codeswitching: Evidence from Africa*. Oxford Studies in Language Contact.

Notenbloom, L. (2009). Why do i get odd characters instead of quotes in my documents? http://askleo.com.

Okita, T. (2009). Data cleaning for word alignment. In *Proc. ACL-IJCNLP'09*, Singapore.

Palladius and Popov, P. S. (1888). *Китайско-русский словарь (Chinese-Russian Dictionary).* Beijing, China. https://archive.org/details/11888.

Pellom, B. and Hacioglu, K. (2001). Sonic: The University of Colorado continuous speech recognizer. Technical Report TR-CSLR-2001-01, University of Colorado, Boulder, Colorado.

Plitt, M. and Masselot, F. (2010). A productivity test of statistical machine translation postediting in a typical localisation context. *Prague Bulletin of Mathematical Linguistics*, 93.

Rarrick, S., Quirk, C., and Lewis, W. (2011). MT detection in web-scraped parallel corpora. In *Proceedings of the 13th Machine Translation Summit (MT Summit XIII)*, Xiamen, China.

Resnik, P. (1998). Parallel strands: A preliminary investigation into mining the web for bilingual text. In *Proc. AMTA'98*, volume 1529 of *Lecture Notes in Artificial Intelligence*. Springer.

Schwartz, L., Anderson, T., Gwinnup, J., and Young, K. (2014). Machine translation and monolingual postediting: The AFRL WMT-14 system. In *Proc. WMT'14*, Baltimore, Maryland.

Simard, M. (2014). Clean data for training statistical MT: The case of MT contamination. In *Proc. AMTA '14*, Vancouver, Canada.

Simard, M., Foster, G. F., and Isabelle, P. (1992). Using cognates to align sentences in bilingual corpora. In *Proc. Theoretical and Methodological Issues in Machine Translation (TMI'92)*, Montréal, Canada.

Smith, J. R., Saint-Amand, H., Plamada, M., Koehn, P., Callison-Burch, C., and Lopez, A. (2013). Dirt cheap web-scale parallel text from the common crawl. In *Proc. ACL'13*, Sofia, Bulgaria.

Upal, M. A. (2008). Personal correspondence.

Venugopal, A., Uszkoreit, J., Talbot, D., Och, F., and Ganitkevitch, J. (2011). Watermarking the outputs of structured prediction with an application in statistical machine translation. In *Proc. EMNLP'11*, Edinburgh, Scotland.

Wu, D. (1994). Aligning a parallel English-Chinese corpus statistically with lexical criteria. In *Proc. ACL'94*, Las Cruces, New Mexico.

Young, K. M., Gwinnup, J., and Reinhart, J. (2012). Reversing the Palladius mapping of Chinese names in Russian text. In *Proc. AMTA'12*, San Diego, California.

Zampieri, M., Tan, L., Ljubešić, N., and Tiedemann, J. (2014). A Report on the DSL Shared Task 2014. In *Proc. Workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects*, pages 58–67, Dublin, Ireland.

Zukerman, E. (2013). Review: Amara is a web-based service that lets anyone transcribe and translate online video. http://www.pcworld.com/article/2032787/review-amara-is-a-webbased- service-that-lets-anyone-transcribe-and-translate-online-video.html.

APPENDIX C: Qahira: A Word Alignment Viewer and Editor

Qahira: A Word Alignment Viewer and Editor

Jeremy Gwinnup

SRA International[†]
5000 Springfield Street, Suite 200
Dayton, OH 45431

jeremy.gwinnup.ctr@us.af.mil

Abstract

We present Qahira, an updated version of the Cairo Word Alignment Tool developed at the 1999 JHU summer workshop. Qahira improves on Cairo by adding editing capabili- ties and better display of non-Latin scripts. These improve- ments allow users to verify and annotate alignments, correct alignments and word ordering, and sort sentences by alignment score to identify alignment errors.

1. Introduction

The Cairo Word Alignment Tool [1] was developed at the 1999 JHU Workshop [2] as a means to visualize word align- ments for use in machine translation systems. Unfortunately, it has not seen significant updates since it was published. One issue Cairo had was difficulty in displaying non-Latin char- acters in various file encodings. In order to use Cairo with these languages, significant portions of the code were up- dated. During the updating process, the name of the tool was changed to Qahira (short for al-Q¯ahirah), the modern name of Cairo.

Statistical models of relationships between words can be computed and applied to many natural language processing problems. These alignment models are often used in word-based and phrase-based machine translation systems. Brown et. al. [3] describe a variety of statistical methods known as the IBM Models. GIZA++ [4] implements Models 1-5 in ad- dition to the Hidden Markov Alignment Model [5]. Despite development of many other word aligners, GIZA++ and its multi-threaded variant MGIZA [6] still provide word alignments for many statistical machine translation systems today.

Over the years, many efforts were made to improve the original IBM models. Fast Align[7] and Fast Align NG [8] improve on Model 2 by simplifying the parameter optimization process. RegAligner [9] corrects deficiencies in Mod- els 3 and 4 via parameter normalization and Giza-Sharp [10] address deficiencies in Model 4 by recasting them in the Baysian framework. PostCAT [11, 12] takes a different ap- proach by adding additional posterior constraints to the IBM models to improve alignment quality. GIZA++-kn [13] add

smoothing of fractional counts to GIZA++.

The IBM Models and their descendants are statisti- cally generated word alignment models. Heuristic based approaches such as the Competitive Linking Algorithm (CLA)[14, 15] create alignments by using a greedy algo- rithm to compute one-to-one word alignments. Och and Ney [4] show that word alignment models with a first-order dependence and a fertility model outperform models based on a simple heuristic, but [16] show that models from both GIZA++ and CLA can be combined to improve phrase-based machine translation system performance.

When translating language pairs where one language is morphologically rich and the other language is not, word alignment can be difficult due to the variety of word inflections in the morphologically rich language. Creating word alignments from morphemes can help as shown in [17] [18].

The IBM Models create asymmetric alignments with many source words linking to a target word but not the reverse. Machine translation systems typically train alignment models for both source-to-target and target-to-source, then symmetrize the resulting alignments to improve alignment quality [19].The Berkeley Aligner [20, 21] trains the two alignment models jointly in order to reduce word alignment errors. SymGIZA++ [22] modifies GIZA++ to operate bidirectionally to achieve the same goal.

Denero and Klein [23] describe an alternative to phrase extraction by proposing a supervised joint word alignment and phrase extraction process. PIAlign [24] implements this process in an unsupervised manner.

The Nile word aligner [25, 26] is used to demonstrate that training a discriminative alignment model with a simple hierarchical search using a small amount of human-edited data improves alignment accuracy.

BIA [27] implements a phrase-based word alignment decoder that integrates tightly into a phrase-based machine

[†] This work is sponsored by the Air Force Research Laboratory under Air Force contract FA-8650-09-D-6939-029.

translation system allowing for better tuning of word alignment metrics useful to the machine translation system.

2. RelatedWork

The original Cairo tool was implemented in order to view the alignments and parameters of the IBM models, specifically Model 3. This model utilizes four types of parameters: translation probability, fertility, distortion and NULL-insertion. Cairo provides a means to visualize the word alignments as a series of lines connecting words in the source and target sentences. It also allows for the display of a reference translation. Multiple sets of alignments for a given sentence pair are each given their own line colors for differentiation.

I-Link [28, 29] combines an unsupervised word aligner with a human-in-the-loop process to incrementally improve the quality of word alignments. The user interface provides a means to align words and provide morphological information for those links. Working in batches of sentences, the user can improve the quality of alignments by correcting errors in the automated results. The unsupervised portion of the process then continues with the addition of knowledge from the userannotated alignments.

Alpaco [30] allows viewing and editing of word and phrase alignments in a graphical user interface.

The UMAICSWord Alignment Interface [31] has an emphasis on creating alignments from scratch. It has a similar alignment interface in that the source and target sentence are displayed and the user is able to create word alignment links between the two sentences.

HandAlign [32] takes a similar approach to the UMAICS Word Alignment Interface in presenting an interface to draw word alignment links between a source and target sentence. This tool differs in that it displays portions of documents for the current source and target sentence to provide additional context when creating word alignments. HandAlign also provides the ability to specify phrase alignments.

Callison-Burch et. al [33] uses a web-based grid display that allows users to create alignments that are then combined with alignments from GIZA++.

ICA [34] is an web-based tool developed as part of UPLUG framework. This tool uses clues supplied from various UPLUG components to help automatically determine links between both individual words and phrases. These automatic alignments can then be viewed in the tool.

Yawat (Yet AnotherWord Alignment Tool) [35] is a webbased tool that presents a series of parallel sentences in a web page table that can be clicked on for further investigation. Clicking an icon attached to the sentence pair brings up a grid display showing the word alignments where the source sentence words are displayed on one axis and the target sentence words are displayed on the other. Word alignment cells can be selected to edit alignments. Hovering over grid cells will highlight words in the source and target sentence display. Kwipc (Key Words In Parallel Context) [36] extends Yawat's display with the ability to use regular expressions to search for specific sentence pairs.

The LDCWord Aligner [37] was developed to both annotate and word align sentence pairs. The user is presented with a display of a document pair with the current sentence highlighted. Additional sentences are displayed in order to provide additional context for alignment and annotation. Words from the current sentence are displayed in two columns allowing links to be made between matching words. The user can also mark words with various notes, such as link correctness or incorrectness, typographic or tokenization errors and other useful information.

Picaro [38] is a command-line based tool to display word alignments in a grid layout in a terminal. This tool is intended as a quick diagnostic check to inspect results of word alignment.

Qahira is based on the original Cairo implementation and added the ability to directly read the word alignments output by GIZA++. The code that handles display of translation probability, fertility, and distortion metrics was removed. The ability to edit alignments and additional support for display of non-Latin script languages were also added. Additional details and capabilities are outlined in section 3.

3. Qahira

Qahira is implemented as a Java application using the SWT interface toolkit[1]. SWT's goal is to provide access to the host operating system's user interface capabilities while still running as portable code. This allows Qahira to take advantage of internationalization capabilities present in modern operating systems, especially in the area displaying non-Latin characters.

The user interface, shown in Figure 1, consists of a word alignment panel which displays the source and target sentences as parallel rows with the words arranged in text boxes from left to right. When aligning words where one language is in a right-to-left script, the letters comprising each word will be shown correctly but the sentence as a whole will read left-to-right. The two large text boxes immediately below the word alignment panel show the sentences in their entirety to aid the user in viewing the sentence in the correct format. If a sentence is in a right-to-left script, it will display right-to-left as expected.

Below the alignment and sentence display is a table displaying the set of sentence pairs that have been loaded. The user can click on a sentence pair entry to view it in the alignment panel. Additional information such as score, sentence id, and edit status are also displayed. The currently selected sentence and current alignment file are shown in the application's status bar.

[1] http://www.eclipse.org/swt

3.1. Starting Qahira

Qahira is packaged as a standalone jar file (or app bundle on OS X) and can be started in one of two ways. First, a user can simply click on the jar or package in the file viewer, which brings up the user interface as shown in Figure 1. Second, the user can issue the following command at the prompt to achieve the same result:

```
java -jar qahira.jar
```

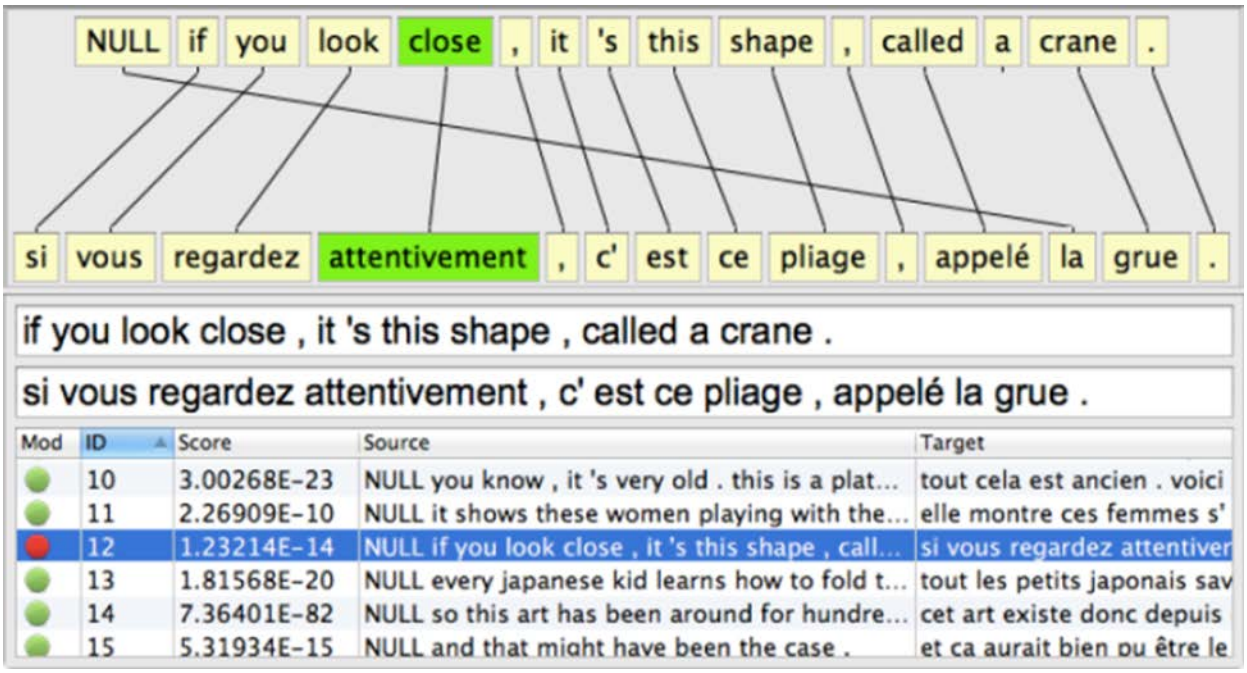


Figure 1: The Qahira user interface

Once the application is open, the user can then open an alignment file by selecting 'Open A3' or 'Open Advanced' to open a set of alignments in a variety of formats.

3.2. Editing Links

Users can edit word alignments in Qahira by clicking on either a source or target sentence word, then clicking a word in the opposite sentence to create a link. Multiple links can be made by clicking successive words. Links can be deleted by clicking on the two linked words in succession. There is a menu option that will allow a user to delete all alignment links if a sentence is so poorly aligned that a fresh start is needed.

3.3. Reordering Words in Sentences

Some alignment editing tasks may require the ability to fix word order errors in source or target sentences. Selecting 'Word Drag' from the 'Options' menu allows the user to reorder words in sentences by clicking and dragging a word to the correct position. Any attached links are preserved during the repositioning process. The ability to reorder words can be disabled if the user needs to ensure the original word order is preserved.

3.4. Glossaries for Translation

It can be useful to display possible translations for words if the user does not speak the source or target language. Using a simple glossary in LDC's LCTL Lexicon format[2] allows the display of a tooltip containing possible simple translations with part-of-speech to aid the user in determining whether an alignment link is correct. Figure 2 shows a Chinese-English sentence pair with a tooltip appearing over the selected word with a pinyin pronunciation as well as two possible glossary translations. In this case, the glossary entry confirms that the Chinese word is correctly aligned to the word 'health'.

3.5. Auditing and ClassifyingWord Alignments

Qahira provides the ability to audit word alignments in multiple ways. In the first way, the user can select the 'Verify' mode from the application's 'Mode' menu. This allows the user to click on the links between words to cycle between 'Good' alignments displayed in green, 'Bad' alignments displayed in red, or alignments that should not exist in gray. Depending on the nature of the auditing task, users may be asked to either mark or remove bad links. Removing bad links improves the the quality of the word alignments. Marking bad links allows review of the alignment algorithm to diagnose bad alignment behaviors.

[2] https://www.ldc.upenn.edu/collaborations/past-projects

Alternately, the user can select ‘Sure/Probable’ from the mode menu to rank alignments as described in [39]. In this mode, ‘Sure’ links are shown in green and ‘Probable’ links are shown in blue as demonstrated in Figure 3. An additional click will mark that link for deletion when the alignments are

Figure 2: Displaying Chinese-English Alignments with Glossary Tooltip

saved before moving to the next sentence pair.

Annotating or editing alignments can get tedious after an extended session. Qahira keeps track of edited sentence pairs by marking a colored ball in the first column of the sentence pair table as shown in Figure 1. When the user edits or annotates the sentence pair in the current session, the ball will change from green to red. If a user quits the program and resumes work later, the saved A3 file can be loaded and previously edited entries will be denoted with an orange ball.

An experiment was performed on the NIST OpenMT 2009 Urdu-English data set where the initial poor alignments were selected from the output of GIZA++ and handcorrected for the forward-only alignments as outlined in subsection 3.6. Approximately 8% of the worst alignments were improved but only yielded a slight improvement in BLEU [40] score over an unsupervised-only word aligner baseline. It would be useful to perform this correction process on the symmetrized alignments to better examine the impact of post-editing the word alignments as shown in [33].

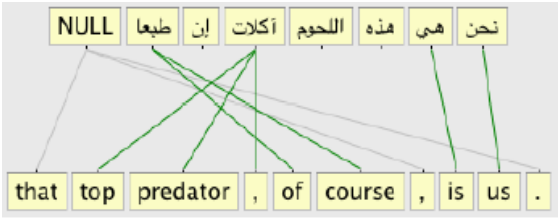


Figure 3: Displaying Arabic-English alignments with Sure/Probable link designations

3.6. Sorting Alignments for Triage

Sentences can be sorted by alignment score by clicking the 'Score' column header as shown in Figure 1. Sorting by alignment score allows users to easily identify poorly aligned sentences. These poorly aligned sentences can then be analyzed or corrected.

4. FutureWork

We intend to integrate a lightweight word aligner such as the Java implementation[3] of FastAlign [7] to generate an initial set of word alignments. These alignments could then be edited by annotators, allowing Qahira to unction independently of a machine translation system.

Additionally, we intend to add support for other glossary formats to provide additional word meaning cues.

Lastly, we intend to pursue performance and functionality improvements to increase productivity of the tool.

5. Acknowledgements

The author would like to thank Katherine Young for her years of testing and suggestions for improvement for both Qahira and this paper. The author would also like to thank Lane Schwartz for the insightful comments and feedback.

6. References

[1] N. A. Smith and M. E. Jahr, "Cairo: An alignment visualization tool," in *Proceedings of the Language Resources and Evaluation Conference (LREC 2000)*, Athens, Greece, May/June 2000, pp. 549–552.

[2] Y. Al-Onaizan, J. Curin, M. Jahr, K. Knight, J. Lafferty, D. Melamed, F.-J. Och, D. Purdy, N. A. Smith, and D. Yarowsky, "Statistical machine translation," Final Report, JHU Summer Workshop, Tech. Rep., 1999.

[3] P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, and R. L. Mercer, "The mathematics of statistical machine translation: Parameter estimation," vol. 19, no. 2. Stroudsburg, PA, USA: MIT Press, June 1993, pp. 263–311. [Online]. Available: http://dl.acm.org/citation.cfm?id=972470.972474

[4] F. J. Och and H. Ney, "A systematic comparison of various statistical alignment models," *Computational Linguistics*, vol. 29, no. 1, pp. 19–51, 2003.

[5] S. Vogel, H. Ney, and C. Tillmann, "Hmm-based word alignment in statistical translation," in COLING 1996

[3] https://github.com/dowobeha/fast_align.java

Opinions, interpretations, conclusions and recommendations are those of the author and not necessarily endorsed by the United States Government. Cleared for Public Release on 23 Sep 2014. Originator reference number RH-14-112965. Case number 88ABW-2014-4499.

Volume 2: *The 16th International Conference on Computational Linguistics*, 1996, pp. 836–841. [Online]. Available: http://www.aclweb.org/anthology/ C96-2141

[6] Q. Gao and S. Vogel, "Parallel implementations of word alignment tool," in *Software Engineering, Testing and Quality Assurance for Natural Language Processing*, Columbus, Ohio, June 2008, pp. 49–57.

[7] C. Dyer, V. Chahuneau, and N. A. Smith, "A simple, fast, and effective reparameterization of IBM model 2," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, June 2013, pp. 644–648. [Online]. Available: http://www.aclweb.org/anthology/ N13-1073

[8] D. Gelling and T. Cohn, "Simple extensions and pos tags for a reparameterised ibm model 2," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Baltimore, Maryland: Association for Computational Linguistics, June 2014, pp. 150–154. [Online]. Available: http://www.aclweb.org/anthology/ P/P14/P14-2025

[9] T. Schoenemann, "Training nondeficient variants of ibm-3 and ibm-4 for word alignment," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, August 2013, pp. 22–31. [Online]. Available: http://www.aclweb.org/anthology/P13-1003

[10] Y. Gal and P. Blunsom, "A systematic bayesian treatment of the ibm alignment models," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, June 2013, pp. 969–977. [Online]. Available: http://www.aclweb.org/anthology/N13-1117

[11] K. Ganchev, J. V. Graca, and B. Taskar, "Better alignments = better translations?" in *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics*, Columbus, Ohio, June 2008.

[12] J. V. Graca, K. Ganchev, and B. Taskar, "Expectation maximization and posterior constraints," in *Proceedings of 20th Workshop on Advances in Neural Information Processing Systems*, 2008.

[13] H. Zhang and D. Chiang, "Kneser-ney smoothing on expected counts," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland: Association for Computational Linguistics, June 2014, pp. 765–774. [Online]. Available: http://www.aclweb.org/anthology/P14-1072

[14] I. D. Melamed, "Models of translational equivalence among words," *Computational Linguistics*, vol. 26, no. 2, pp. 221–249, 2000.

[15] B. Chen, R. Cattoni, N. Bertoldi, M. Cettolo, and M. Federico, "The ITC-irst SMT system for IWSLT- 2005," in *Proceedings of the 5th International Workshop on Spoken Language Translation*, 2005.

[16] B. Chen and M. Federico, "Improving phrase-based statistical translation through combination of word alignments," *FinTAL 2006*, pp. 356–367, 2006.

[17] E. Eyig¨oz, D. Gildea, and K. Oflazer, "Simultaneous word-morpheme alignment for statistical machine translation," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, June 2013, pp. 32–40. [Online]. Available: http://www.aclweb.org/anthology/ N13-1004

[18] ——, "Multi-Rate HMMs for word alignment," in *Proceedings of the Eighth Workshop on Statistical Machine Translation*. Sofia, Bulgaria: Association for Computational Linguistics, August 2013, pp. 494–502. [Online]. Available: http://www.aclweb.org/anthology/ W13-2262

[19] P. Koehn, F. J. Och, and D. Marcu, "Statistical phrase-based translation," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1, ser*. NAACL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 48–54. [Online]. Available: http://dx.doi.org/10.3115/1073445.1073462

[20] P. Liang, B. Taskar, and D. Klein, "Alignment by agreement," in *Proceedings of the 2006 Conference of the North American Chapter of the Association for Computational Linguistics*, 2006.

[21] J. DeNero and D. Klein, "Tailoring word alignments to syntactic machine translation," in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 2007.

[22] M. Junczys-Dowmunt and A. Sza, "Symgiza++: Symmetrized word alignment models for machine translation," in *Security and Intelligent Information Systems (SIIS), ser*. Lecture Notes in Computer Science, P. Bouvry, M. A. Klopotek, F. Leprvost, M. Marciniak, A. Mykowiecka, and H. Rybinski, Eds., vol. 7053. Warsaw,

Poland: Springer, 2012, pp. 379–390. [Online]. Available: http://emjotde.github.io/ publications/pdf/mjd2011siis.pdf

[23] J. DeNero and D. Klein, "Discriminative modeling of extraction sets for machine translation," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden: Association for Computational Linguistics, July 2010, pp. 1453–1463. [Online]. Available: http://www.aclweb.org/anthology/P10-1147

[24] G. Neubig, T. Watanabe, E. Sumita, S. Mori, and T. Kawahara, "An unsupervised model for joint phrase alignment and extraction," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*, Portland, Oregon, June 2011.

[25] J. Riesa, A. Irvine, and D. Marcu, "Feature-rich language-independent syntax-based alignment for statistical machine translation," *in Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, July 2011, pp. 497–507.

[26] J. Riesa and D. Marcu, "Hierarchical search for word alignment," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 2010, pp. 157–166.

[27] P. Lambert and R. E. Banchs, "Bia: a discriminative phrase alignment toolkit," *The Prague Bulletin of Mathematical Linguistics (PBML)*, no. 97, pp. 43–53, 2012. [Online]. Available: http://ufal.mff.cuni.cz/pbml/97/art-lambert-banchs.pdf

[28] L. Ahrenberg, M. Andersson, and M. Merkel, "A system for incremental and interactive word linking," in In *Proceedings of the Third International Conference on Language Resources and Evaluation* (LREC, 2002, pp. 485–490.

[29] L. Ahrenberg, M. Merkel, and M. Petterstedt, "Interactive word alignment for language engineering," in *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics - Volume 2*, ser. EACL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 49–52. [Online]. Available: http://dx.doi.org/10.3115/1067737.1067746

[30] T. Pederson, "Alpaco," http://www.d.umn.edu/tpederse/parallel.html, 2003.

[31] R. Hwa and N. Madnani, "The UMIACS word aligner interface," http://www.umiacs.umd.edu/nmadnani/alignment, 2004.

[32] H. Daume, "HandAlign," http://www.umiacs.umd.edu/hal/HandAlign, 2004.

[33] C. Callison-Burch, D. Talbot, and M. Osborne, "Statistical machine translation with word- and sentence-aligned parallel corpora," in *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, Barcelona, Spain, July 2004, pp. 175–182. [Online]. Available: http://www.aclweb.org/anthology/P04-1023

[34] J. Tiedemann, "ISA & ICA two web interfaces for interactive alignment of bitext," in In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*, 2006.

[35] U. Germann, "YAWAT: Yet another word alignment tool," in *Proceedings of the ACL-08: HLT Demo Session (Companion Volume)*, Columbus, Ohio, June 2008, pp. 20–23.

[36] ——, "Two tools for creating and visualizing subsentential alignments of parallel text," in *Proceedings of the Linguistic Annotation Workshop*. Association for Computational Linguistics, 2007, pp. 121–124.

[37] Linguistic Data Consortium, "LDC word aligner," https://www.ldc.upenn.edu/language-resources/tools/ ldc-word-aligner, 2009.

[38] J. Riesa, "Picaro: A simple word alignment visualization tool." http://code.google.com/p/picaro, 2012.

[39] R. Mihalcea and T. Pedersen, "An evaluation exercise for word alignment," in *HLT-NAACL 2003 Workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, R. Mihalcea and T. Pedersen, Eds. Edmonton, Alberta, Canada: Association for Computational Linguistics, May 2003, pp. 1–10.

[40] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (ACL '02), Philadelphia, Pennsylvania, July 2002, pp. 311– 318.

APPENDIX D: User's Guide to Experiment Reader

Experiment Reader

Introduction

The Experiment Reader web application was developed to help sort through the enormous amount of scoring data created during machine translation. There needed to be way to view and sort the scores, statistics, dates and configuration files generated with each translation process.

The scoring data is saved within stats files during the MT process. Scripts were created to read through selected directories and parse out the scoring data. This information was then saved out to a MySQL database so that the scores could be individually displayed and sorted on the front end.

Front End GUI

Upon first bringing up the Experiment Reader you will have the options of going to the Admin Page, the iBLEU Scorer, selecting the Score View and choosing your directory of scores.

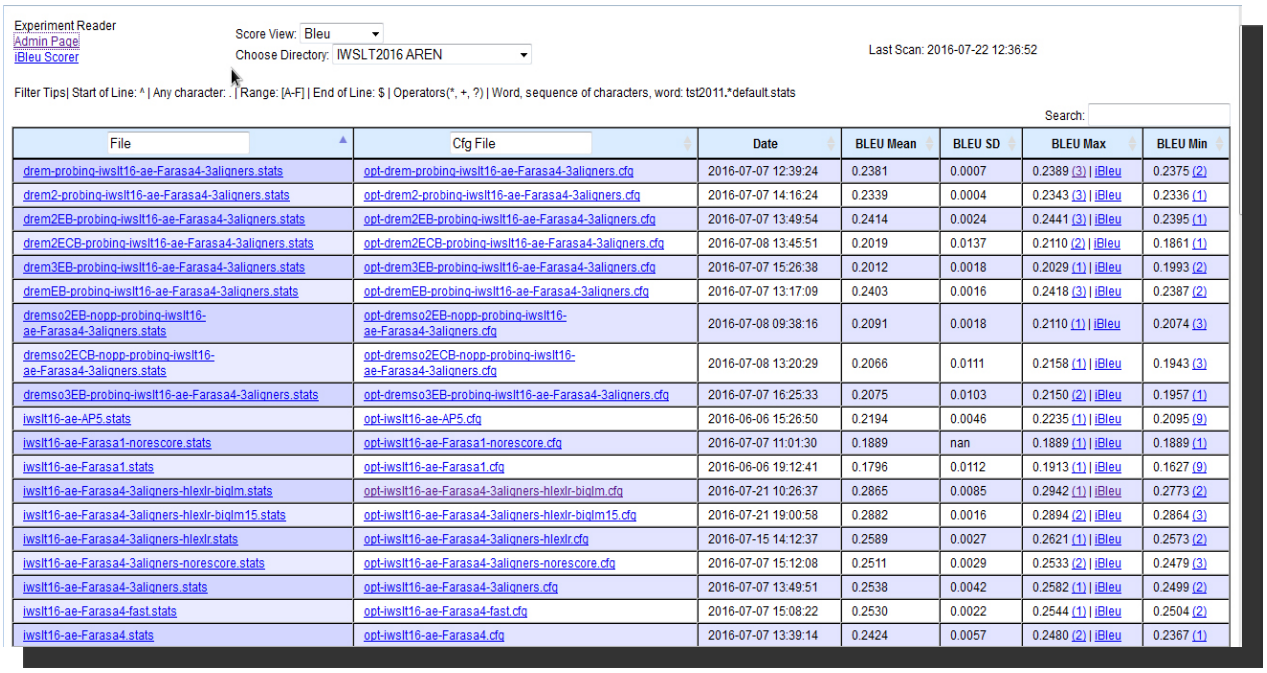
A basic front end was initially developed to control how the data could be displayed and sorted. Each configuration and stats file displayed is also a hard link to the actual file so that the user can bring up the complete file for viewing.

Over time, as the interface was used and the data sets grew larger, new requests were made for sorting and displaying the information. For this next phase of development, Jquery was adopted to help in the management of the front end. JQuery is a free, open source JavaScript library for dynamic update and control of web pages incorporating various features of client-side scripting.

The DataTables Jquery plugin is used to manage the database output as a dynamic, editable table. Each category (directory, file, cfg file and numerous scores) may be clicked on for sorting the data. Directories and various score categories could be selected through drop-down selectors to allow the user full control of the scores they would want to review.

At any time a user can select which directory can be rescanned for new scoring data and export it all to a comma-separated file for viewing offline or to integrate into other data manipulation software.

It was also necessary that other peripheral information be available and easily accessible. If a user was browsing the scores and needed more specific information on which configuration file was used and its contents they need only click on the Cfg file.



Experiment Reader
Admin Page
iBleu Scorer

Score View: Bleu
Choose Directory: IWSLT2016 AREN

Last Scan: 2016-07-22 12:36:52

Filter Tips| Start of Line: ^ | Any character: . | Range: [A-F] | End of Line: $ | Operators(*, +, ?) | Word, sequence of characters, word: tst2011.*default.stats

Search:

File	Cfg File	Date	BLEU Mean	BLEU SD	BLEU Max	BLEU Min
drem-probing-iwslt16-ae-Farasa4-3aligners.stats	opt-drem-probing-iwslt16-ae-Farasa4-3aligners.cfg	2016-07-07 12:39:24	0.2381	0.0007	0.2389 (3) | iBleu	0.2375 (2)
drem2-probing-iwslt16-ae-Farasa4-3aligners.stats	opt-drem2-probing-iwslt16-ae-Farasa4-3aligners.cfg	2016-07-07 14:16:24	0.2339	0.0004	0.2343 (3) | iBleu	0.2336 (1)
drem2EB-probing-iwslt16-ae-Farasa4-3aligners.stats	opt-drem2EB-probing-iwslt16-ae-Farasa4-3aligners.cfg	2016-07-07 13:49:54	0.2414	0.0024	0.2441 (3) | iBleu	0.2395 (1)
drem2ECB-probing-iwslt16-ae-Farasa4-3aligners.stats	opt-drem2ECB-probing-iwslt16-ae-Farasa4-3aligners.cfg	2016-07-08 13:45:51	0.2019	0.0137	0.2110 (2) | iBleu	0.1861 (1)
drem3EB-probing-iwslt16-ae-Farasa4-3aligners.stats	opt-drem3EB-probing-iwslt16-ae-Farasa4-3aligners.cfg	2016-07-07 15:26:38	0.2012	0.0018	0.2029 (1) | iBleu	0.1993 (2)
dremEB-probing-iwslt16-ae-Farasa4-3aligners.stats	opt-dremEB-probing-iwslt16-ae-Farasa4-3aligners.cfg	2016-07-07 13:17:09	0.2403	0.0016	0.2418 (3) | iBleu	0.2387 (2)
dremso2EB-nopp-probing-iwslt16-ae-Farasa4-3aligners.stats	opt-dremso2EB-nopp-probing-iwslt16-ae-Farasa4-3aligners.cfg	2016-07-08 09:38:16	0.2091	0.0018	0.2110 (1) | iBleu	0.2074 (3)
dremso2ECB-nopp-probing-iwslt16-ae-Farasa4-3aligners.stats	opt-dremso2ECB-nopp-probing-iwslt16-ae-Farasa4-3aligners.cfg	2016-07-08 13:20:29	0.2066	0.0111	0.2158 (1) | iBleu	0.1943 (3)
dremso3EB-probing-iwslt16-ae-Farasa4-3aligners.stats	opt-dremso3EB-probing-iwslt16-ae-Farasa4-3aligners.cfg	2016-07-07 16:25:33	0.2075	0.0103	0.2150 (2) | iBleu	0.1957 (1)
iwslt16-ae-AP5.stats	opt-iwslt16-ae-AP5.cfg	2016-06-06 15:26:50	0.2194	0.0046	0.2235 (1) | iBleu	0.2095 (9)
iwslt16-ae-Farasa1-norescore.stats	opt-iwslt16-ae-Farasa1-norescore.cfg	2016-07-07 11:01:30	0.1889	nan	0.1889 (1) | iBleu	0.1889 (1)
iwslt16-ae-Farasa1.stats	opt-iwslt16-ae-Farasa1.cfg	2016-06-06 19:12:41	0.1796	0.0112	0.1913 (1) | iBleu	0.1627 (9)
iwslt16-ae-Farasa4-3aligners-hlexlr-biglm.stats	opt-iwslt16-ae-Farasa4-3aligners-hlexlr-biglm.cfg	2016-07-21 10:26:37	0.2865	0.0085	0.2942 (1) | iBleu	0.2773 (2)
iwslt16-ae-Farasa4-3aligners-hlexlr-biglm15.stats	opt-iwslt16-ae-Farasa4-3aligners-hlexlr-biglm15.cfg	2016-07-21 19:00:58	0.2882	0.0016	0.2894 (2) | iBleu	0.2864 (3)
iwslt16-ae-Farasa4-3aligners-hlexlr.stats	opt-iwslt16-ae-Farasa4-3aligners-hlexlr.cfg	2016-07-15 14:12:37	0.2589	0.0027	0.2621 (1) | iBleu	0.2573 (2)
iwslt16-ae-Farasa4-3aligners-norescore.stats	opt-iwslt16-ae-Farasa4-3aligners-norescore.cfg	2016-07-07 15:12:08	0.2511	0.0029	0.2533 (2) | iBleu	0.2479 (3)
iwslt16-ae-Farasa4-3aligners.stats	opt-iwslt16-ae-Farasa4-3aligners.cfg	2016-07-07 13:49:51	0.2538	0.0042	0.2582 (1) | iBleu	0.2499 (2)
iwslt16-ae-Farasa4-fast.stats	opt-iwslt16-ae-Farasa4-fast.cfg	2016-07-07 15:08:22	0.2530	0.0022	0.2544 (1) | iBleu	0.2504 (2)
iwslt16-ae-Farasa4.stats	opt-iwslt16-ae-Farasa4.cfg	2016-07-07 13:39:14	0.2424	0.0057	0.2480 (2) | iBleu	0.2367 (1)

Search & Filter

An often requested update to the front end was the ability to search and filter information so that the display was easier to manage and read. Search functionality was also added to enable filtered results from the overall score sets or by a specific column such as the file name or config file.

Adding to the search functionality was a means to lock down the initial sort value with the SHIFT key allowing a secondary parameter for sorting, so that one could lock onto the file with the highest Bleu Score then sort by the Meteor Score and see if similar configurations rise to the top.

IBLEU Scorer

iBleu (http://desilinguist.org) is a JavaScript-based tool created by Nitin Madnani to examine the output from statistical machine translation and give it a Bleu score down to the segment level.

With the SCREAM Lab's closed network, the iBleu code was edited to allow for translation requests to be sent to the Lab's own copy of Systran for translation comparison.

Work has progressed on linking to iBleu directly from the score sets on the main page.

Administrative Tools

The Admin Page is a single file (admin.php) consisting of 4 sections. 2 of the sections are brought in through frames and the other 2 are dynamic lists.

Directory Activation: This section is a frame (dir_admin.php) for activating and deactivating the directory paths that have been scanned and displayed on the front end. On the display end is a

list of all directories with a radio button option to activate/deactivate, triggering an update SQL query to toggle activation based on dir_id.

Create Path Directory: This section (test_network.php) is where a new directory path can be added for inclusion into the Experiment Reader. This uses a function to drill down through network pathways to the desired directory and perform a scan of the files. If a specific directory cannot be found in this tool, it means that the directory (or its source) has not had FSMount applied to them. A System Administrator will need to perform this action before the directory will show in the system. Once a directory is selected for path creation the results are added to the exp_dir table and scan of the direcrtory should be performed.

Scan/Update Directory: This section is a dynamic list of active directories that links off to dir_scanner.php and using the dir_id, rescans the directory and updates the scores in the database.

Create CSV: This section is a dynamic list of active directories that link off to maskeCSV.php and using the dir_id will parse the current scores into a comma-separated-value file for portability into other programs.

Structure

The Experiment Reader resides on WWW/htdocs/expreader.

Direcory structure:

expreader/ – all main files reside in the root

- css/ – all style files
- ibleu-2.6.2/ – the iBleu application
- jquery/ – Jquery functionality for display
- js/ – validation code

MySQL:

Server: Www

Database: expreader

Username: expreader

Password: expreader

Tables: exp_data (indexed scoring and file-relevant information

exp_dir (relevant direcotry paths, scan names and dates)

APPENDIX E: Submodularity for Speech and Language Applications

Final Report

Submodularity for Speech and Language Applications

Prepared for

Air Force Research Laboratory

Contract Number: FA8650-09-D-6939

Submitted by

Katrin Kirchhoff

University of Washington

January 4, 2016

REPORT DOCUMENTATION PAGE	*Form Approved* **OMB No. 0704-0188**

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
11/30/2015	Final	09/01/2014-11/30/2015

4. TITLE AND SUBTITLE
Submodularity for Speech and Language Applications

5a. CONTRACT NUMBER
FA8650-12-2-7263

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Dr. Katrin Kirchhoff
Dept. of Electrical Engineering
University of Washington
Box 352500
Seattle, WA, 98195

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
SRA International, Inc.
SRA C4ISR Center
5000 Springfield Street, Suite 200
Dayton, Ohio 45431

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory
711 Human Performance Wing
Airman Systems Directorate
2255 H Street
Wright-Patterson AFB, Ohio 45433

10.SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RHXB

11.SPONSORING/MONITORING AGENCY REPORT NUMBER

12. DISTRIBUTION AVAILABILITY STATEMENT
Distribution Statement A: Approved for public release. Distribution is unlimited

13. SUPPLEMENTARY NOTES
88ABW-2016-1904, cleared 15 April 2016

14. ABSTRACT The Commanders Predictive Environment (CPE) program objective was to

15. KEYWORDS
natural language processing, machine translation, language modeling, machine learning, big data, optimization

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b.ABSTRACT	c. THIS PAGE			19b. TELEPONE NUMBER (*Include area code)*
UNCLASS	**UNCLASS**	**UNCLASS**	none	79	

This page left blank intentionally.

Table of Contents

List of Figures

List of Tables

Preface

Present-day statistical language and speech processing applications, such as speech recognition or machine translation, can draw on unprecedented amounts of acoustic and text data for most of the mainstream languages of interest. However, the need to process large training data sets requires significant resources and expertise and may lead to computational bottlenecks. Recently, submodularity has been developed as a potential framework for addressing such problems. Submodularity is a framework for discrete optimization first developed in in mathematics, economics, circuit theory, and operations research. Recently, it has been applied to problems in machine learning and to document summarization and data subset selection in speech recognition and machine translation. The objective of this project was to further advance submodular techniques for applications in language or speech processing, including developing submodular methods for sparse feature subset selection and phrase table pruning in statistical machine translation systems, and investigating their scalability to large data sets. This report describes the outcomes of this project.

Acknowledgments

The work reported here is joint work with by Prof. Jeff Bilmes and Yuzong Liu, PhD Candidate, Department of Electrical Engineering, University of Washington.

This page left blank intentionally.

1. Introduction

Present-day statistical language and speech processing, such as speech recognition or machine translation, can draw on unprecedented amounts of acoustic and text data for most of the mainstream languages of interest. The need to process large training data sets (e.g., newswire texts, social media text, podcasts and other audio streams) requires significant resources and expertise and may lead to computational bottlenecks for certain types of statistical models with long training times. In addition, some applications in language processing require not only reducing data sets but also reducing large models down to smaller sizes, or selecting the best set of system parameters. Recently, submodularity has been developed as a potential framework for addressing such problems. Submodularity (Edmonds, 1970; Fujishige, 1991) is a framework for discrete optimization first developed in in mathematics, economics, circuit theory, and operations research. More recently, submodularity has attracted much interest in machine learning where it has been applied to a variety of problems, such as clustering, sensor placement, crowd-sourcing, etc. Submodular functions are functions with the property of diminishing returns, i.e.,

$f(X \cup \{v\} - f(X) \geq f(Y \cup \{v\}) - f(Y)$ for all $X \subseteq Y, v \notin Y$

where *X* and *Y* are two sets and {v} is an individual data item. That is, the incremental value (gain) of item *v* decreases when added to a larger rather than a smaller set. Submodular functions are a natural model for a variety of data summarization, compression, and subselection tasks. In addition, they are easy to optimize and, most importantly, they provide theoretical performance guarantees.

In our previous IARPA-funded project on submodular data selection for machine translation (MT) and speech recognition (Kirchhoff et al., 2013) submodular data subset selection was shown to significantly outperform previous state-of-the-art data selection techniques for both acoustic and text data.

The goal of the present project was to further investigate the application of submodularity to other natural language processing (NLP) tasks and a wider range of statistical models. Specifically, the following research topics were addressed:

- Improve the scalability of submodular data subset selection to large data sets
- Explore additional features for MT data subset selection, in particular those features modeling syntactic structure for use with syntax-based statistical machine translation (SMT) models
- Apply submodular optimization techniques to reduce the size of sparse feature sets in SMT systems
- Apply submodular techniques to phrase table pruning for SMT

In addition, we have investigated another application, viz. submodular data selection for statistical language modeling, which is of relevance to both language and speech

processing, and we have laid the preliminary theoretical groundwork for a novel data selection procedure. The following sections describe each of these topics separately.

2. Improving the Scalability of Submodular Data Selection

2.1 Introduction

Submodular data selection for machine translation as detailed in (Kirchhoff & Bilmes, 2014) was shown to outperform previously proposed methods by a statistically significant margin. This method was based on the following formulation of a submodular feature-based function:

(1) $f(S) = \sum_{u \in U} w_u \;\; g(\sum_{s \in S} m(u,s))$

where *S* is a subset of a ground set (full data set) *V*, *U* is a set of linguistic features, *w* is a feature-specific weight, *g* is a concave function, and *m(u,s)* is a relevance function indicating the important of feature *u* in sample *s*, e.g., a weighted count of *u* in sentence *s*. This function is submodular. For the purpose of data selection this function needs to be maximized subject to a budget constraint that indicates the maximum amount of data to be selected, e.g., the maximum number of sentences or words to be included in the subset *S*.

(2) $\max_{S \subset V} f(S) \;\; s.t. |S| \prec k$

where *f* is the submodular valuation function, *S* is a subset, and *k* is the budget constraint.

Maximization of this function is achieved by the greedy algorithm described below:

Algorithm 1 Framework of GREEDY SELECTOR in pseudocode.

Given: Ground set $V = \{v_i\}$, a list of cost $C = \{c_i\}$ associated with the elements in V, $i = 1, \cdots, N$, a given budget B, and a similarity graph W where w_{ij} is the pairwise similarity between v_i and v_j.
Initialize $S \leftarrow \emptyset$, a priority queue $\rho \leftarrow \emptyset$
for $i = 1, 2, \cdots, N$ **do**
  $\delta \leftarrow f(v_i)$
  $\rho.push(tuple(i, \delta))$
end for
while $\sum_{i \in S} c_i \leq B$ **do**
  $k^* \leftarrow \rho.top().key$
  $\rho.pop()$
  $\delta = f(S \cup \{k^*\}) - f(S)$
  if $\delta > \rho.top().value$ **then**
    $S \leftarrow S \cup \{k^*\}$ {submodularity guarantees that $\delta \geq f(S \cup \{k\}) - f(S), k \in V \setminus S$ }
  else
    $\rho.push(tuple(k^*, \delta))$ {re-sort otherwise}
  end if
end while
return S

Figure 1: Greedy algorithm for the maximization of budgeted submodular functions.

A major problem in expanding the use of this submodular data selection technique to more applications is the scalability to larger data sets. Present-day data sets for statistical machine translation or language modeling often contain hundreds of millions or even billions of words. The standard formulation of the greedy algorithm above requires storing all data elements in a priority queue in central memory, with the subsequent iterative selection of items from the top of the queue. For very large data sets this procedure is computationally inefficient or even infeasible. A naïve way of rendering the application of the greedy algorithm feasible on large data sets would be to split the data into *n* chunks, select *k/n* items from each chunk using the standard greedy algorithm, and combine the resulting sets. However, without any communication between the individual chunks, redundant items could be selected in each chunk, ignoring the property of submodularity and the theoretical guarantees it entails. Another naïve procedure would be to compute the marginal gain of each data item in a parallel fashion, then communicate the results back to a central process, select one item, and communicate the new set of eligible items back to the individual sub-processes. This, however, requires too much computational overhead for process synchronization and communication. Therefore, efficient approximate techniques need to be developed that require neither excessive inter-process communication nor keeping all data in a central memory but that still provide theoretical performance guarantees.

2.2 Approach

Some prior approaches have been developed to the problem of submodular function optimization on large datasets. These include several distributed versions of the greedy algorithm for the budgeted maximization of submodular functions, in particular (Chierichetti et al., 2010) and (Kumar et al., 2013); in addition, multi-stage frameworks have been developed (Mirzasoleiman et al., 2013; Wei and Bilmes, 2014). We have investigated the latter two approaches in the context of data subset selection for NLP applications.

The idea of (Mirzasoleiman et al., 2013) is to have two stages of function optimization. In the first stage, the ground set *V* is distributed across *n* partitions. The greedy algorithm described above is then used to find sets of *K* elements from each partition. The resulting intermediate sets are merged, and the greedy algorithm is run again on the merged set to select *l* elements, where *l* is permitted to be larger than the desired cardinality *k* (to account for the approximate nature of the framework). It was shown in (Mirzasoleiman et al., 2013) that the resulting solution has a guarantee of approximating the optimal solution with the factor $\frac{(1-e^{-\frac{K}{k}})(1-e^{-\frac{1}{K}})}{\min(n,k)}$.

In some cases, this guarantee can be improved further by exploiting the structure of the data set *V*. Unlike other approaches (Chierichetti et al., 2010), this method does not involve multiple passes over the entire data and it can be implemented very efficiently.

(Wei and Bilmes, 2014) have introduced complementary modifications to the original greedy algorithm that addresses the complexity of function evaluation and also aims at reducing the size of the initial ground set *V*, thus speeding up the greedy algorithm. First, approximate function evaluation is introduced that does not require identifying the element with the best marginal gain overall; instead, items within a fraction of the best marginal gain are accepted. The modification is to line 11 in Figure 1: the term $\delta > p.top.value()$ changes to $\delta > \beta * p.top.value()$, where β is a slack factor ranging between 0 and 1. Introducing a slack factor reduces the number of function evaluations and re-sorts (line 14) in the priority queue; the slack factor can also be changed for each iteration of the greedy algorithm. In addition, a pruning procedure is used that eliminates items from the initial ground set *V* before running the optimization algorithm: Given an ordering of all items in the set according to their gain conditioned on all other items (which is computed by one initial pass over the data), a cut-off threshold can be set below which the ground set need not be evaluated. This approach can be combined with that of (Mirzasoleiman et al., 2013) in that it can be run on any of the sub-partitions used in the latter approach.

2.3 Results and Discussions

We experimented with both (Mirzasoleiman et al., 2013)'s and (Wei and Bilmes, 2014)'s methods. The former was used for large-scale language modeling tasks, and the results are described in the corresponding section (Section 6. Language Modeling). The latter was used for

statistical machine translation experiments on the WMT 2014 medical translation task (http://statm.org/wmt14/medical-task), which requires translating medical documents from German into English. The training set consists of 650M words on the source side and consists of several data sources, including the EMEA corpus, data from Wikipedia, UMLS, two parallel corpora from patent databases (COPPA and PatTR), a corpus of medical journal abstracts (MuchMore), and Europarl. The development and test data consist of summaries of medical articles and have 10k words and 21k words, respectively.

A phrase-based statistical MT system was trained on this task. The BLEU score on the test set is 0.278. This performance was matched with a subset of as little as 10% of the training data, using the baseline data subset selection method developed previously. Our goal was to see whether the same performance could be obtained using approximate techniques.

To analyze the effects of the modifications proposed by (Wei and Bilmes, 2014), we first evaluated the effect of the β slack factor on data selection speed. Table 1 shows how the number of re-sorts and the total CPU time as factors of different values of β. CPU time was measured on an Intel Xeon E5-2960 CPU with 2.99 GHz.

β	Avg. # re-sorts per sample	Total CPU time (seconds)	% overlap with original subset	BLEU
1.0	106.51	981.02	N/A	0.287
0.75	27.88	815.96	95.5%	0.287
0.5	21.05	741.67	90.25%	0.287
0.25	15.84	694.26	84.00%	0.287
0.1	9.46	724.48	73.25%	0.285

Table 1: Effect of slack factor on data selection speed and MT performance.

As expected, the average number of re-sorts (Column 2) and CPU seconds (Column 3) go down as the slack factor decreases. Column 4 shows the overlap (in complete sentences) with the original set of sentences selected when $\beta = 1.0$, which is equivalent to the baseline selection method. Column 5 shows the resulting BLEU score of a system trained from the corresponding set. We observe that that even for small values of β more than 70% of the originally selected sentence ids are the same. Moreover, the approximative method has little effect on the BLEU score, most likely because the utterances that are selected – even though they may not be 100% identical to the ones in the original subset - still cover a sufficient number of the n-grams in the test set.

% of ground set	Total CPU time (seconds)	% overlap with original subset	BLEU
100%	981.02	N/A	0.287
75%	1053.78	92.78	0.287
50%	950.76	81.21	0.287
25%	890.51	60.52	0.283
10%	738.97	36.70	0.279

Table 2: Effect of pruning of ground set on speed and performance.

Table 2 shows a similar comparison for different levels of initial pruning of the ground set *V*. Here we see that moderate pruning (eliminating e.g., 25% of *V*) occurs a slight penalty in speed due to the need to threshold-test every sample for constructing the initial priority queue, which outweighs the reduction in the number of function evaluations once the priority queue has been built. More aggressive pruning results in a clear improvement in speed. The performance of the resulting MT systems is comparable to that of the baseline system except for very high levels of pruning (eliminating more than 80% of the ground set).

2.4. Conclusions

We found that the speed of submodular data subset selection can be reduced significantly by introducing approximations into the basic accelerated greedy algorithm, in particular a slack factor for function evaluation and initial pruning of the ground set *V*. These speed-ups may be particularly advantageous when performing subset selection of large data sets on the fly. With respect to speed alone, using a slack factor seems preferable to pruning when it is feasible to fit all data into memory. However, for those scenarios where the size of the data exceeds the available memory, pruning will be required. The two methods can be combined with each other as well as with other approximative methods, such as (Mirzasoleiman et al., 2013; 2014).

It should also be considered that the larger bottleneck for feature-based selection functions in particular is the extraction of the features themselves. The CPU times listed in Table 1 and Table 2 do not include the initial feature extraction for the training and test data but only reading in feature values and (if desired) feature weights stored offline and then performing the steps in Figure 1. Feature extraction for the training and test data used in the above experiments takes around 800 seconds on this data set, about as long as the actual subset selection itself. This could greatly be reduced by utilizing smaller feature spaces.

3. Integration of Additional Features Into Feature-Based Submodular Functions

The formulation of the feature-based submodular function in Equation (1) offers the possibility of utilizing a wider range of features than previously considered in data selection for machine

translation. In addition to standard n-gram features utilized in (Kirchhoff & Bilmes, 2014) we have considered several obvious choices for additional features, including hypothesized target-language n-grams for the test data, confidence values resulting from word alignments, as well as parse tree based features encoding the underlying syntactic structure of sentences.

3.1. Hypothesized Bilingual Features

3.1.1 Introduction

In our previous work the feature set consisted of source-language n-grams (up to a user-defined value of *n*, typically 5 to 7) that occur in both the training set and in the development/test sets. For a bilingual version of the feature set, target-language n-grams occurring in the development set references were used in (Kirchhoff & Bilmes, 2014). This led to slight improvements in most data conditions.

In order to improve over this version of the feature set, target-language n-grams extracted from test set translation hypotheses produced by the MT system could be considered. Several previous studies in statistical machine translation have made use of hypothesized translations as data: Bertoldi & Federico (2009) utilized "hallucinated" references (i.e., translation hypotheses) for system tuning. In (Irvine & Callison-Burch, 2014), translation hypotheses were used to generate novel phrase translations.

3.1.2 Approach

Our baseline system is a phrase-based statistical Arabic-English MT system, trained on a variety of LDC-released parallel corpora totaling approximately 185M words on the source side. The development set is the NIST MT06 set; the test set is the NIST MT09 set. For further details on the system see (Kirchhoff & Bilmes, 2014). We tested four different data conditions for data subselection, viz. 10%, 20%, 30% and 40% of the original data set. 1-best translation hypotheses were generated for the test set using the MT system trained on the subselected data set for each of the four different data conditions. Unknown words were stripped from the MT output and n-grams up to the order of 5 were extracted. These n-grams were added to the feature set *U*. Tf-idf weights were computed for all features, including the hypothesized features.

3.1.3 Results and Discussion

Table 3 shows BLEU scores for data subselection with monolingual features, bilingual features with only development set references, and bilingual features based on translation hypotheses on the test set. The BLEU score of the baseline system utilizing 100% of the data is 0.4257.

	10%	20%	30%	40%
Monolingual features	0.4303	0.4334	**0.4371**	0.4349
Bilingual features from dev set	**0.4330**	**0.4395**	0.4333	**0.4366**
Hypothesized target-language features	0.4325	0.4354	0.4317	0.4295

Table 3: MT results (BLEU) using bilingual features extracted from translation hypotheses.

We see that the latter improved the MT performance on small data subsets (10% and 20%) slightly over monolingual features only. However, performance deteriorated on the larger subsets, and the improvements from bilingual features that were extracted from the dev set only were larger in all cases.

3.1.4 Conclusions

The use of n-gram features hypothesized by an MT system may be of value when reducing the original data set down to very small sizes. As the subset grows larger, however, the hypothesized target-language features may bias the system towards including too many low-quality sentence pairs from the training set, leading to worse results. This method could potentially be improved by including confidence weights for hypothesized n-grams. For this particular translation task, target-language features extracted from the development set references provided a larger boost in performance than hypothesized test set features.

3.2 Confidence-Weighted Features

3.2.1 Introduction

The standard feature-based selection function above solely focuses on the coverage of features in the test set; it does not incorporate any measure of the quality of training samples, i.e. whether a given training sample is a good translation pair or not. In order to address this issue, we have run experiments where additional confidence measures were utilized during data selection.

3.2.2 Approach

During the standard selection process the gain of a given sample in the context of the already-selected subset is normalized by the sentence cost. Thus, in each iteration t of the selection algorithm, the set A_t is as follows:

(3) $A_{t+1} = A \cup \ argmax_{a \in V \setminus \{a\}} \frac{f(a|A_t)}{|a|^{\alpha}}$

where $|a|$ is the length (number of words) of sample a. Confidence values are incorporated by multiplying the gain term by an additional factor $\gamma(a)$:

(4) $A_{t+1} = A \cup \ argmax_{a \in V \setminus \{a\}} \frac{f(a|A_t)}{|a|^{\alpha}} \gamma_a$

The confidence factor for a sentence in the training set is computed as follows: we first compute the average of the word alignment scores resulting from the source-to-target and target-to-source alignments of the sentence pair, normalized by sentence length:

(5) $\gamma_a = \frac{p_{wa}(e_a|f_a)+p_{wa}(f_a|e_a)}{|e_a|+|f_a|}$

where f_a is the source side and e_a is the target side of sample a, and p_{wa} is a word alignment probability derived from a word alignment model (the assumption being that such a model is available, either from an existing system for the same language pair, or from some outside model). In our case, the scores were obtained from an IBM-4 word alignment model. Since the scores showed a very large range they were binned (non-uniformly) into 10 bins, and fixed confidence scores ranging between 0.0 and 1.0 were associated with each bin. This method eliminates samples from consideration that have very low alignment scores by essentially assigning them a zero weight. Others are weighted in relation to their alignment probability.

3.2.3 Results

Initial experiments showed that the BLEU scores decreased by 0.05 absolute on average (from the baseline numbers shown in Table 3) due to confidence weighting. An analysis showed that the above weighting scheme is closely correlated with sentence length. It tends to eliminate longer sentences because they tend to have lower alignment scores (despite the normalization for sentence length in Equation (5)), whereas short sentences usually have highly reliable alignment scores. Thus, longer sentences that contribute valuable n-grams are tend to be discarded.

3.2.4 Conclusions

While confidence features in general might improve training data subselection, especially when the data is noisy, confidence scores based on sentence-level word alignment scores seem to have a detrimental effect, since longer sentences have inherently less reliable word alignments. The use of confidence features could be revisited in the future using other methods of identifying unreliable translations, such as semantic features. Furthermore, n-gram level confidence scores could be explored.

3.3. Structural Syntactic Features

3.3.1 Introduction

The feature sets used previously included only surface n-gram features. A wide range of additional features could be utilized, in particular features encoding the syntactic structure of the source (and/or target) sentence. These features might be helpful to translation models that explicitly model syntactic structure, such as tree-to-string models (Hopins and Kuhn, 2007; Zhang et al., 2007a; Wu et al., 2010) or tree-to-tree models (Zhang et al., 2007b; Ambati et al., 2009). By including syntactic features those data samples can be selected that resemble the test data not only in terms of surface word strings but also with respect to the underlying syntactic structure.

3.3.2 Approach

There is a large number of possible ways of expressing syntactic structures in terms of discrete features. The most common method is to extract features from parse trees (dependency trees or constituency trees), e.g. by enumerating tree substructures. Parse tree feature spaces are an instance of nested feature spaces that grow exponentially with the sizes of the objects they describe. In principle, the feature space consists of all possible substructures that can be extracted from the parse trees derived from the training data, which is an infeasible number of features to handle.

We chose to represent syntactic structure in the form of dependency trees, using the Stanford lexicalized parser as an annotation tool. In order to limit the size of the feature space we consider only subtrees of a user-specified depth *d* that are anchored in a terminal element. That is, for each lexical item in the source sentence, that subtree is extracted that contains the item's parent nodes up to *d* levels above the terminal level, as well as the immediate left and right siblings of the top-most node considered. The immediate parent node of the terminal node in this representation is always a part-of-speech tag, whereas the higher-level nodes indicate phrasal categories. Thus, the subtrees provide information about the intermediate phrasal structure the word occurs in; however, the tree is anchored to a word as it would otherwise be too general to provide meaningful information during data selection (a large variety of sentences can have similar syntactic structure at the phrasal level).

Each of the complete subtrees is considered a feature. For example, in the annotated sentence shown in Figure 2.

```
(ROOT (S (NP (NN time)) (VP (VP (MD wo) (RB n't) (VP (VB ease)
(NP (PRP$ your) (NN pain))))
```

Figure 2: Syntactic annotation for parse feature extraction.

the syntactic features for d=1 are:
time-NN-NULL-NULL

wo-MD-NULL-RB

n't-RB-MD-VP
ease-VB-NULL-NP
your-PRP$-NULL-NN
pain-NN-PRP$-NULL

For d = 2, the following syntactic features are added:

time-NN-NP-NULL-VP
wo-MD-VP-NULL-NULL
n't-RB-VP-NULL-NULL
ease-VB-VP-RB-NULL
your-PRP$-NP-VB-NULL
pain-NN-NP-VB-NULL

Syntactic features are added to the features set *U* in Equation (1) and are weighted by tf-idf, like their n-gram counterparts.

3.3.3 Results and Discussion

Parse-tree based features were evaluated on an English-Chinese machine translation task using the UM corpus (Tian et al., 2014). The training set consists of 2.1M sentence pairs (40M source words); the test set consists of 3554 sentence pairs (40k source words). The source side of the data was annotated with dependency trees using the Stanford lexicalized parser, and a tree-to-string based SMT system was trained using Travatar (Neubig, 2013). The baseline performance on the test set is 0.3403, which is a state-of-the-art performance, outperforming the baseline score of 0.3155 for a phrase-based, non-syntax based SMT system given in (Tian et al., 2014).

Subsets between 10% and 40% of the training data were selected using the standard data selection method that utilizes source-language n-gram features only, and the tree-based systems were retrained on each set. Table 4 (first row) shows the corresponding BLEU scores on the test set, which represent averages of three different systems resulting from different weight optimization runs. The best performance is obtained with a subset of 20% at 0.3430. The second row shows the change in performance when tree-based features are added. We observe small (not statistically significant) but consistent improvements over the baseline. The feature space increases from around 300k for n-gram features to 340K when parse tree features are included.

	Data set sizes			
System	**10%**	**20%**	**30%**	**40%**
Baseline (n-gram features)	0.3383	0.3430	0.3413	0.3420
+ parse tree features	**0.3410**	**0.3433**	**0.3427**	**0.3450**

Table 4: BLEU on UM-corpus test set with and without parse tree based features.

3.3.4 Conclusions

The addition of parse tree features derived from dependency tree representations resulted in small but consistent improvements in MT performance when used with a tree-to-string translation model. This method may be suitable when selecting data for syntax-based translation models, since the annotation of the entire training corpus by a parser is independently required for training the translation model. A variety of other parse-based features could be explored; however, it is important to manage the size of the feature set and avoid too much computational overhead when extracting the features. In view of the potential computational bottleneck incurred by feature extraction (see Section 2.4. Conclusions), initial experiments with feature hashing (Ravichandran et al., 2005) were conducted. Locality-sensitive feature hashing uses hashing techniques to map a high-dimensional set of features into a lower-dimensional space in a way that preserves the similarity between features. While the feature space can be reduced by an order of magnitude in this way, this technique led to a significant loss in performance in initial experiments. Future work using alternative dimensionality reduction techniques, drawing e.g. on recent work on using neural networks to extract linguistic features, could address this problem.

4. Submodular Feature Selection in Statistical Machine Translation

4.1 Introduction

Many MT system makes use of a large but sparse set of discrete features (Chiang et al., 2009) in the log-linear translation model. These are features such as n-grams of various orders, word deletion or insertion features that indicate which source (target) word has been deleted (inserted), phrase length features, etc. The dimensionality of this feature set can be very large; however, most features are not relevant to any given translation hypothesis and will only occur a small number of times in the total data set. Since system tuning with large feature sets is slow and may lead to overfitting in the system, it is advisable to reduce the feature set as much as possible without losing information. To this end we have investigated submodular feature selection.

4.2 Approach

Our approach to submodular feature selection is based on (Liu et al., 2013; Kirchhoff et al., 2013) and was modified for the sparse, discrete features typically used in SMT systems. The approach utilizes graph-based submodular functions for selecting a subset of the existing feature set. First, a pairwise similarity graph is constructed over all features, where nodes represent individual features and edges are weighted with pairwise feature similarity scores. These scores represent the pointwise mutual information between features, and they are computed from their frequencies in the parallel training data. For example, for the features under consideration, the mutual information between target word translation features and target bigram features might be high. After the graph has been constructed, a graph-based submodular function (such as graph cut, saturated graph cut, facility location, etc. – see (Kirchhoff et al., 2013)) is optimized using the greedy algorithm in order to select a subset matching the desired budget. The feature

selection algorithm is run offline before general MT system tuning is initiated, in order to prune features that are highly redundant with each other. Feature weights then need to be optimized only for those features that survive feature selection.

The above procedure selects a subset A of sparse features with the goal of eliminating redundancy only. In addition to this basic method we developed a new method which we call *test-set adaptive feature selection*. This is similar to test-set adaptive data selection in that the graph-based selection criterion *g(A)* is supplemented by a relevance term *r(A)* that indicates the relevance of the sparse features to the test data:

(6) $f(A) = g(A) + r(A) = \sum_{i \in V} \max_{j \in A} s_{ij} + \lambda \sum_{j \epsilon A} w_j c_{test}(j)$

where *A* is a feature subset, *V* is the ground set, *s* is a similarity weight, *w* is a feature weight obtained during system tuning, λ is a weighting parameter, and $c_{test}(j)$ is the count of feature *j* in the test set. For target-language features, the translation hypotheses produced for the test set are used to extract features. The motivation here is to select not only those features that are inherently non-redundant but features that occur frequently in the test data and that receive a large weight during tuning.

A third version of the selection function integrates feature selection with feature weight tuning. Feature selection is run after every iteration of feature weight tuning. The selection function integrates over different feature weights produced in *T* different iterations of the weight tuning process, with a decay parameter α for each iteration:

(7) $f(A) = g(A) + r(A) = \sum_{i \in V} \max_{j \in A} s_{ij} + \lambda \sum_{j \epsilon A} (\sum_{t=1}^{T} \alpha_t w_{tj}) c_{test}(j)$

This is intended to lend more stability to the feature selection process and to avoid features from being pruned that happen to receive a low weight in a single iteration.

4.3 Results and Discussion

In order to evaluate the selection of sparse features for MT systems we trained baseline systems with sparse features for the Transtac spoken dialog translation task for Iraqi Arabic-English, and for the Arabic-English NIST task. The systems utilize the following sparse features:

- Word translation features
- Source word deletion features
- Target word insertion features
- Target-side bigram features
- Phrase-length features

All features were restricted to the most frequent *N* vocabulary items (*N = 50*), which already reduces the initial size of the feature set. After the sparse features were extracted they were

subselected by the graph-based submodular method with the facility location function before undergoing system optimization, i.e. feature weight estimation. kbMira was used for optimizing the feature weights.

Initial experiments were performed on the Transtac system, which is a smaller scale system with a training set of 761k sentences (7M words); the development and test sets contain 7000 sentences (62k words) and 2900 sentences (29k words). Its full sparse feature set contains 3000 features. Only the basic feature selection method (non-iterative, non-adaptive) was used for this system. Table 5 shows the resulting BLEU scores for the baseline systems and two systems with feature selection; the best results were obtained when reducing the feature set to 50% of its original size.

System	**BLEU**
Baseline (no sparse features)	0.321
+ 100% of sparse features	0.327
+ 10% subset of sparse features	0.329
+ 50% subset of sparse features	**0.334**

Table 5: BLEU scores for the Transtac task, feature subset selection.

The reduction of the sparse feature set leads to an actual improvement in MT performance of almost 1 BLEU point total compared to the system with the full feature set.

We next tested sparse feature selection in the NIST MT system, using the baseline (non-adaptive, non-iterative) method for feature selection. The size of the initial feature set was 7k. The BLEU scores in Table 6 show that the performance of the subselected systems matches that of the full set.

System	**BLEU**
Baseline (no sparse features)	0.423
+100% of sparse features	0.427
+ 10% subset of sparse features	0.424
+ 20% subset of sparse features	0.427
+ 50% subset of sparse features	0.427

Table 6: BLEU scores for NIST translation task, feature subset selection.

In our experiments on the NIST 2009 Arabic-English MT task test-set adaptive sparse feature selection yielded slightly but non-significantly worse results than simple redundancy-based feature selection. The following table shows BLEU scores on the MT09 Arabic-English evaluation set, for different types of selection procedures:

System	BLEU
Baseline (no feature selection)	0.427
Submodular non-adaptive selection, 20% subset	0.427
Submodular test-set adaptive selection, 20% subset	0.426
Submodular non-adaptive selection, 50% subset	0.427
Submodular test-set adaptive selection, 50% subset	0.426

Table 7: BLEU scores on NIST task with adaptive vs. non-adaptive feature selection.

Finally, we tested iterative feature selection integrated with weight tuning, which prunes the sparse feature set after each iteration of weight tuning. Experiments with iterative feature selection on the NIST MT task yielded significantly worse results than non-iterative selection, decreasing the BLEU score by 1% absolute on the NIST MT09 test set. Our analyses showed that the composition of the feature set fluctuates strongly between different iterations, despite the integration over different sets of weights shown in Equation (7). Thus, weights could not be tuned reliably within a fixed number of iterations, and the resulting feature sets and weights were unreliable.

4.4 Conclusions

Our experiments with submodular feature selection showed that it is possible to reduce the set of sparse features by a large amount (90% on the smaller Transtac task, 80% on the NIST task) without a loss in performance. For a reduction of 50% the MT performance even increased appreciably in the Transtac system; no increase was observed in the larger MT system. Test-set adaptive feature selection did not provide gains over non-adaptive selection; again, the reason may be the unreliable nature of features hypothesized on the test set.

5. Submodular Phrase Table Pruning

5.1 Introduction

A major component in statistical MT systems is the phrase table, which lists all possible phrase translations extracted from the training set. Phrase tables can contain millions or even billions of entries; although these are typically filtered for the purpose of testing, the resulting smaller phrase tables can still be too large for efficient decoding (e.g., under resource-constrained conditions such as hand-held devices). Thus, it is desirable to prune phrase pairs from the table to

decrease the amount of storage and computation needed without incurring a loss in MT performance. Several methods for *phrase table pruning* have been developed in the past. The most basic method is *threshold-based pruning*. Given a phrase table where each entry specifies a mapping between a target phrase e and a source phrase f, along with a set of scores, entries can be pruned when either of the phrase translation probabilities *p(e|f)* or *p(f|e)* fall below a given threshold. Another method is based on statistical significance testing (Howard et al., 2007). The idea is to prune those phrases that are not reliable translations but merely chance phrase pairs resulting from spurious word alignments. Thus, a phrase pair is eliminated when its p-value (computed from its observed frequency in the training data using a hypergeometric distribution) is higher than a pre-defined threshold threshold. Finally, the relative-entropy approach (Zens et al., 2012) seeks to find a pruned phrase table for which the Kullback-Leibler (KL) divergence (or relative entropy) between its probability distribution and that of the original unpruned phrase table, $D(p'(e|f)||p(e|f))$, is minimal. In order to compute this exactly, all possible subsets of the phrase table would have to be constructed and the relative entropy would have to be computed, which is computationally infeasible. In practice, the simplifying assumption is made that individual phrase pairs contribute to the KL divergence independently, and phrase pairs are pruned in a single pass according to $p(e,f)[\log p(e|f) - \log(p'(e|f)] < \theta$, where θ is a threshold. The pruned score $p'(e|f)$ is computed by finding the maximum score out of all possible decompositions of the phrase pair into smaller phrase pairs (see (Ling et al., 2012)):

$$(8) \qquad p'(e|f) = \max_{s \in S_{(e,f)}: target(s)=e} [\prod_{i=1}^{|s|} p(s_i)]$$

where

$$(9) \qquad S_{(e,f)} = \{(e', f'): e' = substr(e) \wedge f' = substr(f)\}^{\{2\}+}$$

The set *S(e,f)* is the set of all decompositions of the phrase pair *(e,f)* into smaller sub-phrases. The best (i.e., highest probability) decomposition is determined by dynamic programming, with the constraint that the concatenated target sub-phrases must yield *e* ("forced decoding"). The probability of that decomposition (the sum of the log-probabilities of the component phrase pairs) is *p'(e|f)*. When a segmentation into shorter phrases cannot be found, the phrase pair is assigned a constant value that prevents it from being pruned. Entropy-based pruning has been shown to outperform significance-based and threshold-based pruning (Zens et al., 2012) and heuristic pruning based on counts/probability thresholds. In (Ling et al., 2012) the evidence in favor of the relent vs. the significance-based pruning method is less clear. (Ling et al., 2012b) combines both methods by taking the *min* of the relent and the significance values.

The problem with all of the above approaches is that they do not take inter-dependencies between phrases into account. Phrase pairs are pruned in a random order, regardless of whether they may be parts of longer phrase pairs or not. A phrase pair that is pruned may be a crucial component in the composition of other, longer phrases, which may have an effect on the scores according to which other phrase pairs are pruned. Previous methods did not find a solution to this problem.

5.2 Approach

We have formulated two submodular objectives for the problem of phrase table pruning. Both take the goal of minimizing the KL-divergence between the original phrase table and the pruned table as a starting point, similar to the objective in relative-entropy pruning.

We express the divergence between the probability distribution of the full phrase table P_V and that of its subset, P_S, as:

(10) $$D(P_V||P_S) = \sum_{e,f} P_V(e,f)\, log\, \frac{P_V(e,f)}{P_S(e,f)}$$

The objective is to minimize this divergence; however, minimization of submodular functions subject to constraints (such as the size of *S*) is NP-hard, and no satisfactory solution with theoretical guarantees has been found yet. Thus, based on Equation (10) we define the valuation function f*(S)* for a subset *S* of P_V as

(11) $$f(S) = \sum_{f,e\in S} p(e,f) \log m_{e,f}(S) - \sum_f p(f) \log m_f(S)$$

where *m(S)* is the relative-frequency estimate obtained from subset *S*. Minimization of (10) corresponds to maximization of (11). (11) in turn is not a simple function but the difference of two submodular functions. We have investigated possible methods for this problem, building on (Iyer and Bilmes, 2012). For unconstrained minimization of the difference between submodular functions, each of the two terms can be replaced with its modular upper (lower) limit, resulting in submodular-supermodular, supermodular-submodular, or modular-modular optimization procedures with theoretical guarantees. In our case constraints need to be placed on *S*.
For cardinality-constrained optimization of a function like (11), approximate versions of the optimization procedures in (Iyer and Bilmes, 2012) can be defined that do not provide optimal solutions but good heuristic solutions in many cases. However, we have the additional constraint that there needs to be a way of producing a probability estimate for *p(e|f)* even when the phrase pair *(e,f)* is removed from the table. Incorporating these dependencies into the submodular objective unfortunately leads to an intractable optimization problem.

The second approach to formulating phrase table pruning as a submodular objective is through the following valuation function:

(12) $$f(S) = \sum_{(e,f)\in S} P_V(e,f) \log P_S(e|f) + \sum_{(e,f)\notin S} P_V(e,f) \log P_S(e|f)$$

where $logP_S(e|f)$ is equivalent to the maximum probability estimate resulting from forced decoding as explained in Equation 8. This function could be maximized, subject to cardinality constraints on *S*, using the standard greedy algorithm. However, forced-decoding of the entire subset *S* needs to be re-run. After initial attempts at finding an efficient implementation for this strategy, this approach was abandoned as computationally too complex.

As an alternative to the computationally intensive submodular methods, we therefore designed a simpler solution to the problems of taking inter-phrasal dependencies into account. Under this approach we compute relevance scores that can be associated with a given phrase pair, and that can be combined with other scores for the purpose of pruning. The relevance score for a phrase pair *(e,f)* is the relative frequency with which it occurs in the maximum-probability decomposition (Equation 8) of all longer phrases that it is contained in. Relevance scores are then normalized over the entire phrase table and used in a weighted combination with significance-based or relative entropy scores. Thus, we utilize a modular selection function, but we enrich it with a feature that expresses inter-dependencies between different data samples and that thus acts a proxy for a submodular selection criterion.

5.3 Results and Discussion

As a baseline system we used our in-house Iraqi Arabic-English MT system trained for the Transtac task (translation of spoken dialogs, see e.g., (Kirchhoff et al., 2015)). The complete phrase table extracted from the training data has 11M entries; the filtered phrase table for the test set has 867k entries. The baseline BLEU score on the test set is 0.333.

We first tested several baseline methods for phrase table pruning, including significance-based pruning and relative-entropy based pruning. Threshold-based pruning methods perform significantly worse than these – e.g., a decrease in BLEU of 1.5 was found for a reduction of the phrase table of only 10%). This is likely due to the fact that training data for this task is sparse and not very noisy; moreover, high phrase translation probabilities (esp. for 1-count phrase pairs) are not necessarily indicative of highly reliable translations but result from sparse data. Columns 2 and 3 of Table 8 show results for relative entropy and significance based pruning. In both cases, phrase pairs were ordered according to their scores obtained by the respective methods, and the top N % were retained in the pruned table. N was varied from 10 to 70. Unlike (Zens et al., 2012) but in line with (Ling et al., 2012) we found that significance-based pruning is equivalent in performance to relative entropy-based pruning, and superior when the phrase table is pruned very aggressively, retaining only 10-40%. Column 4 shows results obtained using combined relative entropy and significance scores, as described in (Ling et al., 2012b). Columns 5 and 6 show results obtained when combining the relative entropy or significance-based methods with relevance scores. Adding relevance scores outperform all baseline methods; in the 10% and 20% conditions, the difference is statistically significant. It is noticeable that some pruned tables (relative entropy + relevance in the 40% - 70% range) even outperform the baseline system performance of 33.3.

% of phrase table	Relent	Signif	Relent+Signif (w = 0.3,0.7)	Relent+Relev (w = 0.3,0.7)	Signif+Relev (w = 0.3,0.7)
10	0.275	0.307	0.302	0.323	**0.326**
20	0.306	0.325	0.324	0.328	**0.333**
30	0.318	0.335	0.334	0.333	**0.336**
40	0.325	0.336	0.334	**0.335**	0.332
50	0.330	0.332	0.335	**0.338**	0.332
60	0.332	0.332	0.331	**0.338**	0.331
70	0.333	0.331	0.331	**0.337**	0.332

Table 8: BLEU (%) scores for different phrase table pruning methods (relent = relative entropy, signif = significase-based, relev = relevance score; w = combination weights).

5.4. Conclusions

Developing an efficient submodular technique for phrase table pruning turned out to be more complicated than expected. While designing an appropriate submodular objective function is not problematic per se, the corresponding optimization techniques are as yet not efficient enough to be used in practice. An alternative strategy was therefore developed that uses a modular selection procedure with an additional feature that expresses inter-dependencies between different phrase pairs in the table, viz. a relevance score that measures the importance of a phrase pair to the larger phrase pairs it is contained in. When combining this score with baseline methods, statistically significant improvements were obtained over state-of-the-art pruning methods.

6. Language Modeling

6.1 Introduction

Another obvious application for submodularity is the selection of training data for language modeling. For many languages, a large amount of monolingual text data is available to train language models (LMs), which typically exceeds the size of parallel data corpora in MT. At the same time, there is a strong tendency towards neural modeling techniques, including recurrent neural network language models (RNNLMs) (Mikolov et al., 2010) or long short term memory (LSTM) networks. Since these models are rapidly becoming the state of the art, we have additionally explored submodularity in combination with neural LMs. Data subset selection is particularly relevant in this context due to the long training times of neural LMs.

6.2 Approach

Data sets for language modeling typically include terabytes of data. The feature representations of such data sets cannot fit into a central memory and thus cannot be processed by the basic greedy algorithm shown in Figure 1. Thus, one of the approximate methods discussed in Section 2 needs to be used. The most appropriate approximate method is the two-pass strategy proposed in (Mirzasoleiman et al., 2013): the data is first split into manageable chunks, each of which is processed separately and in parallel by the standard greedy algorithm. The top N utterances are then selected from each chunk and are combined into a new data set, which is then processed by the greedy algorithm in a second pass.

In addition to the scalability issue we were interested in investigating whether any interactions exist between the data selection method and the modeling technique, e.g., whether neural LMs differ from standard backoff LMs in how data subselected by different methods is utilized. For example, data selected by the cross-entropy method of (Moore et al., 2012) is inherently more redundant.

6.3 Results and Discussion

The data set for our language modeling experiments consists of approximately 2.5B words of English data drawn from a variety of LDC corpora (e.g., the English Gigaword, ANC, and HARD corpora, the target sides of the GALE MT data sets, etc. – see (Kirchhoff and Bilmes, 2014)).

The first goal was to scale data subset selection to the larger data size. For a set of this size, we can only use the approximate two-pass method discussed above. The entire data was split into approximately 1000 chunks. From each chunk, the top-ranked sentences equaling 20M words each were selected and combined into a new corpus, which was then processed with the exact submodular data selection method. Since the entire original data representation does not fit into memory we cannot compare the results to non-approximate data selection. However, we can compare submodular data selection to other baseline methods, such as the cross-entropy method, in order to assess how the approximate submodular method compares to more established methods.

In our initial experiments we subselected the training data using both the cross-entropy and the submodular methods and then trained 5-gram backoff language models on sets ranging from 10M to 70M words. The data was selected for language modeling in our NIST Arabic-English MT system; thus, the data was selected to match the n-grams found in the target-language references of the MT06 set (the development set), and in the target side of the MT system's phrase table filtered for the MT09 test set.
The vocabulary of the LMs was restricted to the set of unique words in those n-grams. Words not in vocabulary were mapped to an <UNK> token; the probability for this token is proportional to its frequency in the training data (rather than using dedicated probability estimation for unknown words). Witten-Bell smoothing was used. Their perplexity was evaluated on the references of the MT06 set; the results are shown in Table 9.

Data set size (# words)	Submodular selection		Xent selection	
	NW	**WB**	**NW**	**WB**
10M	253.55	459.29	290.92	1146.88
20M	230.89	482.32	263.91	1020.82
30M	229.82	501.15	243.02	1003.83
40M	232.45	517.08	231.01	958.48
50M	233.45	519.23	223.27	913.62
60M	227.21	487.62	217.16	889.93
70M	210.09	432.43	211.73	862.22

Table 9: Perplexities for 5-gram backoff LMs trained using either submodular or cross-entropy (Xent) data selection.

Performance is reported separately on newswire (NW) and web (WB) data genres; however, LM data was selected jointly for both conditions, using the combined development set as a query set. The differences in perplexity between the submodular and cross-entropy methods are considerable for small data sets (10M – 30M) in the newswire condition. In order to reach the same performance as the submodular model in the 20M case, the cross-entropy model requires twice the amount of data. Submodular data selection outperforms cross-entropy selection in all of the web data condition.
The differences in perplexity are most likely due to the fact the cross-entropy method does not control for redundancy. Thus, more data is needed in the case of cross-entropy selection to achieve the same performance as the language models trained using submodular data selection. Moreover, since the data was selected jointly for both genres in a single pass, the cross-entropy model achieves very poor performance on the web data genre due to the preponderance of redundant sentences matching the newswire data.

Next, RNNLMs were trained and evaluated on the same data. The models are unidirectional RNNs with gated recurrent units (GRUs) and a hidden layer size of 200. They use an unlimited history (up to the beginning of the sentence). Models were trained using noise-contrastive estimation. The vocabulary was limited as above, with out-of-vocabulary words being mapped to a single <UNK> token.

Data set size (# words)	Submodular		Xent	
	NW	**WB**	**NW**	**WB**
10M	103.40	262.28	165.19	373.13
20M	98.13	265.06	148.99	376.81
30M	97.39	269.41	140.97	380.27
40M	95.70	276.73	137.03	385.05
50M	108.34	248.59	138.57	408.91
60M	112.80	253.33	136.85	441.68
70M	111.35	256.84	134.60	384.37

Table 10: Perplexities for RNNLMs trained using submodular vs. cross-entropy (Xent) data selection. (NW = newswire, WB = web data).

In general, the RNNLMs achieve lower perplexities than their back-off model counterparts. The cross-entropy RNNLMs seem more robust to inherent data redundancy than the cross-entropy backoff models; however, RNNLMs trained using submodular data selection still achieve lower perplexities than the RNNLMs trained with cross-entropy data selection under all conditions.

Finally, the best submodular and cross-entropy RNNLMs were used for second-pass rescoring in our NIST Arabic-English MT system. This system already uses submodular data selection for downselecting the parallel training data for the translation model. N-best lists of up to 200 unique hypotheses per sentence were generated in a first decoding pass; the hypotheses were then rescored with the RNNLMs, and the feature weights for the log-linear model in the system were jointly re-optimized by minimum error rate training (MERT) on the development set. Table 11 shows the MT results on the MT09 test set.

	Data set sizes			
Model	**10%**	**20%**	**30%**	**40%**
Baseline LM	0.4289	0.4344	0.4378	0.4360
+RNNLM-Xent	0.4324	0.4365	0.4373	0.4379
+RNNLM-Submod	**0.4351**	**0.4385**	**0.4415**	**0.4410**

Table 11: BLEU scores on the NIST Arabic-English MT09 test set. The baseline indicates first-pass decoding results without subsequent rescoring.

6.4. Conclusions

Training data selection for language modeling is another obvious application for submodular data selection. We have compared submodular data selection against cross-entropy based data selection for both back-off 5-gram models and the RNNLMs. Due to the size of the training data set, approximate submodular data selection had to be used. We observe that the resulting models still outperform models trained using cross-entropy

data selection in perplexity and subsequent MT performance when used in a second-pass rescoring step.

7. New theoretical approaches

7.1 Introduction

The standard submodular objective function we have been using for query-based data selection (selection of data based on an existing test or query set) attempts to simultaneously select data that is relevant to the query data while minimizing the redundancy in the selected data set. These two goals inherently compete with each other.
In addition to the main topics addressed above we have therefore conducted preliminary work on developing a new submodular approach for query-based data selection that provide a cleaner theoretical foundation for this problem.

7.2 Approach

To this end we have developed a two-step procedure that first selects a subset of the data that is relevant to (i.e., maximally redundant with) the test set. In a second step the inherent redundancy in the select set is removed. Given a valuation function f_1 and a ground set V that can be divided into training set V_{trn} and V_{tst}, the goal in the first step is to find the maximal subset S of V_{trn} such that $f(S|V_{tst}) < \varepsilon$, i.e. the conditional information of S given the test set is minimal. In other words, the subset S is required to be maximally redundant with the test set, yielding a subset with high similarity to the test data. The conditional information is computed as

(13) $f_1(S|V_{tst}) = f_1(S \cup V_{tst}) - f_1(V_{tst})$

where S is a subset, V_{tst} is the test (query) set, ε is a hyperparameter, and f_1 is a valuation function (e.g., the same feature-based selection function as before, based on a feature set derived from the joint training and test data). All sentences for which $f(S|V_{tst})$ is less than a threshold ε are selected for the initial subset. The resulting data set is then subject to a second stage with a valuation function f_2 designed to remove inherent redundancy. Both f_1 and f_2 are implemented as feature-based functions as before. The difference between f_1 and f_2 is in the definition of the feature set U, which is derived jointly from the original training and test sets for f_1 and from the subset S only for f_2. Thus, the original feature set only needs to be explored during the first stage in which each sentence is processed individually; for the more complex second stage, in which dependencies among different sentences are considered, the feature set is smaller. This criterion lends a stronger theoretical foundation to the data selection procedure: whereas the original selection function was designed to achieve both data compression and relevance-based selection in one step, the current scheme separates both goals and optimizes them separately. However, the additional requirement is defining the hyperparameter ε.

7.3 Results and Discussion

In initial experiments on the NIST Arabic-English MT task the two-step procedure did not show improvements (see Table 12 below). At small data set sizes, the 2-step procedure performed worse whereas it was slightly but non-significantly better at larger data set sizes. However, only a small set of values for the hyperparameter ε has been investigated so far. It is possible that ε needs to be adjusted individually for a given data set size.

Selection procedure	**10%**	**20%**	**30%**	**40%**
1-step	0.4302	0.4334	0.4371	0.4349
2-step	0.4195	0.4237	0.4380	0.4365

Table 12: BLEU scores on NIST MT09 test set for different selection procedures.

At small data set sizes, the 2-step procedure performed worse whereas it was slightly but non-significantly better at larger data set sizes. However, only a small set of values for the hyperparameter ε was investigated. It is possible that ε will need to be adjusted individually for a given data set size.

7.4 Conclusions

The initial theoretical and experimental work on the 2-step data selection method is valuable as a starting point but needs to be fine-tuned and investigated empirically in more detail.

Summary Conclusions

In sum, this project has made the following contributions:

1. We have investigated several methods for accelerating and scaling submodular data selection to large data sets using approximate methods. Our results have shown that these methods still result in good performance in practice, yielding improvements over established baseline methods for data selection. Using approximate methods, it has been possible, for the first time, to conduct submodular data selection for language modeling on terabytes of data.
2. We have investigated additional types of features for use in feature-based submodular function for data selection. These included hypothesized target-language features, confidence values indicating the reliability of a sentence pair in the training set, and parse-based features. While the first two did not provide any benefit, parse-based features resulted in small but consistent gains in a syntax-based tree-to-string translation model. The drawback of adding additional feature types is the increase in the dimensionality of the feature set. Initial work on reducing the dimensionality using feature hashing

techniques resulted in a loss in performance; alternative dimensionality reduction techniques will need to be addressed in future work.

3. Submodular feature selection was applied to the problem of reducing the number of sparse features in SMT systems. It was possible to reduce sparse feature sets substantially without a loss in MT performance. In some cases, the reduction of the feature set also resulted in better MT performance.
4. Two theoretical submodular objectives were developed for the problem of pruning phrase tables in SMT systems; however, the optimization algorithms required for these objectives turned out to be too computationally complex (e.g. involving repeated forced decoding of the phrase table). By contrast, a different phrase table pruning method was developed which simply integrates a relevance score expressing dependencies between different phrases in the phrase table into the scoring function for each phrase pair; phrase table subsets are then selected using simple modular selection. This method is easily scalable and showed statistically significant improvements over standard phrase table pruning methods.
5. We have applied submodular data selection to the task of large-scale language modeling, using the more recently developed recurrent neural language models in addition to backoff n-gram models. Submodular data selection led to significant improvements in model perplexity compared to cross-entropy based data selection and resulted in models that improved the performance of an SMT system in a second-pass rescoring procedure.
6. Finally, we have laid the groundwork for a new theoretical approach to data selection that consists of a two-step selection procedure where relevance and redundancy are optimized separately. Initial experimental investigations were inconclusive; further work is need to fine-tune and evaluate this method.

Future Work

A major trend characterizing current language and speech processing is the move towards neural modeling techniques. This includes recurrent neural networks (RNNs) or long-short term memory (LSTM) models for language modeling and speech recognition; neural encoder-decoder models for machine translation, or convolutional neural networks for whole-sentence modeling. In spite of our promising initial results described in Section 6. Language Modeling, it is still unclear exactly how these models are affected by the choice of training data, and how model properties can be taken into account during the subselection of training data. Future work will therefore analyze submodular data subselection methods based on functions that directly incorporate properties of the model to be trained on the subsets. Another potential application for submodularity is to prune large neural networks: it is often the case that neural networks, once trained to the desired level of performance, can be pruned for test purposes without a loss in performance. A submodular pruning criterion might allow models to be pruned down to even smaller sizes. Finally, another possible direction for future work is to address parallel training of neural networks and other models from the point of view of submodularity: for large data sets, training data is often partitioned into parallel chunks that are then processed in parallel, before combining partial parameter estimates into the final model. This model of training is rapidly being adopted for neural models in language processing when subset selection is not desired. Submodular optimization could help define better ways of partitioning the data.

Acronym List

EMEA	European Medicines Agency
GRU	Gated recurrent unit
IARPA	Intelligence Advanced Research Projects Agency
LDC	Linguistic Data Consortium
LM	Language model
LSTM	Long short-term memory
MERT	Minimum error rate training
MT	Machine translation
NIST	National Institute of Standards and Technology
RNN	Recurrent neural network
RNNLM	Recurrent neural network language model
SMT	Statistical machine translation
UMLS	Unified Medical Language System

References

V. Ambati, A. Lavie and J. Carbonell, "Extraction of Syntactic Translation Models from Parallel Data using Syntax from Source and Target Languages", *Proceedings of the Twelfth Machine Translation Summit (MT Summit XII)*, 2009

N. Bertoldi and M. Federico, "Domain Adaptation for Statistical Machine Translation with Monolingual Resources", *Proceedings of the EACL 4th Workshop on Machine Translation*, 2009

D. Chiang, K. Knight, and W. Wang, "11,001 New Features for Statistical Machine Translation", *Proceedings of HLT/ NAACL*, 2009, pp. 218-226

F. Chierichetti, R. Kumar, and A. Tomkins. Max-cover in Map-Reduce. *Proceedings of WWW*, 2010

J. Edmonds, "Submodular functions, matroids and certain polyhedral", in *Combinatorial Structures and their Applications*, Gordon & Breach, 1970, pp. 69-87

S. Fujishige, "Submodular functions and optimization", *Annals of Discrete Mathematics 47*, 1991

S. Green, S. Wang, D. Cer and C.D. Manning, "Fast and adaptive online training of feature-rich translation models", *Proceedings of ACL*, 2013

M. Hopkins and J. Kuhn, "Machine Translation as Tree Labeling", *Proceedings of SSST, NAACL-HLT 2007 / AMTA Workshop on Syntax and Structure in Statistical Translation*, 2007

A. Irvine and C. Callison-Burch, "Hallucinating Phrase Translations for Low Resource MT", *Proceedings of CONLL*, 2014

R. Iyer and J. Bilmes, "Algorithms for Approximate Minimization of the Difference between Submodular Functions", *Proceedings of Uncertainty in Artificial Intelligence* (2012)

H. Johnson, J. Martin, G. Foster and R. Kuhn, "Improving Translation Quality by Discarding Most of the Phrasetable", *Proceedings of EMNLP/CoNLL* 2007, pp. 967-975

K. Kirchhoff, Y. Liu and J. Bilmes, "Classification of developmental disorders from speech using submodular feature selection", *Proceedings of Interspeech*, 2013, pp. 187-190

K. Kirchhoff, J. Bilmes, K. Wei, Y. Liu, A. Mandal, C. Bartels, "A Submodularity Framework for Data Subset Selection", Technical Report AFRL-RH-WP-TR-2013-0108, September 2013

K. Kirchhoff and J. Bilmes, “Submodularity for Data Selection in Machine Translation”, *Proceedings of EMNLP*, 2014, pp. 131-141

K. Kirchhoff, Y.C. Tam, C. Richey and W. Wang, "Morphological Modeling for Machine Translation of English-Iraqi Arabic Spoken Dialogs", *Proceedings of NAACL*, 2015

R. Kumar, B. Moseley, S. Vassilvitskii, and A. Vattani. Fast greedy algorithms in MapReduce and streaming. *Proceedings of SPAA*, 2013

W. Ling, J. Gracia, Isabel Trancoso and A. Black, “Entropy-based pruning for phrase-based machine translation”, *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012, pp. 962–971

W. Ling, N. Tomeh, X. Guang, A. Black and I. Trancoso, “Improving Relative-Entropy Pruning using Statistical Significance”, *Proceedings of the 25th International Conference on Computational Linguistics (Coling 2012)*, 2012b

Y. Liu, K. Wei, K. Kirchhoff, Y. Song and J. Bilmes, “Submodular feature selection for high-dimensional acoustic score spaces”, *Proceedings of ICASSP*, 2013, pp. 7184-7188

B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause, Distributed submodular maximization: identifying representative elements in massive data. *Proceedings of Neural Information Processing Systems (NIPS)*, 2013

B. Mirzasoleiman, A. Badanidiyuru, A. Karbasi, J. Vondrak, and A. Krause, “Lazier than lazy greedy”, arXiv:1409.7938, 2014

R.C. Moore and W. Lewis, “Intelligent Selection of Language Model Training Data”, *Proceedings of ACL*, 2010, pp. 220-224

G. Neubig, “Travatar: A Forest-to-String Machine Translation Engine based on Tree Transducers”, *Proceedings of the ACL (Demonstration Track)*, 2013

D. Ravichandran, P. Pantel and E. Hovy, “Randomized Algorithms and NLP: Using Locality Sensitive Hash Functions for High Speed Noun Clustering”, *Proceedings of ACL*, 2005

L. Tian, D. F. Wong, L. S. Chao, P. Quaresma, F. Oliveira, S. Li, Y. Wang and Y. Lu, "UM-Corpus: A Large English-Chinese Parallel Corpus for Statistical Machine Translation", *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, 2014

M. Tomá, M. Karafiát, L. Burget, J. Èernocký and S. Khudanpur, ”Recurrent neural network based language model, In: *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2010

K. Wei and J. Bilmes, Fast multi-stage submodular maximization, *Proceedings of the International Conference on Machine Learning (ICML)*, 2014

X. Wu, T. Matsuzaki and J. Tsujii, “Fine-Grained Tree-to-String Translation Rule Extraction”, *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 2010

R. Zens, D. Stanton and P. Xu, “A Systematic Comparison of Phrase Table Pruning Techniques”, *Proceedings of EMNLP/CoNL*L, 2012

D. Zhang, M. Li, C.H. Li and M. Zhou, “Phrase Reordering Model Integrating Syntactic Knowledge for SMT”, *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007a

M. Zhang, H. Jiang, A. Aw, J. Sun. S. Li and C.L. Tan, “A Tree-to-Tree Alignment-based Model for Statistical Machine Translation”, *Proceedings of the MT Summit XI*, 20

LIST OF ACRONYMS & GLOSSARY

A3 File	The output of the GIZA++ word alignment program.
A3Metric	A collection of SCREAM Lab developed software tools to examine and modify results of word alignment.
ACL	Association for Computational Linguistics
Addicter	Automatic Detection and DIsplay of Common Translation Errors; A machine translation error analysis tool.
AFRICOM	U.S. Africa Command
AFRL	Air Force Research Laboratory
AFRL-H	A hierarchial machine translation system from AFRL SCREAM Lab.
AFRL-J	An AFRL SCREAM Lab machine translation system employing neural and statistical transliteration.
AFRL-K	A phrase-based machine translation system from AFRL SCREAM Lab.
AFRLv8 system	A phrase-based machine translation system employing rule-based transliteration of unknown words, from AFRL SCREAM Lab.
AMTA	Association for Machine Translation in the Americas
Aspell	An open-source spell checking utility.
ASR	automatic speech recognition
AWS	Amazon Web Services
BLAS	Basic Linear Algebra Subroutines
BLEU	Bilingual Evaluation Understudy; A metric used in machine translation scoring.
BPE	byte-pair encoding
C/C++	This notation refers to two compiled programming languages, C, and an extension of C, C++, both of which are suited to performing low-level machine instructions.
clang	A C language based front-end for the LLVM compiler.
CLUSTERGEN	A speech synthesizer for use with the Festival Speech Synthesis System.
CMU	Carnegie Mellon University
Common Crawl	An open repository of web crawl data.
CoNLL	Conference on Computational Natural Language Learning
CPAN	Comprehensive Perl Archive Network
CPU	central processing unit
CRF	conditional random field
CSLM	Continuous Space Language Model

CSTR	Centre for Speech Technology Research
CTB	Penn (University of Pennsylvania) Chinese Treebank
CUDA	A registered trademark of Nvidia, it refers to a programming interface for GPU computing.
DGEMV	A subroutine in the Nvidia CUDA BLAS Library that performs a particular matrix-vector operation.
DNN	deep neural network
DoD	Department of Defense
DTIC	Defense Technical Information Center
EDIN	University of Edinburgh
EMNLP	Empirical Methods in Natural Language Processing
Experiment Reader	A SCREAM Lab tool for comparing machine translation scoring results.
Farasa program	A word segmenter tool for Arabic.
Festival Speech Synthesis System	A free multi-lingual text-to-speech synthesis suite from University of Edinburgh.
FestVox	An initiative run by Carnegie Mellon University for advancing speech synthesis research.
Flite	Festival-lite; An open-source text-to-speech synthesizer from Carnegie Mellon University.
GATE	General Architecture for Text Engineering; An open-source suite of tools for natural language processing from the The University of Sheffield.
GIZA++	A word alignment tool, extended from its predecessor, GIZA, which originated from Johns Hopkins University.
GlottHMM	A Hidden Markov Model based speech synthesis system.
Google Tesseract	An open-source optical character recognition engine from Google from University of Helsinki Department of Speech Sciences.
GPGPU	general purpose graphics processing unit
GPU	graphics processing unit
gzip	GNU zip; A file compression utility.
Haystack	A SCREAM Lab computing testbed environment facilitating I/O, visualization, processing, and evaluation of speech and machine translation technologies.
HDD	hard disk drive
HindEnCorp	A parallel Hindi-English corpus extracted from internet sources from the Institute of Formal and Applied Linguistics Charles University, Czech Republic.

HindMonoCorp	A monolingual corpus of Hindi extracted from internet sources from the Institute of Formal and Applied Linguistics Charles University, Czech Republic.
Hjerson	An open-source error analysis program for machine translation, named after a fictional character in Agatha Christie books, from German Research Center for Artificial Intelligence (DFKI).
HLT	human language technology
HMM	Hidden Markov Model
HPC	high performance computing
HTK	HMM Toolkit
HTML	Hypertext Markup Language
HTS	HTK for Speech Synthesis
HTTP	Hypertext Transfer Protocol
I/O	input, output
IT	information technology
iBLEU	A machine translation scoring tool from Nitin Madnani.
ICER	Information Operations Cyber Exploitation Research
ID	identification
Intel	A manufacturer of computer processing hardware.
ISO	International Organization for Standardization
ISR	intelligence, surveillance, and reconnaissance
IWSLT	International Workshop on Spoken Language Translation
Jane	An open-source statistical machine translation system from RWTH Aachen University.
Java	A cross-platform object-oriented programming language, originally developed by Sun Microsystems.
JavaScript	A high-level scripting language with origins in web page development.
JHU	Johns Hopkins University
Joshua MT	An open-source statistical machine translation system from the Center for Language and Speech Processing at the Johns Hopkins University.
jQuery	A free, open source JavaScript library for dynamic update and control of web pages incorporating various features of client-side scripting.
LCTL	Less Commonly Taught Languages
LDC	Linguistic Data Consortium; An open consortium of universities, libraries, corporations and government research laboratories that is hosted by the University of Pennsylvania .

Levenshtein distance	A metric, named after Vladimir Levenshtein, for quantifying the dissimilariy between two sequences, such as words.
Linux	A UNIX-based operating system developed for use on personal computers.
LLVM	A code development project providing C++ libraries for a variety of uses.
LPC	Linear Prediction Coefficient
LSTM	long, short term memory model
LTO-3 tape	Linear Tape-Open; a magnetic storage media used for data backup.
LTP Segmenter	Language Technology Platform; A word segmenter module in the LTP Chinese suite of applications.
Malt Parser	A language-independent dependency parser tool from Johan Hall, Jens Nilsson, and Joakim Nivre.
Meteor	A machine translation evaluation and scoring tool from Michael Denkowski, Carnegie Mellon University.
MFCC	Mel Frequency Cepstrum Coefficient
MGIZA++	A word alignment tool from Qin Gao, based on GIZA++ with the addition of multi-threading capability.
MIT-LL	Massachusetts Institute of Technology (MIT) Lincoln Laboratory
MLSA	Mel Log Spectrum Approximation
mmap	A computer system call that maps file or device into memory.
Moses	An open-source statistical machine translation system.
MT	machine translation
MT-ComparEval	An open-source tool for evaluating and comparing machine translation outputs.
MWE	multi-word entity
MySQL	An open-source relational database management system.
Mystem	A morphological analyzer tool from Yandex, for use on Russian.
NASIC	National Air and Space Intelligence Center
NATO	North Atlantic Treaty Organization
NE	named entity
n-gram	A unit sequence of text or speech corpus used in language modeling.
NLP	natural language processing
NNJM	Neural Network Joint Model
NPLM	Neural Probabilistic Language Model
NVidia	A manufacturer of graphics processing unit hardware.
OCR	optical character recognition
OOV	out-of-vocabulary

Palladius	A system of transcribing Chinese names into cyrillic, created by Pyotr Ivanovich Kafarov.
Parsey McParseface	A pre-trained English parser included with TensorFlow SyntaxNet.
PDF	Portable Document Format
PER	position-independent word error rate
Perl	An interpreted programming language generally regarded for its regular expression and text processing capabilities.
PHP	Hypertext Preprocessor; An HTML-embedded scripting language.
PKU	Peking University Treebank
PLF	Python Lattice Format
Porter Stemmer	A stemming algorithm for English words, originating from Martin Porter.
POS	parts-of-speech
PowerPoint	A software tool for authoring and rendering briefing slides, created by Microsoft Corporation.
PPTX	PowerPoint Presentation file format
Pravda	An online Russian newspaper.
Qahira	A supervised word alignment editing tool.
QED	Qatar Computing Research Institute (QCRI) Educational Domain
Qualitative	A machine translation evaluation tool with hybrid machine translation capability, from German Research Center for Artificial Intelligence (DFKI).
Raytheon BBN	Raytheon BBN Technologies; A research and development center within Raytheon Company.
Reverse Palladius	A tool for ensuring proper Russian-to-English translation of Chinese names appearing in Russian text.
Revised Hjerson	A SCREAM Lab variant of the Hjerson error analysis program.
RevP	Shorthand for the AFRL *Reverse Palladius* tool.
RFTagger	An annotation tool from Helmut Schmid and Florian Laws.
RNNLM	Recurrent Neural Network Language Model
ROMIP	Russian Information Retrieval Evaluation Seminar
RT	Retweet; a construct of social media platform, *Twitter*.
SATA	Serial ATA; An interface to computer storage devices.
SCLITE	A speech recognition scoring tool in the National Institute of Standards and Technology (NIST) Speech Recognition Scoring Toolkit (SCTK).
SCREAM	Speech and Communication Research, Engineering, Analysis, and Modeling
SCREAMStemmer.java	A SCREAM Lab stemmer tool for use on Russian.

SIGHAN	Special Interest Group on Chinese Language Processing; An subordinate organization of the Association for Computational Linguistics.
SLF	Standard Lattice Format
SMT	statistical machine translation
sspell	“SONIC Spell”; A phonetic pronunciation extractor program within the SONIC speech recognition toolkit.
SONIC	A toolkit for speech recognition research from the The University of Colorado.
SPO	Special Program Office
Stanford Parser	A statistical natural language parser from Standford University.
Stanford Segmenter	A word segmenter from Stanford University.
Stemka	A morphological analyzer tool for use on Russian and Ukrainian.
SV	subject-verb
SVO	subject-verb-object
Systran	A commerial machine translation software package.
TED Talks	Technology, Entertainment, and Design; A series of conferences featuring speakers discussing a variety of topics and experiences in various languages.
TensorFlow SyntaxNet	An open-source neural network framework for the TensorFlow software library.
TreeTagger	A language-independent annotation tool from Helmut Schmid.
TriggerLM	Trigger-Based Lexicon Model
TweeboParser	A dependency parser for English tweets from Carnegie Mellon University.
URL	uniform resource locator
VarCon	Variant Conversion; a database for converting between American, British, Canadian, and Australian vocabularies.
VS	verb-subject
Wall Street Journal	Speech corpora datasets were created from news texts from the Wall Street Journal newspaper, available from the Linguistic Data Consortium.
WER	word error rate
Windows	A series of operating systems designed for personal computers from Microsoft Corp.
WUM	Work Unit Manager
WMT	Workshop on Statistical Machine Translation
XenC	An open-source data selection tool for use in natural language processing.
XHTML	Extensible Hypertext Markup Language

XML	Extensible Markup Language
Yandex	A Russian technology company and internet search engine provider.
ZFS	A file system by Oracle Corporation.
ZIP	A ubiquitous archive file format.