AFRL-RI-RS-TR-2019-063

DOMAIN-SPECIFIC INSIGHT GRAPHS

UNIVERSITY OF SOUTHERN CALIFORNIA

MARCH 2019

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

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 is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2019-063 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
EDWARD DEPALMA
Work Unit Manager

/ S /
TIMOTHY A. FARRELL
Deputy Chief, Information Intelligence
Systems and Analysis Division
Information Directorate

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

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
MARCH 2019	FINAL TECHNICAL REPORT	SEP 2014 – SEP 2018

4. TITLE AND SUBTITLE

DOMAIN-SPECIFIC INSIGHT GRAPHS

5a. CONTRACT NUMBER
FA8750-14-C-0240

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62702E

6. AUTHOR(S)

Pedro Szekely and Mayank Kejriwal

5d. PROJECT NUMBER
MEMX

5e. TASK NUMBER
00

5f. WORK UNIT NUMBER
10

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Southern California
4676 Admiralty Way # 1001
Marina del Rey, CA 90292

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RIEA
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2019-063

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Developing scalable, semi-automatic approaches to derive insights from a domain-specific Web corpus is a longstanding research problem in the knowledge discovery and web communities. The problem is particularly challenging in illicit fields, such as human trafficking, where traditional assumptions concerning information representation are frequently violated. In the Domain-Specific Insight Graphs project (DIG),we developed technology to build end-to-end investigative knowledge discovery and search systems, focused primarily on illicit Web domains. The technologies include components for information extraction, semantic modeling and query execution, and was tested in on a variety of real-world domains, including a human trafficking Web corpus containing over 100 million pages. The prototype includes a GUI that was used by US law enforcement agencies to combat illicit activity. The research results were widely disseminated in multiple publications in journals and conferences, and the software produced is publicly available on Github under the MIT license.

15. SUBJECT TERMS

Knowledge Graphs, Information Extractions, Web Search, Domain-Specific, Users Interface

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**EDWARD DEPALMA**
U	U	U	UU	88	19b. TELEPHONE NUMBER (*Include area code*) **N/A**

Table Of Contents

List of Figures

List of Tables

Listings

1 SUMMARY

Developing scalable, semi-automatic approaches to derive insights from a domain-specific Web corpus is a longstanding research problem in the knowledge discovery and web communities. The problem is particularly challenging in illicit fields, such as human trafficking, where traditional assumptions concerning information representation are frequently violated. In the Domain-Specific Insight Graphs project (DIG), we developed technology to build end-to-end investigative knowledge discovery and search systems, focused primarily on illicit Web domains. The technologies include components for information extraction, semantic modeling and query execution, and was tested in on a variety of real-world domains, including a human trafficking Web corpus containing over 100 million pages. The prototype includes a GUI that was used by US law enforcement agencies to combat illicit activity. The research results were widely disseminated in multiple publications in journals and conferences, and the software produced is publicly available on Github under the MIT license.

2 INTRODUCTION

The DARPA MEMEX program was established with the intent of building domain-specific search systems that could quickly generalize to an arbitrary domain. A domain in this context is a relatively diffuse term, as there may not be an explicit ontology describing it. A class of domains that is of interest to the MEMEX program is investigative in nature, whereby consumers of the technology are investigators and field analysts who are investigating illicit activity believed to have a significant Web footprint in terms of advertisements, transactions and reviews (or all of them).

In today's digital era, most people rely on search engines like Google and Bing for their needs. Like the smartphone, tablet and desktop, these engines are good examples of technology that are optimized for the general populace, with commercially oriented goals. Historically, and even presently, domains where building specialized technology for search and analytics are necessary tend to have massive commercial or military implications. Many agencies at the state and local levels, even in a developed country like the United States, cannot afford such proprietary, state-of-the-art technology in the present resource-strapped environment.

The search systems that have been built under MEMEX are designed to adapt in a very general, user-centric way to arbitrary domains by (1) taking sparse specifications of the domain from a user, (2) using these specifications in interactive focused crawling systems to scrape webpages that are likely relevant to fulfilling some subset of the user's informational needs, and (3) extracting useful structured information from the corpus of webpages and building an intuitive search and analytics engine over this constructed 'knowledge base'. The process, expected to be iterative, is refined over time as users invest more effort into the system.

Based on discussions with actual investigators, we can 'break down' investigative search queries into categories: lead generation and lead investigation. Taking Google as an analogy, lead generation is like clicking on the 'I feel lucky' button, or looking at trending topics. Lead investigation, which can be evaluated in more concrete ways, starts when the user already has a lead in hand. This lead may come from a tip-off on the ground, a suspicious statement or arrest record, or a matter of public record, like the name of a person who has recently gone missing. Lead investigation questions can be classed into one of four categories (using DARPA and NIST terminology): point

fact, cluster facet, cluster identification and cluster aggregate.

Although not obvious, the lead investigation questions do provide some intuition into why generic search engines like Google are not adequate for satisfying such information needs (at least not without significant engineering). Two problems, information obfuscation and ambiguity, are respectively related to the quality of the information in the ads and to the difficulty of automating a longstanding Artificial Intelligence (AI) problem called Information Extraction (IE). Concerning the former, online sex ads are written with the intent of being indexed by certain attributes (mainly location and the kind of sex service being advertised), but other attributes (phone number, social media ids and addresses) are deliberately obfuscated in creative ways so that they are human (but not machine) readable. For example, the following text fragment "AVAILABLE NOW! ?? - (1 two 1) six 5 six - 0 9 one 2" contains th phone number (121) 656-0912 that a human would be able to recognize and dial, but an IE system (also, Google) would have a hard time extracting and indexing in a database, unless it was extensively tuned or engineered. Similarly, ambiguity arises when there is confusion about whether a word (e.g., Charlotte) links to the city in North Carolina or a person. Humans use context to make this determination, and while there have been significant developments in machine reading methods, overall accuracy is still quite low, especially for 'difficult' domains like human trafficking [12],[13].

Even without obfuscation and ambiguity, the cluster and aggregate questions illustrate another kind of facility not offered by Google. These questions require a user to retrieve a set of related ads e.g. via a shared phone number1 (for the cluster questions), and also to aggregate quantities across the set. This is a task that involves both search and analytics currently not included in the Google interface.

Unlike lead investigation questions, lead generation questions were significantly more controversial, as users were divided on what makes for a useful (or more importantly, a not harmful) lead generation question, and even on whether explicitly incorporating lead generation into a domain-specific search system was warranted or the best use of resources. This was an eye-opening insight that significantly changed the way we approached our efforts in the final year of the MEMEX program (2017).

In the initial phase of the program, one of the task challenges that technical performers were called upon to participate in was that of predicting whether a given set of scraped sex activity ads (each set representing a 'case') indicated some level of trafficking. It was believed that this was a useful problem, and that an imputed 'risk score' (similar to a credit score) for such a set could be used to generate leads. Many investigators who were told about this task challenge felt that the problem was misguided, if not impossible to solve, for several reasons. One important reason is that the ad often does not contain enough information for such predictions, and algorithms may end up being biased in favor of certain ad characteristics (e.g., racy words in the text). Some investigators also feel that they already have enough leads to investigate, and that they are not looking to generate more leads. Others felt that they could not trust the outputs of a machine learning system without some explanation.

This report describes Domain-Specific Insight Graphs (DIG), the technology developed at the USC Information Sciences Institute to support the search capabilities described previously. DIG is a complex ensemble of various adaptive technologies that is ultimately designed to help users perform investigative search. The system was optimized for investigative querying in the online sex trafficking domain, although it is currently also being extended to address search in arbitrary domains. DIG includes:

- A suite of IE technologies that use rules, heuristics and advanced machine learning to automatically extract a variety of relevant structured attributes from webpages, including phones, emails, names, ages, physical attributes, sex services, and locations. We refer to this collective set of interlinked extractions as a knowledge graph.
- Indexing techniques for ensuring fast retrieval over both the knowledge graph and the text in the webpages .
- A cached copy of every processed webpage, in case the webpage is taken offline on the live Web (as it often is), and also to foster trust in, or verify, IE.
- Provenance data about which algorithm led to which extraction in the knowledge graph.
- Images and image similarity search based on deep neural network-based computer vision.
- Big Data architecture to support streaming ingestion of new ads.
- A user interface to enable investigators to search the knowledge graph and visualize it from multiple perspectives.

3 METHODS, ASSUMPTIONS AND PROCEDURES

3.1 Information Extraction

Table 1 provides an overview of the extraction technology used in DIG. In this report, we discuss the Inferlink tool, focused on extraction from short-tailed domains, the NLP rule editor focused on long-tailed domains, and work to improve the precision of all extractions. The other algorithms described in Table 1 are widely used in the *natural language* IE community; we provide relevant references that guided the design of those algorithms. We note that, except for the Inferlink tool, the semantic types of extractions output by the other algorithms are pre-determined. The readability text extractor (RTE) is an off-the-shelf text scraper that takes HTML as input and outputs a text sequence. Its hyperparameters can be tuned to yield either high recall or high precision[1]. The remaining extractors individually process the high-precision and high-recall text output by RTE and yield a set of extractions each with the corresponding metadata. Each extractor requires a different level of manual effort. For example, the NLP rule-based extractor can use hand-crafted rules that use NLP features (POS tags etc.) to extract street addresses from the text. Ideally, if large quantities of training data had been available, a Conditional Random Field (CRF) could be used, such as for eye-color and hair-color attributes. Unfortunately, street addresses are extremely sparse and irregular in human trafficking data. On the other hand, off-the-shelf street address extractors also yielded extremely noisy performance.

3.1.1. Inferlink Extractor

The *Inferlink extractor* is a *wrapper-based* semi-supervised tool that takes a collection of HTML pages from a top-level Web domain as input [1], [42]. Wrappers are common in Web-based

[1]For the latter, the text may be much 'cleaner' but could potentially be missing useful sentences and paragraphs.

Table 1: Overview of the knowledge graph construction (KGC) extraction technology ontologically guided by the investigative schema.

Extraction Technology	**Investigative Attributes**	**Level of Effort/Engineering**	**Relevant References**
Inferlink (Template clustering+wrapper-induced rules)	All 'structured' attributes	10-20 minutes trained expertise per Web domain (e.g. backpage.com)	[42], [35], [1]
Readability Text Extractor	Text	Hyperparameter tuning	[3]
Conditional Random Fields	Eye-color, Hair-color	Labeled data, feature engineering, hyperparameter tuning	[47], [55]
Dictionaries and Entity Sets+contextual classification using word embedding features	Name, Location (City, State, Country), Nationality, Ethnicity, Service	Procuring dictionaries, large text corpus, small set of labeled annotations/attribute	[53], [17], [30], [32]
Regular Expressions and Custom Programs	Email, Height, Phone, Posting-date, Price, Review-id, Social-media-id, Title, Weight	Programming regular expressions	[43], [34]
NLP rule	Street-address	Crafting the rule	[18], [8]

information extraction systems. Inferlink is a wrapper-based tool that operates in several steps [1]. All steps can potentially be unsupervised, but for best performance, manual intervention from an expert is required in the last stages.

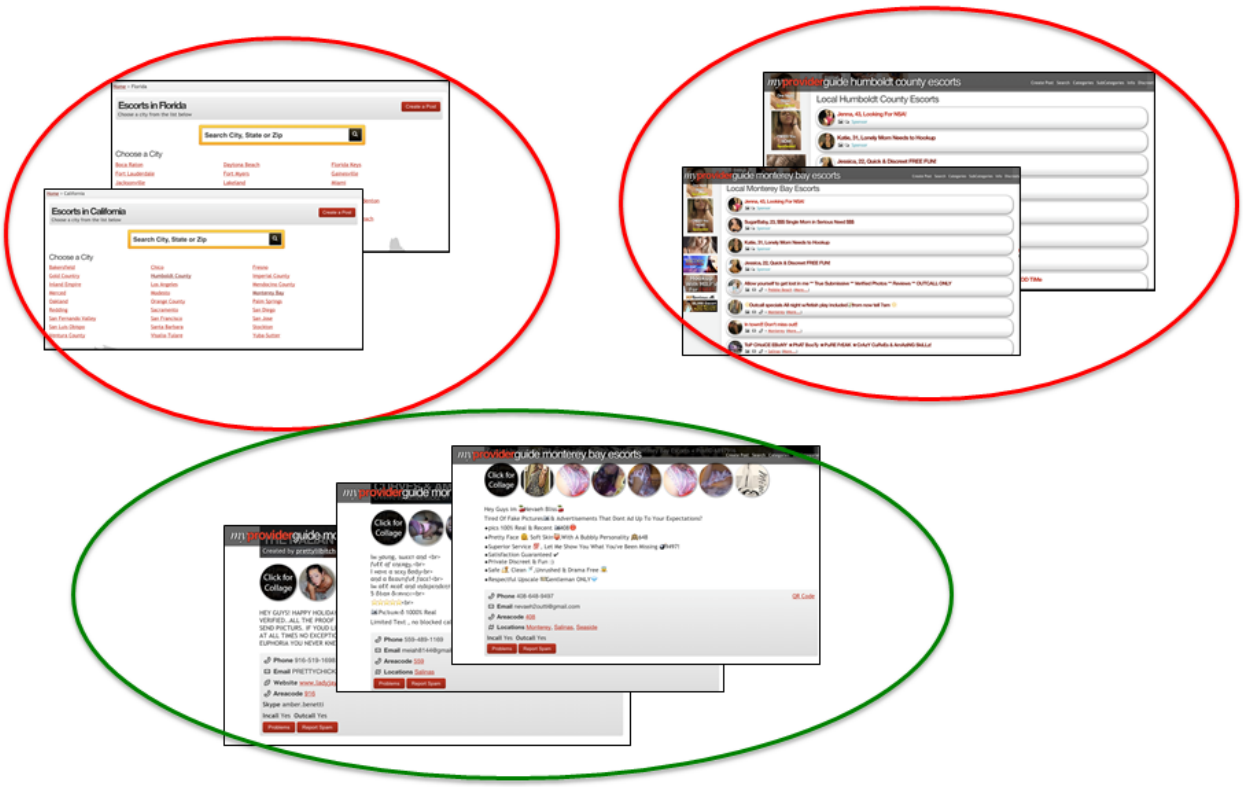

Figure 1: The unsupervised "template clustering" step in the Inferlink information extractor. The top two clusters provide listings and category enumerations and are irrelevant for either search or extraction. The bottom cluster contains ads from which *structured attributes* can be extracted and *semantically typed* in terms of investigative schema attributes.

In a first step, the extractor clusters the pages in an unsupervised fashion using structural information such as the HTML templates. An example output from template clustering is illustrated in Figure 1. Once the clusters are obtained, an expert familiar with the tool chooses the *semantically relevant* cluster (In the figure, this would be the bottom cluster) and applies an unsupervised wrapper to the cluster. By semantically relevant, we mean that pages in the cluster, though similar *structurally*[2], are not useful for the purposes of either KGC or search as they do not directly describe escorts. The wrapper is unsupervised because it is able to use the similarities and differences between pages in the cluster to extract 'structured' attributes. In the final step, which may be manual or automatic, each attribute must be semantically typed in terms of the correct attribute from the investigative schema. Because there are thousands of Web domains in the HT subject domain, and the limited time accorded to the expert per domain, we only use the Inferlink extractor on the largest Web domains. Semantic typing is manual, guaranteeing the precision of an Inferlink extraction, when obtained for a given webpage.

[2]In that sense, the cluster is 'correct'.

The clustering is designed to be both robust and unsupervised; this is why users already see the clusters (with sample pages) when they first open a TLD in Inferlink. Once a cluster is picked by the user as being relevant, the wrapper algorithms in Inferlink extract structured elements from each HTML page in the cluster, and show them to a user in a column layout, as illustrated in Figure 1. Other elements of user experience, including assigning field names to columns (denoted as the semantic typing step) were described earlier. The final output of Inferlink processing (per TLD) is a set of *machine-processable rules* that include automatically generated regular expressions that identify the beginning and end of each extraction. These rules are executed on a server on every webpage (from that TLD), including webpages not belonging to the initial training cluster that the user selected and curated. Heuristics are used to identify incorrect extractions from 'out of cluster' pages. An important advantage of the rule-based execution is that, once the rules are stored on a server, they can be *re-used* across user sessions or even *shared* by different sets of users. Rule execution is also computationally efficient: while the clustering and rule inference steps are expensive (5 minutes/TLD), even complex Inferlink rules-sets are able to execute in milliseconds per webpage. Inferlink continues to be actively maintained by a private company.

Figure 2: Input sentences from the online sex advertisement domain. Named entities were incorrectly extracted by pre-trained modules from SpaCy due to the unusual nature of the domain.

3.1.2. Rule-Based Extractor

An option for creating extractors for domain-specific entitties is for experts to look at examples from the domain, and specify sets of rules that are both interpretable and that experts can 'play' with. Since the expert usually has no programming expertise, such rules must be specifiable in an intuitive manner, and be amenable to example-based 'trial and error'. An added benefit arises when such rules can be combined with other extraction modules, such as a pre-trained machine learning-based named entity recognizer, or a Semantic Web resource like GeoNames[3].

In DIG we built a GUI-based rule specification system that allows users to write expressive and intuitive rule-sets, including rules not currently allowable in NLP packages like SpaCy [4], without any knowledge of either programming or regular expressions. A visualization of the system dashboard is produced in Figure 3. To ensure fast, correct and scalable execution, the system compiles each rule in the rule-set into a set of SpaCy rules that can be executed in the

[3] http://www.geonames.org/

backend to produce sets of extractions from raw data. Additionally, we have integrated the rule editor and compiler with the DIG architecture [33], [31], which permits (1) combining extractions from the system with extractions from more advanced modules like Conditional Random Fields, (2) recording extraction provenance, which allows users to view the impact of their specifications on overall quality. Concerning real-world usage, the system has already been used for fulfilling a variety of domain-specific KGC needs, including extracting attributes (e.g., phone numbers) in the online sex ad domain (to assist investigators track down leads), securities fraud, illegal weapons sales and 'ordinary' domains like museums.

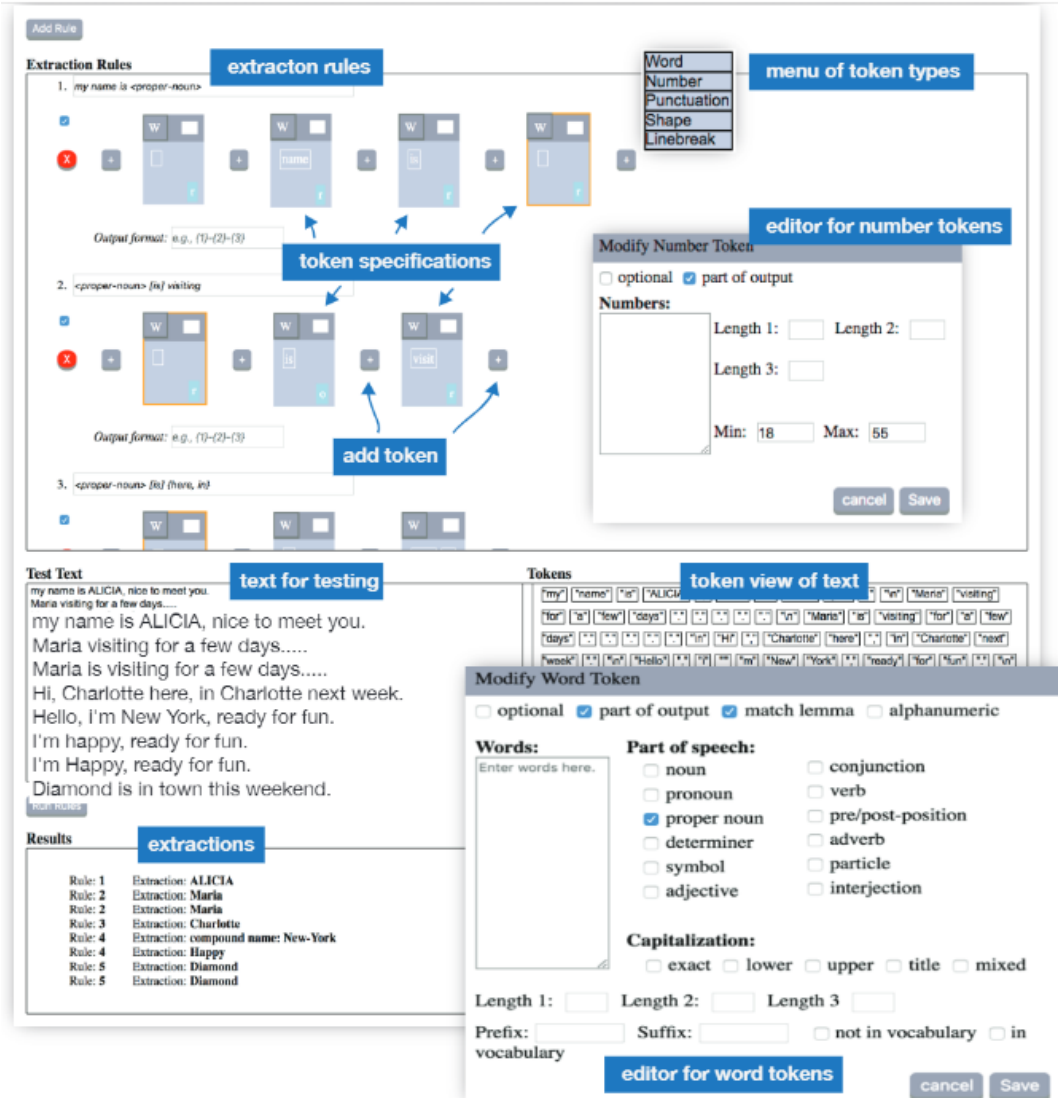

Figure 3: (Underlaid) The rule specification dashboard. (Overlaid) Configuration options for an atom that can be included in a larger rule template.

System and User Experience The example in Figure 2 illustrates one possible user experience whereby the user wants to extract 'common' named entities like names or locations, but is unable

Figure 4: Sentences containing uncommon (but useful to experts) attributes such as phone number and age that need to be extracted.

to rely on standard packages like SpaCy or Stanford NER due to the unusual nature of the domain (in this case, human trafficking). Instead, the user starts by looking at some example sentences, and forming rule templates using the 'template atoms' (also called token specifications) shown in Figure 3. For example, the topmost rule template, labeled self-explanatorily as 'my name is <proper-noun>' is composed of four atoms. Each atom is highly configurable, and can simply be constants (such as 'name'), or NLP artefacts like particles or adverbs, or regular expression artefacts like alphanumeric sequences (overlaid portion of Figure 3).

A second scenario arises when the attribute (and potentially, the domain) is uncommon in terms of publicly available tools being able to extract them from raw data (Figure 4). Because of the regular syntactic nature of attributes like telephone numbers or ages, as well as contextual regularities in the surrounding text, users are often able to get good coverage by specifying rules.

More specifically, a rule-based extractor for an entity consists of a collection of rules. Each rule targets extraction in a specific context with high precision. Multiple rules are used to extract an entity in multiple contexts. The extractions of a collection of rules are the union of the extractions of each rule, excluding extractions whose extent is completely contained within the extent of other extractions.

The input to the rule extractor is a sequence of tokens. Each rule consists of a sequence of token specifications. Each token specification defines a Boolean function $f(token)$. A rule is said to match a sequence of tokens if each token specification evaluates to true for each token in the sequence. The extraction of a rule on a matched sequence of tokens is the set of tokens matched by token specifications that have been marked as output tokens (orange border in Figure 3). By default the output is formatted as a string by concatenating the output tokens separated by space. Users can customize the output using a simple template language.

Token specifications can be marked as optional. When token specifications are optional, the rule editor behaves as if two rules had been defined, one with the optional token and one without the optional token. When a rule contains multiple optional tokens, the cross-product of all possible rules is considered. The extractions of rules with optional consist of the union of all maximal length rules in the cross-product.

The DIG rule editor is designed to extract entities defined using letters, numbers and symbols. Examples include entities such as person, organization and locations, extracted in traditional NER tools, as well as entities with specific syntax such as phone numbers and email addresses. The default DIG tokenizer defines any non-alphanumeric character as a separate token. Users can also customize the tokenizer. The five different token specifications currently supported by the system are enumerated in Table 2.

Table 2: Token specifications supported by the rule editor.

Token type	Description
Word tokens	match tokens consisting of a sequence of alphanumeric characters (see Figure 3). The word-token specification can be defined explicitly (by listing specific tokens) and implicitly (based on token characteristics) such as NLP features like POS tags and lemmatization, membership in a vocabulary e.g., English word, and syntactic features such as capitalization, length, prefix and suffix).
Number tokens	match tokens consisting of a sequence of digits (see Figure 3), and can also be defined explicitly by listing specific numbers to match, or implicitly based on range and number of digits.
Punctuation tokens	select a subset of punctuation symbols.
Shape tokens	match tokens consisting of a sequence of alphanumeric characters. Shapes can be defined explicitly using sequences of characters "X", "x" and "d", where "X" matches an upper-case letter, "x" matches a lower-case letter and "d" matches a digit. Shape tokens can also be defined implicitly using part of speech tags, a prefix and a suffix.
Line-break tokens	match tokens that break text into multiple lines and are defined by a minimum and maximum number of line-break characters.

Implementation and Applications The rule extractor is implemented as an extension to the SpaCy rule extractor, which offers similar, but simpler token specifications. Each DIG token specification is mapped to a set of SpaCy token specifications that collectively match a superset of the tokens that should be matched by the DIG token specification. Each DIG rule is implemented by the collection of SpaCy rules resulting from the creation of a cross-product of the SpaCy token specifications in the each collection. A post-processing step removes redundant matches for those cases when the collection of SpaCy token specifications matches a superset of the tokens that should be matched by the DIG token specification. For example, SpaCy does not support minimum and maximum limits for number tokens. In this case, the SpaCy rule matches arbitrary numbers, and the post-processing step removes matches for numbers outside the desired range. A further post-processing step removes that are strict subsets of matches produced by other SpaCy rules in the collection of all SpaCy rules generated from a single DIG rule.

The software implementing the DIG rule editor is available on GitHub. The processor[4] and user interface[5] are both publicly available. The full system featured in this demonstration can also be downloaded[6]. All software carries an MIT license.

Concerning real-world applications, the DIG rule editor has been used to define extractors for a variety of entities, including phone numbers for most countries in Europe and North and South America, email addresses, ages, person names, dates, massage parlor names, stock tickers, addresses, social media handles, artist names, as well as a variety of specific extractors for specific firearm classified ads and penny stock promotion web sites.

[4] https://github.com/usc-isi-i2/etk/blob/master/etk/spacy_extractors/customized_extractor.py

[5] https://github.com/usc-isi-i2/spacy-ui

[6] https://github.com/usc-isi-i2/dig-etl-engine

3.1.3. Improving The Precision Of Extractions

Extracting useful entities and attribute values from illicit domains such as human trafficking is a challenging problem with the potential for widespread social impact. Such domains employ atypical language models, have 'long tails' and suffer from the problem of concept drift. We developed a lightweight, feature-agnostic Information Extraction (IE) paradigm specifically designed for such domains. Our approach uses raw, unlabeled text from an initial corpus, and a few (12-120) seed annotations per domain-specific attribute, to learn robust IE models for unobserved pages and websites. Empirically, we demonstrate that our approach can outperform feature-centric Conditional Random Field baselines by over 18% F-Measure on five annotated sets of real-world human trafficking datasets in both low-supervision and high-supervision settings. We also show that our approach is demonstrably robust to concept drift, and can be efficiently bootstrapped even in a serial computing environment.

As real-world illustrative examples, consider the text fragments *'Hey gentleman im neWYOrk and i'm looking for generous...'* and *'AVAILABLE NOW! ?? - (4 two 4) six 5 two - 0 9 three 1 - 21'*. In the first instance, the correct extraction for a *Name* attribute is *neWYOrk*, while in the second instance, the correct extraction for an *Age* attribute is *21*. It is not obvious what features should be engineered in a statistical learning-based IE system to achieve robust performance on such text.

To compound the problem, *wrapper induction* systems from the Web IE literature cannot always be applied in such domains, as many important attributes can only be found in text descriptions, rather than template-based Web extractors that wrappers traditionally rely on. Constructing an IE system that is robust to these problems is an important first step in delivering structured knowledge bases to investigators and domain experts.

There are two main technical challenges that such domains present to IE systems. First, as the brief examples above illustrate, feature engineering in such domains is difficult, mainly due to the atypical (and varying) representation of information. Second, investigators and domain experts require a *lightweight* system that can be quickly bootstrapped. Such a system must be able to generalize from few ($\approx$10-150) manual annotations, but be incremental from an engineering perspective, especially since a given illicit Web page can quickly (i.e. within hours) become obsolete in the real world, and the search for leads and information is always ongoing. In effect, the system should be designed for streaming data.

Our information extraction approach that is able to address the challenges above, especially the variance between Web pages and the small training set per attribute, by combining two sequential techniques in a novel paradigm. The overall approach is illustrated in Figure 5. First, a *high-recall recognizer*, which could range from an exhaustive Linked Data source like GeoNames (e.g. for extracting locations) to a simple regular expression (e.g. for extracting ages), is applied to each page in the corpus to derive a set of *candidate annotations* for an attribute per page. In the second step, we train and apply a supervised feature-agnostic classification algorithm, based on learning word representations from random projections, to classify each candidate as correct/incorrect for its attribute.

Approach Figure 5 illustrates the architecture of our approach. The input is a Web corpus containing relevant pages from the domain of interest, and *high-recall recognizers* typically adapted from freely available Web resources like Github and GeoNames. In keeping with the goals of this work, we do not assume that this initial corpus is static. That is, following an initial short set-up

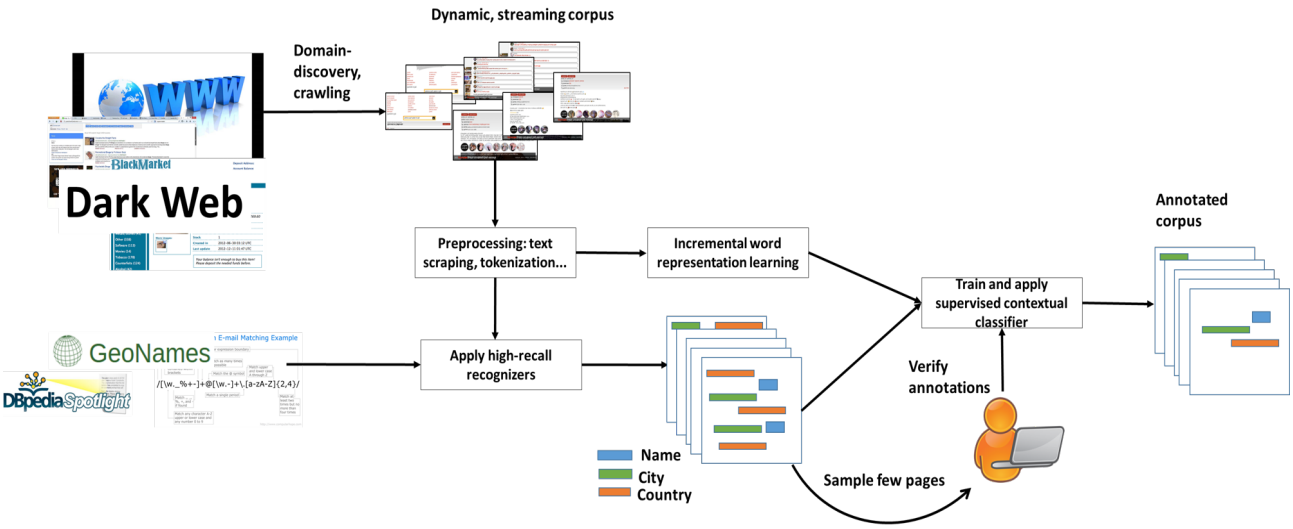

Figure 5: A high-level overview of the proposed information extraction approach

phase, more pages are expected to be added to the corpus in a streaming fashion. Given a set of pre-defined attributes (e.g. City, Name, Age) and around 10-100 manually verified annotations for each attribute, the goal is to learn an IE model that accurately extracts attribute values from each page in the corpus without relying on expert feature engineering. Importantly, while the pages are single-*domain* (e.g. human trafficking) they are *multi-Web domain*, meaning that the system must not only handle pages from new websites as they are added to the corpus, but also *concept drift* in the new pages compared to the initial corpus.

Preprocessing The first module in Figure 5 is an automated pre-processing algorithm that takes as input a streaming set of HTML pages. In real-world illicit domains, the key information of interest to investigators (e.g. names and ages) typically occurs either in the *text* or the *title* of the page, not the template of the website. Even when the information occasionally occurs in a template, it must be appropriately disambiguated to be useful[7]. Wrapper-based IE systems [36] are often inapplicable as a result. As a first step in building a more suitable IE model, we scrape the text from each HTML website by using a publicly available text extractor called the *Readability Text Extractor*[8] (RTE). Although multiple tools[9] are available for text extraction from HTML [26], our early trials showed that RTE is particularly suitable for noisy Web domains, owing to its tuneability, robustness and support for developers. We tune RTE to achieve *high recall*, thus ensuring that the relevant text in the page is captured in the scraped text with high probability. Note that, because of the varied structure of websites, such a setting also introduces noise in the scraped text (e.g. wayward HTML tags). Furthermore, unlike natural language documents, scraped text can contain many irrelevant numbers, Unicode and punctuation characters, and may not be regular. Because of the presence of numerous tab and newline markers, there is no obvious natural language *sentence*

[7]For example, 'Virginia' in South Africa vs. 'Virginia' in the US.

[8]`https://www.readability.com/developers/api`

[9]An informal comparison may be accessed at `https://www.diffbot.com/benefits/comparison/`

structure in the scraped text[10]. In the most general case, we found that RTE returned a set of strings, with each string corresponding to a set of sentences.

To serialize the scraped text as a list of tokens, we use the word and sentence tokenizers from the NLTK package on each RTE string output [8]. We apply the sentence tokenizer first, and to each sentence returned (which often does not correspond to an actual sentence due to rampant use of extraneous punctuation characters) by the sentence tokenizer, we apply the standard NLTK word tokenizer. The final output of this process is a list of tokens. In the rest of this section, this list of tokens is assumed as representing the HTML page from which the requisite attribute values need to be extracted.

Deriving Word Representations In principle, given some annotated data, a sequence labeling model like a Conditional Random Field (CRF) can be trained and applied on each block of scraped text to extract values for each attribute [47], [22]. In practice, as we empirically demonstrate in our evaluations, CRFs prove to be problematic for illicit domains. First, the size of the training data available for each CRF is relatively small, and because of the nature of illicit domains, methods like distant supervision or crowdsourcing cannot be used in an obvious timely manner to elicit annotations from users. A second problem with CRFs, and other traditional machine learning models, is the careful feature engineering that is required for good performance. With small amounts of training data, good features are essential for generalization. In the case of illicit domains, it is not always clear what features are appropriate for a given attribute. Even common features like capitalization can be misleading, as there are many capitalized words in the text that are not of interest (and vice versa).

To alleviate feature engineering and manual annotation effort, we leverage the entire raw corpus in our model learning phase, rather than just the pages that have been annotated. Specifically, we use an unsupervised algorithm to represent each word in the corpus in a low-dimensional *vector space*. Several algorithms exist in the literature for deriving such representations, including neural embedding algorithms such as Word2vec [48] and the algorithm by Bollegala et al. [11], as well as simpler alternatives [61].

Given the dynamic nature of streaming illicit-domain data, and the numerous word representation learning algorithms in the literature, we adapted the *random indexing* (RI) algorithm for deriving contextual word representations [61]. Random indexing methods mathematically rely on the Johnson-Lindenstrauss Lemma, which states that if points in a vector space are of sufficiently high dimension, then they may be projected into a suitable lower-dimensional space in a way which approximately preserves the distances between the points.

The original random indexing algorithm was designed for incremental dimensionality reduction and text mining applications. We adapt this algorithm for learning word representations in illicit domains. Before describing these adaptations, we define some key concepts below.

Definition 1: Given parameters $d \in \mathbb{Z}^+$ and $r \in [0, 1]$, a *context vector* is defined as a $d-$dimensional vector, of which exactly $\lfloor dr \rfloor$ elements are randomly set to $+1$, exactly $\lfloor dr \rfloor$ elements are randomly set to -1 and the remaining $d - 2\lfloor dr \rfloor$ elements are set to 0.

We denote the parameters d and r in the definition above as the *dimension* and *sparsity ratio* parameters respectively.

[10]We also found sentence ambiguity in the actual text displayed on the browser-rendered website (in a few human trafficking sample pages), due to the language models employed in these pages.

Figure 6: An example illustrating the naive Random Indexing algorithm with unigram atomic units and a $(2, 2)$-context window as context

Intuitively, a context vector is defined for every *atomic unit* in the corpus. Let us denote the universe of atomic units as U, assumed to be a partially observed countably infinite set. In the current scenario, every unigram (a single 'token') in the dataset is considered an atomic unit. Extending the definition to also include higher-order ngrams is straightforward, but was found to be unnecessary in our early empirical investigations. The universe is only partially observed because of the incompleteness (i.e. streaming, dynamic nature) of the initial corpus.

The actual vector space representation of an atomic unit is derived by defining an appropriate *context* for the unit. Formally, a context is an abstract notion that is used for assigning *distributional semantics* to the atomic unit. The distributional semantics hypothesis (also called *Firth's axiom*) states that the semantics of an atomic unit (e.g. a word) is defined by the contexts in which it occurs [41].

In this work, we only consider *short contexts* appropriate for noisy streaming data. In this vein, we define the notion of a (u, v)-context window below:

Definition 2: Given a list t of atomic units and an integer position $0 < i \leq |t|$, a (u, v)-context window is defined by the set $S - t[i]$, where S is the set of atomic units inclusively spanning positions $max(i - u, 1)$ and $min(i + v, |t|)$

Using just these two definitions, a naive version of the RI algorithm is illustrated in Figure 6 for the sentence 'the cow jumped over the moon', assuming a $(2, 2)$-context window and unigrams as atomic units. For each *new* word encountered by the algorithm, a context vector (Definition 1) is randomly generated, and the representation vector for the word is initialized to the 0 vector. Once generated, the context vector for the word remains fixed, but the representation vector is updated with each occurrence of the word.

The update happens as follows. Given the context of the word (ranging from a set of 2-4 words), an *aggregation* is first performed on the corresponding context vectors. In Figure 6, for example, the aggregation is an *unweighted* sum. Using the aggregated vector (denoted by the symbol $\vec{a}$), we update the representation vector using the equation below, with $\vec{w}_i$ being the representation vector derived after the i^{th} occurrence of word w:

$$\vec{w}_{i+1} = \vec{w}_i + \vec{a} \tag{1}$$

In principle, using this simple algorithm, we could learn a vector space representation for every atomic unit. One issue with a naive embedding of every atomic unit into a vector space is the presence of *rare* atomic units. These are especially prevalent in illicit domains, not just in the form

Table 3: The compound units implemented in the current prototype

high-idf-units	Units occurring in fewer than fraction θ (by default, 1%) of initial corpus
pure-num-units	Numerical units
alpha-num-units	Alpha-numeric units that contain at least one alphabet and one number
pure-punct-units	Units with only punctuation symbols
alpha-punct-units	Units that contain at least one alphabet and one punctuation character
nonascii-unicode-units	Units that only contain non-ASCII characters

of rare words, but also as sequences of Unicode characters, sequences of HTML tags, and numeric units (e.g. phone numbers), each of which only occurs a few times (often, only once) in the corpus.

To address this issue, we define below the notion of a *compound unit* that is based on a pre-specified condition.

Definition 3: Given a universe U of atomic units and a binary condition $R : U \rightarrow \{True, False\}$, the *compound unit* C_R is defined as the largest subset of U such that R evaluates to True on every member of C_R.

Example: For 'rare' words, we could define the compound unit *high-idf-units* to contain all atomic units that are below some document frequency threshold (e.g. 1%) in the corpus.

In our implemented prototype, we defined six *mutually exclusive*[11] compound units, described and enumerated in Table 3. We modify the naive RI algorithm by only learning a single vector for each compound unit. Intuitively, each atomic unit w in a compound unit C is replaced by a special dummy symbol w_C; hence, after algorithm execution, each atomic unit in C is represented by the single vector $\vec{w}_C$.

Applying High-Recall Recognizers For a given attribute (e.g. *City*) and a given corpus, we define a recognizer as a function that, if known, can be used to exactly determine the instances of the attribute occurring in the corpus.

Definition 4: A recognizer R_A for attribute A is a function that takes a list t of tokens and positions i and $j >= i$ as inputs, and returns *True* if the tokens contiguously spanning $t[i] : t[j]$ are instances of A, and *False* otherwise.

It is important to note that, per the definition above, a recognizer cannot annotate *latent* instances that are not directly observed in the list of tokens.

Since the 'ideal' recognizer is not known, the broad goal of IE is to devise models that approximate it (for a given attribute) with high accuracy. Accuracy is typically measured in terms

[11] That is, an intersection of any two compound units will always be the empty set.

of precision and recall metrics. We formulate a two-pronged approach whereby, rather than develop a single recognizer that has both high precision and recall (and requires considerable expertise to design), we first obtain a list of candidate annotations that have high recall in expectation, and then use supervised classification in a second step to improve precision of the candidate annotations.

More formally, let R_A be denoted as an η-recall recognizer if the expected recall of R_A is at least η. Due to the explosive growth in data, many resources on the Web can be used for bootstrapping recognizers that are 'high-recall' in that η is in the range of 90-100%. The high-recall recognizers currently used rely on knowledge bases (e.g. GeoNames) from Linked Open Data [7], dictionaries from the Web and broad heuristics, such as regular expression extractors, found in public Github repositories. In our experience, we found that even students with basic knowledge of GitHub and Linked Open Data sources are able to construct such recognizers. One important reason why constructing such recognizers is relatively hassle-free is because they are typically *monotonic* i.e. new heuristics and annotation sources can be freely integrated, since we do not worry about precision at this step.

We note that in some cases, domain knowledge alone is enough to guarantee 100% recall for well-designed recognizers for certain attributes. In HT, this is true for location attributes like city and state, since advertisements tend to state locations without obfuscation, and we use GeoNames, an exhaustive knowledge base of locations, as our recognizer. Manual inspection of the ground-truth data showed that the recall of utilized recognizers for attributes like *Name* and *Age* are also high (in many cases, 100%). Thus, although 100% recall cannot be *guaranteed* for any recognizer, it is still reasonable to assume that η is high.

A much more difficult problem is engineering a recognizer to simultaneously achieve high recall *and* high precision. Even for recognizers based on curated knowledge bases like GeoNames, many non-locations get annotated as locations. For example, the word 'nice' is a city in France, but is also a commonly occurring adjective. Other common words like 'for', 'hot', 'com', 'kim' and 'bella' also occur in GeoNames as cities and would be annotated. Using a standard Named Entity Recognition system does not always work because of the language modeling problem (e.g. missing capitalization) in illicit domains. In the next section, we show how the context surrounding the annotated word can be used to classify the annotation as correct or incorrect. We note that, because the recognizers are high-recall, a successful classifier would yield both high precision and recall.

Supervised Contextual Classifier To address the precision problem, we train a classifier using contextual features. Rather than rely on a domain expert to provide a set of hand-crafted features, we derive a feature vector per candidate annotation using the notion of a context window (Definition 3.1.3.) and the word representation vectors. This process of *supervised contextual classification* is illustrated in Figure 7.

Specifically, for each annotation (which could comprise *multiple* contiguous tokens e.g. 'Salt Lake City' in the list of tokens representing the website) annotated by a recognizer, we consider the tokens in the (u, v)-context window around the annotation. We aggregate the vectors of those tokens into a single vector by performing an unweighted sum, followed by $l2$-normalization. We use this aggregate vector as the *contextual feature vector* for that annotation. Note that, unlike the representation learning phase, where the surrounding *context vectors* were aggregated into an existing representation vector, the contextual feature vector is obtained by summing the actual representation vectors.

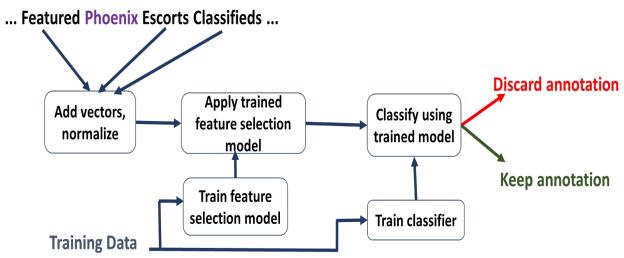

Figure 7: An illustration of supervised contextual classification on an example annotation ('Phoenix')

Table 4: Four human trafficking corpora for which word representations are (independently) learned

Name	Num. websites	Total word count	Unique word count
D-10K	10,000	2,351,036	1,030,469
D-50K	50,000	11,758,647	5,141,375
D-100K	100,000	23,536,935	10,277,732
D-ALL	184,132	43,342,278	18,940,260

For *each* attribute, a supervised machine learning classifier (e.g. random forest) is trained using between 12-120 labeled annotations, and for new data, the remaining annotations can be classified using the trained classifier. Although the number of dimensions in the feature vectors is quite low compared to *tf-idf* vectors (hundreds vs. millions), a second round of dimensionality reduction can be applied by using (either supervised or unsupervised) feature selection for further empirical benefits.

Evaluations We train the word representations on four real-world human trafficking datasets of increasing size, the details of which are provided in Table 4. Since we assume a 'streaming' setting, each larger dataset in Table 4 is a strict superset of the smaller datasets. The largest dataset is itself a subset of the overall human trafficking corpus that was scraped as part of research conducted in the DARPA MEMEX program[12].

Since ground-truth extractions for the corpus are unknown, we randomly sampled websites from the overall corpus[13], applied four high-recall recognizers, and for each annotated set, manually verified whether the extractions were correct or incorrect for the corresponding attribute. The details of this sampled ground-truth are captured in Table 5. Each annotation set is named using the format *GT-{RawField}-{AnnotationAttribute}*, where *RawField* can be either the HTML title or the scraped text. and *AnnotationAttribute* is the attribute of interest for annotation purposes.

System: The overall system requires developing two components for each attribute: a high-recall recognizer and a classifier for pruning annotations. We developed four high-recall recognizers, namely *GeoNames-Cities, GeoNames-States, RegEx-Ages* and *Dictionary-Names*. The first two of these relies on the freely available GeoNames[14] dataset [69]; we use the entire dataset for our experiments, which involves modeling each GeoNames dictionary as a trie, owing to its large

[12] `http://www.darpa.mil/program/memex`

[13] Hence, it is possible that there are websites in the ground-truth that are not part of the corpora in Table 4.

[14] `http://www.geonames.org/`

Table 5: Five ground-truth datasets on which the classifier and baselines are evaluated

Name	Pos. ann.	Neg. ann.	Recognizer Used
GT-Text-City	353	15,783	GeoNames-Cities
GT-Text-State	100	16,036	GeoNames-States
GT-Title-City	37	513	GeoNames-Cities
GT-Text-Name	162	14,337	Dictionary-Names
GT-Text-Age	116	14,306	RegEx-Ages

Table 6: Stanford NER features that were used for re-training the model on our annotation sets

useClassFeature=true	useNext=true
useWord=true	useSequences=true
useNGrams=true	usePrevSequences=true
noMidNGrams=true	maxLeft=1
useDisjunctive=true	useTypeSeqs=true
maxNGramLeng=6	useTypeSeqs2=true
usePrev=true	useTypeySequences=true
wordShape=chris2useLC	

memory footprint. For extracting ages, we rely on simple regular expressions and heuristics that were empirically verified to capture a broad set of age representations[15]. For the name attribute, we gather freely available *Name* dictionaries on the Web, in multiple countries and languages, and use the dictionaries[16] in a case-insensitive recognition algorithm to locate names in the raw field (i.e. text or title).

Baselines: We use different variants of the Stanford Named Entity Recognition system (NER) as our baselines [22]. For the first set of baselines, we use two pre-trained models trained on different English language corpora. Specifically, we use the 3-Class and 4-Class pre-trained models. We use the LOCATION class label for determining city and state annotations, and the PERSON label for name annotations. Unfortunately, there is no specific label corresponding to age annotations in the pre-trained models; hence, we do not use the pre-trained models as age annotation baselines.

It is also possible to *re-train* the underlying NER system on a new dataset. For the second set of baselines, therefore, we re-train the NER models by randomly sampling 30% and 70% of each annotation set in Table 5 respectively, with the remaining annotations used for testing. The features and values that were employed in the re-trained models are enumerated in Table 6. Further documentation on these feature settings may be found on the *NERFeatureFactory* page[17]. All training and testing experiments were done in ten independent trials[18]. We use default parameter settings, and report average results for each experimental run. Experimentation using

[15]The age extractors we used are also available in the Github repository accessed at `https://github.com/usc-isi-i2/dig-age-extractor`

[16]For replication, the full set of dictionaries used may be accessed at `https://github.com/usc-isi-i2/dig-dictionaries/tree/master/person-names`

[17]Documentation accessed at `http://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/ie/NERFeatureFactory.html`

[18]When evaluating the pre-trained models, the training set is ignored and only the testing set is classified.

Table 7: Comparative results of three systems on precision (P), recall (R) and F1-Measure (F) when training percentage is 30. For the pre-trained baselines, we only report the best results across all applicable models

Ground-truth Dataset	Our System (P/R/F)	Re-trained Baseline (P/R/F)	Pre-trained Baseline (P/R/F)
GT-Text-City	0.5207/0.5050/0.5116	**0.9855**/0.1965/0.3225	0.7206/**0.7406/0.7299**
GT-Text-State	**0.7852**/0.6887/**0.7310**	0.64/0.0598/0.1032	0.2602/**0.8831**/0.3993
GT-Title-City	0.5374/0.5524/0.5406	**0.8633**/0.1651/0.2685	0.8524/**0.7341/0.7852**
GT-Text-Name	0.7201/**0.5850/0.6388**	**1**/0.2103/0.3351	0/0/0
GT-Text-Age	0.8993/**0.9156/0.9068**	**0.9102**/0.7859/0.8412	N/A
Average	0.6925/**0.6493/0.6658**	**0.8798**/0.2835/0.3741	0.4583/0.5895/0.4786

other configurations, features and values is left for future studies.

Setup and Parameters: System parameters were set as follows. The number of dimensions in Definition 3.1.3. was set at 200, and the sparsity ratio was set at 0.01. These parameters are similar to those suggested in previous word representation papers; they were also found to yield intuitive results on semantic similarity experiments. To avoid the problem of rare words, numbers, punctuation and tags, we used the six compound unit classes earlier described in Table 3. In all experiments where defining a context was required, we used symmetric $(2, 2)$-context windows; using bigger windows was not found to offer much benefit. We trained a random forest model with default hyperparameters (10 trees, with Gini Impurity as the split criterion) as the supervised classifier, used supervised k-best feature selection with k set to 20, and with the Analysis of Variance (ANOVA) F-statistic between class label and feature used as the feature scoring function.

Because of the *class skew* in Table 5 (i.e. the 'positive' class is typically much smaller than the 'negative' class) we oversampled the positive class for balanced training of the supervised contextual classifier.

The metrics used for evaluating IE effectiveness are Precision, Recall and F1-measure.

In the interests of demonstrating a reasonably lightweight system, all experiments in this paper were run on a serial iMac with a 4 GHz Intel core i7 processor and 32 GB RAM. All code (except the Stanford NER code) was written in the Python programming language, and has been made available on a public Github repository[19] with documentation and examples. We used Python's Scikit-learn library (v0.18) for the machine learning components of the prototype.

Performance against baselines Table 7 illustrates system performance on Precision, Recall and F1-Measure metrics against the re-trained and pre-trained baseline models, where the re-trained model and our approach were trained on 30% of the annotations in Table 5. We used the word representations derived from the D-ALL corpus. On average, the proposed system performs the best on F1-Measure and recall metrics. The re-trained NER is the most precise system, but at the cost of much less recall (<30%). The good performance of the pre-trained baseline on the *City* attribute demonstrates the importance of having a large training corpus, even if the corpus is not directly from the test domain. On the other hand, the complete failure of the pre-trained baseline on the *Name* attribute illustrates the dangers of using out-of-domain training data. As noted earlier, language models in illicit domains can significantly differ from natural language models; in fact,

[19] `https://github.com/mayankkejriwal/fast-word-embeddings`

Table 8: Comparative results of three systems when training percentage is 70

Ground-truth Dataset	Our System (P/R/F)	Re-trained Baseline (P/R/F)	Pre-trained Baseline (P/R/F)
GT-Text-City	0.5633/0.6081/0.5841	**0.9434**/0.3637/0.5000	0.6893/**0.7401**/**0.7128**
GT-Text-State	**0.7916**/0.7269/**0.7502**	0.7833/0.2128/0.2971	0.1661/**0.7830**/0.2655
GT-Title-City	0.6403/**0.6667**/0.6437	**0.9417**/0.3333/0.4790	0.9133/0.6384/**0.7289**
GT-Text-Name	0.7174/**0.6818**/**0.6960**	**1**/0.3747/0.5140	0/0/0
GT-Text-Age	0.9252/**0.9273**/**0.9251**	**0.9254**/0.8454/0.8804	N/A
Average	0.7276/**0.7222**/**0.7198**	**0.9188**/0.4260/0.5341	0.4422/0.5404/0.4268

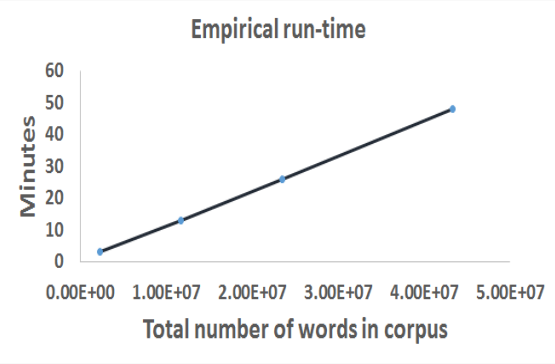

Figure 8: Empirical run-time of the adapted random indexing algorithm on the corpora in Table 4

names in human trafficking websites are often represented in a variety of misleading ways.

Recognizing that 30% training data may constitute a sample size too small to make reliable judgments, we also tabulate the results in Table 8 when the training percentage is set at 70. Performance improves for both the re-trained baseline and our system. Performance declines for the pre-trained baseline, but this may be because of the sparseness of positive annotations in the smaller test set.

We also note that performance is relatively well-balanced for our system; on all datasets and all metrics, the system achieves scores greater than 50%. This suggests that our approach has a degree of robustness that the CRFs are unable to achieve; we believe that this is a direct consequence of using contextual word representation-based feature vectors.

Runtimes: We recorded the runtimes for learning word representations using the random indexing algorithm described earlier on the four datasets in Table 4, and plot the runtimes in Figure 8 as a function of the total number of words in each corpus. In agreement with the expected theoretical time-complexity of random indexing, the empirical run-time is linear in the number of words, for fixed parameter settings. More importantly, the absolute times show that the algorithm is extremely lightweight: on the D-ALL corpus, we are able to learn representations in under an hour.

We note that these results do not employ any obvious parallelization or the multi-core capabilities of the machine. The linear scaling properties of the algorithm show that it can be used even for very large Web corpora. In future, we will investigate an implementation of the algorithm in a distributed setting.

Robustness to corpus size and quality: One issue with using large corpora to derive word

Table 9: A comparison of F1-Measure scores of our system (30% training data), with word representations trained on different corpora

Ground-truth	D-10K	D-50K	D-100K	D-ALL
GT-Text-City	0.4980	0.5058	0.4909	**0.5116**
GT-Text-State	0.7362	0.7385	**0.7526**	0.7310
GT-Title-City	**0.6148**	0.5638	0.5061	0.5406
GT-Text-Name	0.6756	0.6881	**0.6920**	0.6388
GT-Text-Age	**0.9387**	0.9364	0.9171	0.9068
Average	**0.6927**	0.6865	0.6717	0.6658

Table 10: A comparison of F1-Measure scores of our system (70% training data), with word representations trained on different corpora

Ground-truth	D-10K	D-50K	D-100K	D-ALL
GT-Text-City	**0.5925**	0.5781	0.5716	0.5841
GT-Text-State	0.7357	**0.7641**	0.7246	0.7502
GT-Title-City	0.6424	0.6428	0.6364	**0.6437**
GT-Text-Name	**0.7665**	0.7091	0.7333	0.6960
GT-Text-Age	0.9311	**0.9634**	0.9347	0.9251
Average	**0.7336**	0.7315	0.7201	0.7198

representations is *concept drift*. The D-ALL corpora, for example, contains tens of different Web domains, even though they all pertain to human trafficking. An interesting empirical issue is whether a smaller corpus (e.g. D-10K or D-50K) contains enough data for the derived word representations to converge to reasonable values. Not only would this alleviate initial training times, but it would also partially compensate for concept drift, since it would be expected to contain fewer unique Web domains.

Tables 9 and 10 show that such generalization is possible. The best F1-Measure performance, in fact, is achieved for D-10K, although the average F1-Measures vary by a margin of less than 2% on all cases. We cite this as further evidence of the robustness of the overall approach.

Effects of feature selection: Finally, we evaluate the effects of feature selection in Figure 9 on the *GT-Text-Name* dataset, with training percentage set[20] at 30. The results show that, although performance is reasonably stable for a wide range of k, some feature selection is necessary for better generalization.

Contributions The main contributions of this part of the work are: (1) a lightweight feature-agnostic information extraction system for a highly heterogeneous, illicit domain like human trafficking. Our approach is simple to implement, does not require extensive parameter tuning, infrastructure setup and is incremental with respect to the data, which makes it suitable for deployment in streaming-corpus settings. (2) s good generalization even when only a small corpus is available after the initial domain-discovery phase, and is robust to the problem of concept drift encountered in large Web corpora. (3) We test our approach extensively on a real-world human trafficking corpus

[20]Results on the other datasets were qualitatively similar; we omit full reproductions herein.

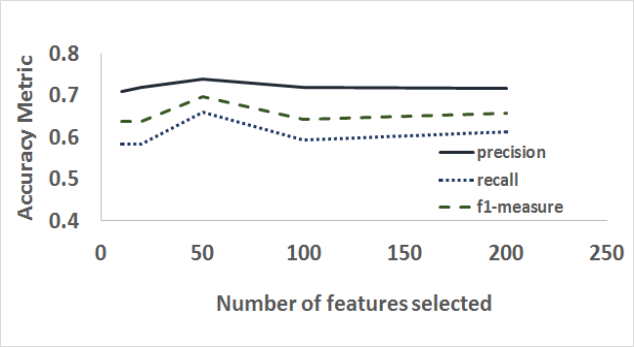

Figure 9: Effects of additional feature selection on the *GT-Text-Name* dataset (30% training data)

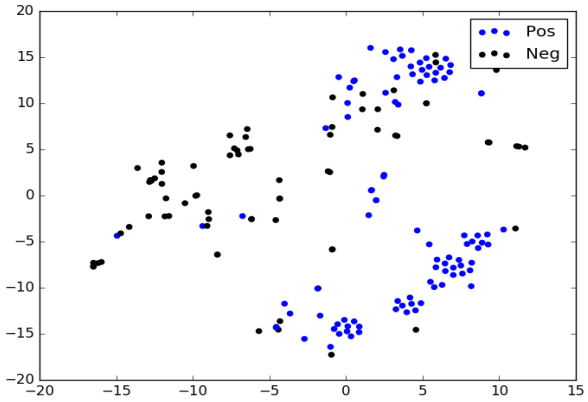

Figure 10: Visualizing city contextual classifier inputs (with colors indicating ground-truth labels) using the t-SNE tool

Table 11: Some representative examples of correct city extractions using the proposed method

. . . 1332 SOUTH 119TH STREET, **OMAHA** NE 68144 . . .
. . . Location: Bossier City/**Shreveport** . . .
. . . to service the areas of **Salt Lake City** Ogden,Farmington,Centerville,Bountiful . . .
. . . 4th August 2015 in **rochester** ny, new york . . .
. . . willing to Travel (Cali, **Miami**, New York, Memphis . . .
. . . More girls from **Salt Lake City**, UT . . .

containing hundreds of thousands of Web pages and millions of unique words, many of which are rare and highly domain-specific. Evaluations show that our approach outperforms traditional Named Entity Recognition baselines that require manual feature engineering. Comparisons against CRF baselines based on the latest Stanford Named Entity Resolution system (including pre-trained models as well as new models that we trained on human trafficking data) show that, on average, across five ground-truth datasets, our approach outperforms the next best system on the recall metric by about 6%, and on the F1-measure metric by almost 20% in low-supervision settings (30% training data), and almost 20% on both metrics in high-supervision settings (70% training data). Concerning efficiency, in a serial environment, we are able to derive word representations on a 43 million word corpus in under an hour. Degradation in average F1-Measure score achieved by the system is less than 2% even when the underlying raw corpus expands by a factor of 18, showing that the approach is reasonably robust to concept drift.

Some domains, for example, were found to have the same kind of structured format as the second row of Table 11 (i.e. *Location:* followed by the actual locations), but many other domains were far more heterogeneous.

The results illustrate the merits of unsupervised feature engineering and contextual supervision. In principle, there is no reason why the word representation learning module in Figure 5 cannot be replaced by a more adaptive algorithm like Word2vec [48]. We note again that, before applying such algorithms, it is important to deal with the heterogeneity problem that arises from having many different Web domains present in the corpus. While earlier results in this section (Tables 9 and 10) showed that random indexing is reasonably stable as more websites are added to the corpus, we also verify this robustness *qualitatively* using a few domain-specific examples in Table 12. We ran the qualitative experiment as follows: for each seed token (e.g. 'tall'), we searched for the two nearest neighbors in the semantic space induced by random indexing by applying cosine similarity, using two different word representation datasets (D-10K and D-ALL). As the results in Table 12 show, the induced distributional semantics are stable; even when the nearest neighbors are different (e.g. for 'tall'), their semantics still tend to be similar.

Another important point implied by both the qualitative and quantitative results on D-10K is that random indexing is able to generalize quickly even on *small* amounts of data. To the best of our knowledge, it is currently an open question (theoretically and empirically), at the time of writing, whether state-of-the-art *neural* embedding-based word representation learners can (1) generalize on small quantities of data, especially in a single epoch ('streaming data') (2) adequately compensate for concept drift with the same degree of robustness, and in the same lightweight manner, as the random indexing method that we adapted and evaluated in this paper. A broader empirical study

Table 12: Examples of semantic similarity using random indexing vectors from D-10K and D-ALL

Seed-token	**D-10K**	**D-ALL**
tall	figure, attractive	fit, cute
florida	california, ohio	california, texas
green	blue, brown	blue, brown
attractive	fit, figure	elegant, fit
open-minded	playful, sweet	passionate, playful

on this issue is warranted.

Concerning contextual supervision, we qualitatively visualize the inputs to the contextual city classifier using the t-SNE tool [46]. We use the ground-truth labels to determine the color of each point in the projected 2d space. The plot in Figure 10 shows that there is a reasonable separation of labels; interestingly there are also 'sub-clusters' among the positively labeled points. Each sub-cluster provides evidence for a similar context; the number of sub-clusters even in this small sample of points again illustrates the heterogeneity in the underlying data.

A last issue that we mention is the generalization of the method to more unconventional attributes than the ones evaluated herein. In ongoing work, we have experimented with more domain-specific attributes such as *ethnicity* (of escorts), and have achieved similar performance. In general, the presented method is applicable whenever the context around the extraction is a suitable clue for disambiguation.

3.2 Knowledge Graph

Our solution to the investigative search problem begins by constructing a *knowledge graph* from the raw corpus of HTML pages, using a variety of information extraction and clustering modules based on an underlying *investigative schema* [15]. We define a knowledge graph (KG) as a *directed, labeled, multi-relational* graph where nodes represent entities, attribute values or entity clusters (*latent* entities), and edges represent relationships.

Because of the imperfection of extraction and clustering modules, the constructed KG is typically very noisy (a non-trivial subset of extracted attribute values is incorrect), large-scale (containing tens of millions of nodes and edges), and is semi-structured (many attribute values are missing, multi-valued and even textual).

3.2.1. Investigative Schema

To facilitate precise search capabilities and preclude errors in solutions to difficult problems such as word sense disambiguation [63], we develop a controlled vocabulary of terms denoted herein as an *investigative schema* $\mathcal{I}$, represented as a labeled, directed, *acyclic* graph, with *subclassOf*, *attributeOf*, *hasValue* and *memberOf* edges. Figure 11 illustrates the investigative schema used in the current prototype. This schema was defined collaboratively with actual investigators and domain experts, and contains four *primary* classes[21], in addition to a set of primitively typed *semantic attributes*[22].

[21] Instances of these classes are considered either *entities* or *latent* entities (for *Vendor*) in our domain of discourse.

[22] That is, the type is either *number*, *date* or *string*.

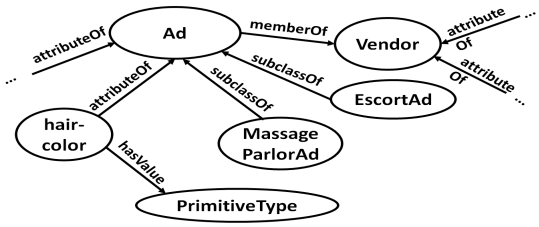

Figure 11: A fragment of the current investigative schema $\mathcal{I}$.

We permit a semantic attribute to be multi-valued (a bag), but each value must have the primitive type of the attribute. Among the four classes, *EscortAd* and *MassageParlorAd* are referred to as *base classes*, themselves sub-classes of the superclass *Ad*; *Vendor* is denoted as a *cluster class*, of which an instance will always be a set of *Ad* instances (hence, the *memberOf* relation). Currently, *EscortAd* and *MassageParlorAd* share semantic attributes and we do not distinguish between them in our data, although in the post-processing module in Figure 5, an implemented supervised machine learning classifier can be used to offer a probabilistic classification.

We define a single set of semantic attributes for the *Ad* subclasses, namely *age, text-content, email, eye-color, hair-color, height, location, name, nationality/ethnicity, phone, posting-date, price, review-id, service, social-media-id, street-address, title, weight* and *multiple-providers*[23]. The *Vendor* cluster class has two attributes: *seed-phone* and *seed-email*, which are the respective *unions* of the phones and emails corresponding to the ads that are members of a *Vendor* cluster instance. The rationale for these attributes is that (1) phones and emails extracted from ads were used for inferring the clusters, as subsequently described, and (2) investigators prefer querying for cluster instances using phone and email facets.

More technically, the investigative schema is a constrained version of a *shallow ontology*, used often in the Semantic Web for schema-rich knowledge graphs such as DBpedia and GeoNames [40], [69]. The ontology defines a domain of discourse, and is critical for precisely capturing domain assumptions. More importantly, the schema defined above is not only simple but designed to be extensible: usually, real-world investigators want flexibility in the available set of semantic attributes e.g. we are looking to add semantic attributes such as *tattoo* and *drug use* to the schema.

3.2.2. Knowledge Graph Construction

An important first step in achieving the goals illustrated in Figure 5 is *knowledge graph construction* (KGC). KGC is an area of research in its own right [57], [19]; we only describe the important elements herein. The *online* component of the system (the entity-centric query engine) that is the technical focus of this work, is *agnostic* to the constructed knowledge graph (KG), although the overall quality of the answers will (in the general case) depend on the KG. To abstract implementation-specific details, we assume the availability of a battery of *attributed information extractors* for each semantic attribute in $\mathcal{I}$:

Definition 1: An *attributed information extractor* (a-IE) X is a 2-element tuple $< f, \mathcal{M} >$, where f, denoted as an information extractor, is a (many-many) mapping from an input string to a set of primitively-typed values, and $\mathcal{M}$ is a non-empty *extraction metadata dictionary* of key-value

[23]A Boolean flag indicating whether multiple people will be offering the service.

pairs.

In the literature, implementations of f include wrappers, text scrapers (for extracting text content from raw HTML pages), regular expressions, dictionaries, entity sets and external knowledge bases like GeoNames [69], [15]. While wrappers and text scrapers take the raw HTML content of a webpage (represented as a single long string) as input, other IEs take the output(s) of text scrapers as their inputs.

The metadata of an a-IE typically includes a succinct representation of expert confidence in f. One compulsory attribute of an a-IE is a *semantic type*, which enables us to *type* each output of f in terms of the attributes in the investigative schema $\mathcal{I}$. A second compulsory attribute that is extremely important for non-technical domain experts navigating the knowledge graph is *provenance*. The simplest example of provenance is the original URL (and in many cases, the *cached* webpage, which may not be online at the time of search.

Definition 2: Given an a-IE $X =< f, \mathcal{M} >$ and an investigative schema $\mathcal{I}$, a *semantic type* is a function mapping the output of f to a semantic attribute in $\mathcal{I}$. The output of f is referred to as an *extracted semantic type*.

Consider a Conditional Random Field (CRF)-based f for the *hair-color* semantic attribute. f would take text as input and output a (possibly empty) set of hair colors (*extracted semantic types*). Suppose also that, during training, the CRF parameters can be optimized to deliver either high expected precision or high expected recall. Rather than choosing between the two, the KGC could have two a-IEs e.g. *hair_color_high_precision*=$< f, \mathcal{M} = \{[isHighPrecision, True], [semanticType, hair\text{-}color]\} >$; similarly, *hair_color_high_recall* is defined, except with a differently trained f, and with *isHighRecall* rather than *isHighPrecision*. Intuitively, the metadata allow us to use the a-IEs in a variety of ways in the query reformulation algorithm. Note that the semantic types permit us to define KGC as a black box process precisely because they explicitly connect elements between the *actual* schema of the KG (the implementation-specific a-IEs) and $\mathcal{I}$.

In DIG, we treat KGC as an independent 'black box' that takes as input a raw corpus and yields a semi-structured knowledge graph; in the evaluations, we quantitatively compare the performance of two independently constructed knowledge graphs using both DIG, and also the DeepDive system [65], [51]. At the time of writing, the extractors listed in Table 1 are continuously being maintained and updated [2]. It is important to note that the subsequently described search engine directly uses the IE metadata to reformulate queries and make ranking decisions, allowing specific implementation details to be abstracted in that step. For instance, the metadata abstraction allowed the search to be conducted over knowledge graphs (constructed using DIG and DeepDive respectively) with two completely different extraction schemas.

3.2.3. Cluster Inference

Cluster inference is the process of deriving the latent entities (instances of the *Vendor* class in Figure 11) in the knowledge graph. The set of real-world cues that can be used to cluster human trafficking entities is a highly contentious issue, and includes text similarity, use of Unicode motifs, and phone and email *co-occurrence* data. While we use the last cue, any clustering algorithm can potentially be used to identify vendor instances. This also explains the rationale behind the cluster attributes (*seed-phone* and *seed-email*); given a 'seed' phone or email, we define the *query-centric* problem of retrieving the 'vendor' instance associated with that seed as the set of ad entities satisfying

the following conditions: the seed was explicitly extracted from at least one of the ads in the set, and there is a *co-occurrence path* of phones and emails between any pair of ads in the set[24]. In other words, every vendor instance represents a *connected component* of ad instances, with an edge defined between two ads if they share phones or emails.

Naively using connected components as clusters is problematic for a number of reasons. First, there are some extremely common but irrelevant phone numbers that show up in a non-trivial number of pages. An example would be a customer service number from Verizon (or the ISP hosting the site), extracted from the bottom of the HTML page. Second, the noisy extractions output by the regular expression-based phone extractor are not independent. For example, a sequence of digits (such as a zipcode followed by a set of prices) is more likely to be mistaken for a phone (by the extractor) than an isolated age. Similar to the first problem, the second problem also leads to nodes in the constructed phone network that result in extremely large connected components (by serving as a 'bridge' between two connected components).

In a previous prototypical solution, we handled this problem by manually blacklisting nodes that had abnormally high degrees. In the current prototype, this step has been successfully automated by using a *random walk-based connected components* algorithm. Connections mediated by 'faulty' nodes are typically extremely weak owing to their high edge degrees, allowing us to control the formation of large connected components.

The result of information extraction and cluster inference is a *multi-relational directed, labeled graph*, denoted henceforth as the *knowledge graph* (KG), with two types of nodes: *Vendor* nodes and *Ad* (equivalently, *entity*) nodes. Every document that is processed by the extraction system is assigned a unique identifier and becomes an *entity* node in the KG.

3.2.4. Knowledge Graph Indexing

For real-time execution of complex queries, the knowledge graph must be properly *indexed* in an appropriate NoSQL (i.e. key-value) document store [27]. Specifically, each ad is considered a document with multiple fields, with each field bijectively mapped to an a-IE. Since not every ad is guaranteed to have an extracted semantic type (or may have multiple such extractions), the documents are semi-structured i.e. will not have values for many fields in the general case. This is because, both for purposes of achieving high recall during search as well as robust handling of unexpected extraction outcomes, values are not necessarily from a controlled vocabulary or closed set (e.g., a set of pre-determined hair colors, or a phone number guaranteed to have an eligible US phone format). Given specific 'analyzer' instructions (the most important of which is the tokenizer used for chunking strings into bags of tokens), suitable inverted indices can be constructed for each such field, with the document id (the entity node) serves as the link between multiple field indices. Given a query in its *Domain-specific Query Language* (DQL), a good NoSQL engine like Elasticsearch or MongoDB makes judicious use of these inverted indices for fast retrieval and ranking.

[24] An example of such a set for the seed phone p_1 is a set containing three ads (say ad_1, ad_2, and ad_3) that respectively contain phones $\{p_1, p_2\}$, $\{p_2, p_3\}$, and $\{p_3, p_4\}$

3.2.5. Indicator Mining

The growth, combined with the ease of sharing information on the Web, has also led to increased illicit activity both on the Open and Dark Web, an egregious example being human trafficking (HT) [5], [68]. The DARPA MEMEX program, which funds research into domain-specific search, has collected hundreds of millions of online sex advertisements, a significant (but unknown) number of which are believed to be sex (and human) *trafficking* instances.

Flagging scraped content from webpages in such domains with activity tags or *indicators*, denoted herein as the *indicator mining* problem, has an investigative and socially motivated purpose not just in online human trafficking but several other domains that have also been studied in MEMEX, including patent trolling, counterfeit electronic sales, illegal weapons sales and narcotics. In recent years, some of these 'illicit' Web domains have been intensely studied by social and computer scientists seeking to profile activity on the Dark Web [70], [72]; however, to the best of our knowledge, *flagging* and *modeling* Web content with *implicit semantic tags* indicating illicit activity has received considerably less attention. A good model for an indicator must incorporate several non-trivial subtleties to be faithful to real-world observations.

Defined naively, an indicator is a flag, typically binary, but potentially multi-categorical, that is suggestive of *suspicious* activity that would warrant investing more resources into investigating the *subject* of the content being flagged.

In the case of the online sex advertisement domain, subjects are not only escorts (usually, the victims of trafficking) but also perpetrator organizations like massage parlors, procurers, as well as reviewers and clients. While the ideal motivation is to directly flag an online sex trafficking document with a single human trafficking indicator, this is extremely problematic in practice, especially without a proper field level investigation. A more attainable goal is to instead use the mined indicators to *guide* further investigation. To distinguish trafficked escorts from non-trafficked escorts, investigators seek *signals* that suggest (whether at present or in the future), for example, that the subject exhibits *movement* between cities or provides *risky* services that were highly correlated with trafficking activity in social science studies. The provenance of the signal can be complex and depend on both 'knowledge' artifacts like named entities or relationships, but also the text. For example, the sentence 'Brand NEW to town and looking for New Fun Friends' suggests movement, not because of an entity like 'town' or 'friends' but because of adjectives like NEW. Correctly modeling and generalizing this notion is challenging both from a formal and an inferential standpoint due to high diversity of content.

In this work, we investigated an *indicator model* and formalism for indicators by defining ontologies and semantics that attempt to provide insight into the implicit semantics, and special nature, of indicators. To express the richness of indicator provenance, our model combines syntactic artifacts developed in the Natural Language Processing community with more traditional domain semantics. We implement our model in an end-to-end indicator mining approach called FlagIt (**Fl**exible and **a**daptive **g**eneration of **I**ndicators from **t**ext). To reduce the burdens of manual supervision and potential user bias, FlagIt combines a lightweight expert system with applied research (both recent and classic) in minimally supervised machine learning, including unsupervised text embeddings [28] and semi-supervised heuristic re-labeling [71], to achieve average F-Measure scores slightly below 80%. FlagIt is simple and extensible, and scales to millions of sentences.

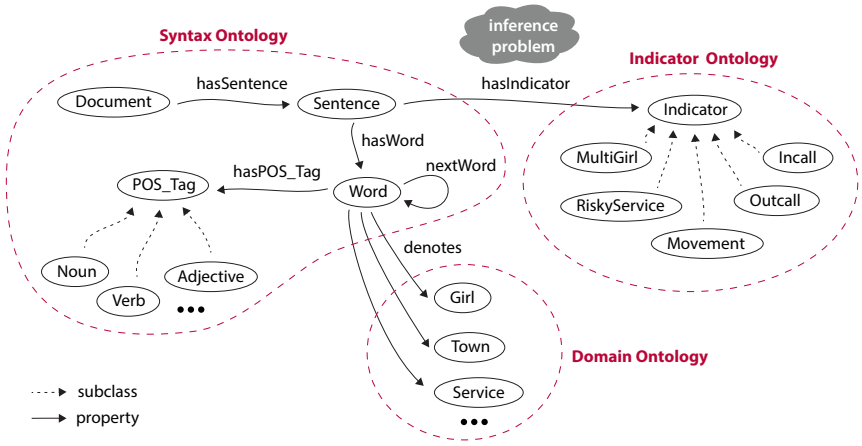

Figure 12: Ontologies for indicator mining.

Formalism We formalize the indicator mining problem using three ontologies (Figure 12). The *syntax ontology* represents the syntactic structure of documents in the input corpus. We assume each document is segmented into sentences, and words are labeled with part of speech (POS) tags, a task that can be easily accomplished using widely available tools such as NLTK[25] or spaCy[26].

The *domain ontology* represents the entities of interest (the figure shows a subset of the classes relevant in the human trafficking domain). Accurate information extraction from sex ads is difficult as the language model used in these ads is unusual, so we use a simple domain ontology with classes to represent the entities of interest. The ontology does not represent relationships among these classes as relation extraction in this domain is extremely difficult. Instead, we assume that relevant words in the corpus are labeled with the appropriate classes in the domain using the denotes property. The labeling task can be easily accomplished by having domain experts create term glossaries for each class, and using these glossaries to tag the words. For example, the domain experts defined class Town using a glossary containing "town", "area", "place" and "city".

The *indicator ontology* represents the indicators of interest. The problem addressed in this work is the inference of the hasIndicator relationship between sentences and indicator classes.

The inference problem may be formalized using the Semantic Web Rule Language[27] (SWRL) by defining rules that relate the terms in the syntax and domain ontologies to terms in the indicator ontology. The formalization would consist of multiple rules where each rule identifies a specific pattern. Consider the following SWRL rule to infer the Movement indicator: clause (1) identifies the word "new", clause (2) identifies a preposition following the first word, clause (3) identifies a word that denotes Town following the preposition, and clause (4) states that all words belong in a sentence. When these conditions are met, the rule infers the Movement indicator for the sentence (clause 5). This rule, would correctly tag sentences such as "Brand NEW to town and looking for New Fun Friends".

[25] http://www.nltk.org
[26] https://spacy.io
[27] https://www.w3.org/Submission/SWRL/

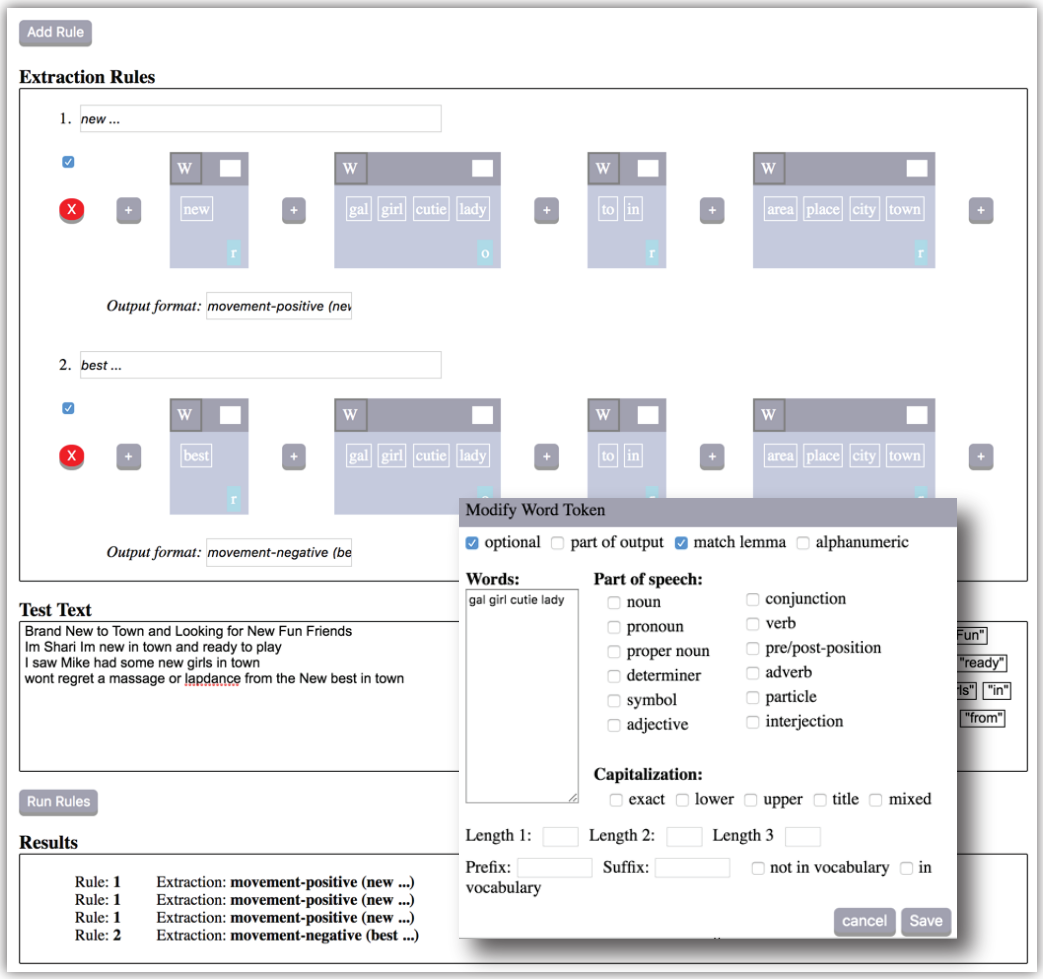

Figure 13: Easy to use interactive editor for defining indicator rules.

$$\text{Word}(?w_1) \wedge \text{label}(?w_1, \text{“new”}) \wedge \tag{2}$$

$$\text{Word}(?w_2) \wedge \text{nextWord}(?w_1, ?w_2) \wedge \text{hasPOS\_Tag}(?w_2, \text{Preposition}) \wedge \tag{3}$$

$$\text{Word}(?w_3) \wedge \text{nextWord}(?w_2, ?w_3) \wedge \text{denotes}(?w_3, \text{Town}) \wedge \tag{4}$$

$$\text{Sentence}(?s) \wedge \text{hasWord}(?s, ?w_1) \wedge \text{hasWord}(?s, ?w_2) \wedge \text{hasWord}(?s, ?w_3) \tag{5}$$

$$\Rightarrow \text{hasIndicator}(?s, \text{Movement}) \tag{6}$$

With additional rules, the indicator mining problem would be formalized precisely, and a SWRL reasoner could be used to produce a reference implementation.

In our work we implemented an equivalent baseline using a graphical user interface (Figure 13) that enables domain experts to create rules with identical semantics to SWRL rules such as the one listed above. Our evaluations show that this approach for formalizing the problem is useful, but doesn't always agree with annotations provided by domain experts. We show that combining the rule-based approach with machine learning techniques yields a practical inferencing mechanism for indicator mining with better agreement with expert annotations.

Approach We construct the sentence corpus from Web resources that are known to contain high levels of human trafficking (HT) activity. The sentences are obtained by executing a webpage pre-preprocessing pipeline that was tuned to scale and generalize to a 54 GB HT corpus containing webpages from numerous *Web domains*. First, we use the Readability Text Extractor[28] (RTE) to scrape descriptive content from each webpage in the corpus. We tuned RTE for high recall: with high probability, all relevant sentences constituting the main content of the webpage are extracted, but some irrelevant sentences (e.g., "taylor profile was posted 14 weeks 4 days ago") may also be extracted. Next, the output of RTE is segmented into a list of sentences by using newlines and consecutive occurrences of three whitespace characters as delimiters. All sentences were converted to lowercase and deduplicated before the minimally supervised machine learning modules (e.g., the text embeddings) but not before executing the lightweight expert system, which relies on capitalization and other text features to perform well.

Given a sentence from this corpus of segmented sentences, the indicator mining problem can be modeled as *multi-label* text classification, assuming that representative training data and feature functions are available. Unfortunately, we have no training data available for this task, and enormous negative class skew (higher than 90% in many cases) in the task domain preempts random sampling and labeling. Because indicator mining is an emerging problem in data mining, it is also not clear what indicator-specific features will lead to good performance for a given indicator. FlagIt attempts to address these challenges in a simple and lightweight manner, as subsequently described.

We implemented five indicator classifiers, namely *incall*, *outcall*, *movement*, *risky* and *multi-girl*. *Incall* explicitly indicates that a client must visit an escort (conversely for *outcall*) at a specified location. *Movement* indicates that the escort is visiting the locale (usually for short periods) from a different previous locale, while *risky* indicates that the escort is willing to engage in risky sexual activities (as determined by domain experts). *Multi-girl* (along with *movement*) is highly correlated with organized sex trafficking activity.

Lightweight Expert System Human trafficking is an 'unusual' domain in that it has features and subtleties not well understood except by experienced domain experts. Such experts include investigators with actual field experience, as well as social scientists who have studied the field academically. While such users are capable of giving examples, and even expressing short rules (with some offline training) using a fairly intuitive syntax, they are non-technical and are not well-versed either in machine learning or in the use of NLP packages like spaCy or nltk. Given the short time frame in which FlagIt[29] had to be developed, as well as sparse personnel, an expert system with many complex rules was neither desirable nor possible.

Instead, we developed a *lightweight expert system* (LES) that relies on (subsequently described) shallow *pattern matching* rules, with each rule exclusively falling into one of four *rule categories* depending on its predictive relationship to the indicator: positive (P), strong positive (Sp), negative (N) and strong negative (Sn). Earlier, we showed a visualization of the actual rule editor that is used for defining the rules; this section develops the technical formalism. The 'strong' qualifier is meant to express user confidence, providing users a simple but effective way of expressing when they are very sure about a rule, and when they are not. These categories are motivated by extensive research

[28]While available at the time of preprocessing, RTE has since been superseded by a new web scraping API called Mercury: `https://www.readability.com/`

[29]End-to-end research and development on FlagIt, now in its final phase, was conducted in the last 5 months of the MEMEX program.

in cognitive science and crowdsourcing showing that asking users for fine-grained confidence outputs (including probabilities) is rife with issues of labeling consistency.

We define a pattern matching rule as a finite sequence of *pattern elements*, where a pattern element can either be a *constant* token (a finite sequence of Unicode characters) or a pattern variable V. In keeping with the lightweight nature of the system, four kinds of pattern variables are considered. We describe these variables below, followed by some representative examples.

Glossary Variable. Given a glossary G (a finite set of tokens), a glossary variable V_G can take exactly one token from G as its value. In some cases, glossaries are easily available from the Web (e.g., a list of common English names) but in other cases, have to be specified either by the expert or through entity set expansion and glossary mining [54]. For the indicators considered in this paper, five glossaries are used for both *incall* and *outcall*, two glossaries each for *movement* and *risky*, and four glossaries for *multi-girl*.

Regular Expression (RegEx). Pattern variables can also be encoded as regular expressions. For speed and efficiency (and also noting that domain experts are non-technical), we limit expressivity of regular expressions in FlagIt. The most commonly used RegEx in FlagIt was to check for tokens possibly symbolizing names and geolocations by specifying capitalization and alphabetic constraints.

NLP Tags. Despite being lightweight, the expert system can make use of tags, such as POS tags and English dependency labels[30], output by NLP packages like spaCy. As a subsequent example illustrates for *incall*, dependency labels, such as an *adjectival modifier* tag, can be used to great effect without overhead.

Named Entities. The LES can also use an expected named entity *type* as a pattern variable specification. The variable is assigned the word, only if it is extracted as a named entity of that type [49].

Example. Given a glossary *place*, containing residence terms and synonyms such as *apartment* and *studio*, and another glossary *priv* containing words such as *private* and *discreet*, a simple but effective Sp rule for *incall* is the sequence $(G_{priv} < -amod, G_{place})$, which would match a pattern such as *private apartment* in the text. The $< -amod$ specifies that the word must have an adjectival modifier dependency tag. This example shows that finite combinations of pattern variables are also possible.

In a slight abuse of notation, we use the symbols P, Sp, N and Sn to respectively refer to the collections of pattern matching rules defined by domain experts. A rule R, when applied to a sentence, yields *True* if at least one pattern is successfully matched in the sentence, said to be *covered* by the rule, and *False* otherwise. Table 13 describes the number of rules per indicator in each collection.

Because rules from multiple rule categories can cover a sentence, the notion of a *rules-set* becomes necessary to enforce a *partition* of the corpus. The FlagIt LES supports seven rules-sets, resulting in a 7-split partition: P OR Sp, N OR Sn, Sp AND N, Sn AND P, Sp AND Sn, P AND N, and Null. The last is when no rule (from any of the four categories) yields *True*. OR in a rule such as P OR Sp indicates that only a rule from P or Sp (or both) covers a sentence. AND in a rule such as Sp AND N indicates that the sentence must be covered by a rule from Sp, as well as from N. Where applicable, Sn takes *precedence* over N and Sp over P. For example, if a rules-set such as (P AND Sp) AND Sn fires, it is considered equivalent to the rules-set Sp AND SN firing.

[30] See `https://spacy.io/docs/api/annotation` for a description.

Table 13: Number of rules defined by domain experts per category.

	Incall	**Outcall**	**Movement**	**Risky**	**Multi-girl**
P	5	14	37	21	9
N	6	4	7	2	0
Sp	3	3	21	1	0
Sn	7	10	40	5	11

Because of the specified precedence, the rules-sets described above are such that a sentence will be covered by exactly one of them, yielding a partition over a sentence corpus. We illustrate how such a partition can be used in a weakly supervised setting to obtain labeled data that can then be used in a minimally supervised machine learning setting.

Furthermore, in addition to producing *explainable outputs*[31], the LES also plays an important role in *training set* (and by extension, *evaluation set*) *construction* by following a *weakly supervised random sampling* scheme. Training set construction is an important problem that plagues many real-world data mining applications that depend on training adaptive systems for good performance [60]. In problems with enormous class skew, such as ours, simple random sampling is not effective. A second problem is that sentences flagged with a positive indicator are often flagged due to a combination of reasons, which makes acquiring a *diverse* training set an important requirement for adequately representing the distribution of data.

We construct a small but diverse training set per indicator by randomly sampling *labeling budget*/7 sentences from each of the seven rules-sets in the partition. In keeping with the minimal supervision goals of this paper, *labeling budget* was restricted to a small number (140), meaning that, in the typical case, 20 sentences were labeled by a user for each rule-set. The only exception is when a rules-set did not cover 20 sentences, in which case we used a form of iterative, re-distributive sampling whereby each sentence covered by that rules-set was labeled, and the rest of the budget was divided between the remaining rules-sets that had more than 20 sentences[32]. In the worst case, the scheme would be reduced to sampling all 140 sentences from the Null rules-set but this never happened in practice for any of the five indicators in FlagIt.

Minimally Supervised Machine Learning While a judicious execution of the lightweight expert system can yield competitive results, as we illustrate below, performance can be significantly improved by incorporating machine learning methods in the pipeline. State-of-the-art machine learning is not minimally supervised, requiring immense labeling effort in the case of deep learning architectures. Another form of manual effort is crafting viable sets of features.

Two options to considerably mitigate or even eliminate both labeling and feature crafting effort are semi-supervised learning and unsupervised text embeddings respectively. Below, we describe how both options can be used *in conjunction* with the lightweight expert system to improve the performance of FlagIt with minimal user effort.

[31]The explanation is implicit and comes from documentation behind the rule that got fired in matching the pattern.

[32]The process is iterative because a rules-set may not now cover the *new* local labeling budget (by definition, greater than 20).

A recent system, *fasttext* [28], released by Facebook Research under an open license has proven to be more successful than widely used models like paragraph2Vec [16], was integrated into FlagIt. The fasttext model has several advantages that make it particularly suitable for the HT domain. In particular, as the name suggests, it is fast by virtue of an optimizations employed in its codebase, allowing rapid prototyping and exploration of embedding hyperparameters such as vector space dimensionality. A second advantage is its ability to generalize to unseen sentences and words with considerably more robustness than alternative document representation models.

Since the weakly supervised random sampling scheme described earlier for constructing the evaluation set for each indicator is extremely small compared to the whole corpus, we posit that *semi-supervised learning* can be used to leverage this body of unlabeled text to further improve the results [71]. We integrate a simple, and classic semi-supervised scheme in FlagIt: first, we train an initial (indicator-specific) classifier on the perfectly labeled training set, and use the classifier to generate a positive label *probability* (for that indicator) for each unlabeled sentence in the overall corpus. Next, we pick some percentage of unlabeled sentences from the 'extreme' ends of the probability distribution and heuristically label these sentences to re-train the classifier. This process can be iterated a certain constant number of steps, or till a convergence criterion is met.

Experiments FlagIt is designed to be both tunable and extensible to new indicators, and in keeping with its secondary goal of being a broader experimental platform for indicator mining from webpages, has several alternate algorithms, embeddings and baseline methods directly integrated into it. In addition to characterizing the performance of FlagIt compared to several competitive baselines, the evaluation goals in this section also include characterizing its generalization potential by plotting indicator-specific learning curves on the evaluation data. Our choice of baselines helps us to analyze the contributions of each of the individual components in FlagIt, including the choice of unsupervised text embedding (fasttext), the use and limits of weak supervision, and the performance gain achieved via semi-supervised learning.

Unlabeled Human Trafficking Corpus: The first version of FlagIt was executed on a 54 GB online sex trafficking corpus crawled over the Web. The corpus is diverse: the total number of unique URLs is 1,000,165 while the number of unique sentences is only 1,919,339, even though webpages contain many more than 2 sentences. Thus, many sentences are repeated across ads.

The total number of unique words (vocabulary) across the sentences is 735,741, which illustrates both that many words are repeated, and that there is a large number of low frequency terms, some of which turn out to be important for the indicator mining problem (e.g., *bbbjtcws, cbj*[33], *facetime*). Overall, the document frequency of words exhibited a clear power-law (i.e. long tail) distribution.

Metrics and Evaluation Protocol: We construct the evaluation set using the method outlined above. Details of the evaluation set are provided in Table 14. We use stratified sampling to split the evaluation set into a training and test set in each experimental trial, with 65% used as the training percentage (about 100 of the 140 sentences). We use the *precision*, *recall* and *F1-Measure* metrics to measure system effectiveness. With one exception (bag of words baseline), we compute the best F1-Measure achieved for each of the baselines over five[34] independent training/testing trials. We report the average of these F1-Measures and plot precision-recall curves.

Baselines: We considered and implemented several baseline strategies to understand the effects

[33]The first two of these examples are short acronyms for certain sex services.

[34]Due to an implementation detail, bag of words is evaluated over ten trials.

Table 14: Profile of the rules-set partition (*part.*) generated via the lightweight expert system on the unlabeled corpus for each indicator (absolute counts in thousands (percentage)), and the manually labeled positive/negative indicators in the sampled evaluation set (*eval.*). *NA* stands for *Not Applicable*. For space reasons, we omit *Outcall* details.

Rules-set	Incall		Movement		Risky		Multi-girl	
	part.	eval.	part.	eval.	part.	eval.	part.	eval.
P OR Sp	25(1.2)	19/1	5.8(0.3)	4/18	50.7(2.6)	25/5	.72(0.04)	21/9
N OR Sn	55(3.0)	5/15	69.9(3.6)	2/20	0(0)	NA	505(26.2)	0/40
Sp AND N	3.4(0.2)	13/7	.42(0.02)	20/2	0(0)	NA	0(0)	NA
Sn AND P	.47(0.02)	11/9	.23(0.01)	5/17	0.04($\approx$ 0)	12/18	.90(0.05)	15/15
Sp AND Sn	0.06 ($\approx$ 0)	16/4	0.01($\approx$ 0)	8/0	0(0)	NA	0(0)	NA
P AND N	10(0.6)	20/0	.92(0.05)	7/15	.18(0.01)	NA	0(0)	NA
Null	1820(95)	0/20	1840(96.0)	0/22	1870(97.3)	0/80	1410(73.7)	1/39

of various FlagIt components:

- **Group of Rules-Sets (GoRS):** The GoRS baseline, implementable in SWRL, is the only non-adaptive baseline we consider, and is the closest equivalent of a standalone, lightweight but *optimistically configured* expert system. The baseline is implemented by further *grouping* the rules-sets in Table 14 into a *binary* (rather than a 7-set) partition to predict the label of a sentence based on which group it is covered by[35]. We consider five intuitive groupings, including a *precision-centric* group (P OR Sp is solely assigned to the positive-label group, while every other rules-set is in the negative-label group), a *recall-centric* group (Null and N OR Sn are solely assigned to the negative-label group) and several midway alternatives.

- **Bag of Words (BoW):** We use the classic BoW baseline by using the entire corpus of 1.9 million sentences to compute IDF (Inverse Document Frequency) statistics, followed by k nearest neighbors majority-vote classification, using cosine similarity on TF-IDF vectors in the training set. We report results obtained over the best performing k (over the set {1,5,10}) for each indicator.

- **Simple fasttext (Sim-ft):** The simple fasttext baseline is identical to FlagIt in every way, except that it does not apply semi-supervised learning. This baseline helps evaluate the benefits of semi-supervised learning for each indicator.

- **Paragraph2Vec (PV):** The PV baseline was one of the first skip-gram neural network-based document embedding models published in the research literature and has obtained state-of-the-art results in the past [16]. Since PV is unsupervised, we apply it to all labeled and

[35]By virtue of being a partition, exactly one group will cover a sentence.

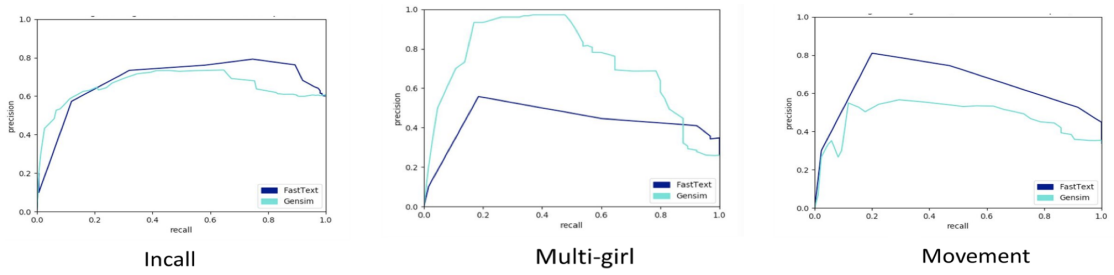

Figure 14: Precision-recall curves of the PV baseline (*gensim*) against Sim-ft (*fasttext*). *Risky* and *Outcall* (omitted herein) are qualitatively similar to *Movement*, with Sim-ft exhibiting a performance advantage.

unlabeled sentences and use the sentence vectors in the training and test sets as feature vectors.

Implementation and Setup: All text preprocessing, rules-set partitioning and representation learning programs were executed on an Amazon EC2 r3.xlarge machine with 4 computing units and 30.5GB memory. All other experiments, especially those concerning trials on the evaluation set, were conducted locally. We implemented all our code in Python, and all packages used in FlagIt are open-source. For the PV baseline [16], we use the implementation in the gensim package . We use the spaCy package for implementing and executing the rules-sets in the lightweight expert system. For the machine learning classifiers, we use Python's scikit-learn library.

We configured fasttext to learn 20-dimensional vectors. For the PV baseline, we trained 12 vector space models over the following parameter combinations: {15, 20} for dimensionality (*dim*), {2, 3, 4} for minimum word frequency (*min_ count*), and {40, 60} for number of iterations (*iter*). Using a training split of the *incall* dataset for development, we picked the best three[36] models i.e. {dim:20, min_count:2, iter:40}, {dim:15, min_count:2, iter:40}, and {dim:20, min_count:4, iter:60}. Concerning the trained machine learning classifier in all experiments except those involving BoW (which uses kNN), we consider both logistic regression and random forest classifiers, and pick the one that performs best.

Table 15: F1-Measure scores for the evaluated approaches.

	Incall	**Outcall**	**Movement**	**Risky**	**Multi-girl**
GoRS	0.79	0.76	0.64	0.77	**0.73**
BoW	0.75	0.78	0.70	0.67	0.65
WS-ft	0.76	0.71	0.50	0.41	0.41
Sim-ft	0.83	0.80	0.73	0.73	0.64
PV	0.77	0.80	0.64	0.70	0.58
FlagIt	**0.85**	**0.89**	**0.77**	**0.79**	0.65

[36]Primarily because of disk space limitations.

Results Table 15 provides F-measure scores averaged over 5 (in the case of BoW, 10) random trials. On four of the five cases, and also the average, FlagIt achieves the best F-Measure performance; the only exception is *multi-girl*, where it is outperformed by GoRS, which emerged as the next best baseline. We believe that the decrease in FlagIt's performance on *multi-girl* may have been due to the importance of numeric features in the text, which text embeddings cannot easily understand or capture.

Concerning the good performance of GoRS over rival statistical approaches, we find that the 'best' group does not change much over the five indicators. For example, for four of the five indicators, the recall-centric group works best, and for the fifth indicator, is second best by a narrow margin. This finding also highlights one limitation often found in weak supervision approaches: they tend to do better on recall than on precision. In the case of GoRS, recall was consistently higher than precision.

Given the good performance of GoRS, we considered the following *weak supervision* experiment: first, use the recall-centric group in GoRS to label every sentence in the unlabeled set as positive or negative; then, train FlagIt on the fasttext feature vectors using the weakly supervised labels, and test the resulting classifier using the entire evaluation set. The single point[37] F1-Measure scores obtained by this baseline (*WS-fasttext*) are illustrated in Table 15. Finally, comparing Sim-ft with FlagIt in Table 15 we find that semi-supervised learning leads to consistent improvement by a few percentage points. Further analysis on precision-recall results, not reproduced herein, showed that it was precision that improved, usually by reducing the number of false positives.

Although we have framed the problem of mining indicators in inferential terms, a sobering fact is that the interaction of indicators in the real world, and the extent of various indicators being present in ads, is not fully understood. This is because large-scale research on this matter has been necessarily limited by the lack of good data (especially with semantics) in the online sex ad domain. The results in the previous section showed that (1) the (explainable) rules specified in GoRS are almost as good as (non-explainable) machine learning, although it is expected that machine learning performance could improve with more labeled data, (2) the absolute F1 performance of GoRS is above 60%, and in many cases, above 75%. Together, these observations illustrate that it may be worthwhile to conduct a *computational social science* (CSS) study on *indicator correlations*, with several caveats in mind, the most important of which is that the rules are not perfect.

We conducted such a study using the lightweight expert system (LES) in FlagIt by recording which category of rules (e.g., positive, strongly positive) were fired for each unique sentence extracted from the 54 GB corpus, followed by a correlation analysis. As explained earlier, indicator mining suffers from the class skew problem i.e. positive rules are never fired for the vast majority of sentences. This can be problematic when computing standard correlations, since the negative samples will skew the metric. Thus, research hypotheses must be framed carefully, and are usually conditional on at least one indicator rule getting fired. For example, we used the data to answer questions such as the following:

1. Which indicators co-occur most often with a (positively flagged) movement indicator?

2. When at least one of movement, incall, outcall, risky and multi-girl positive rules fire, what are the cross- correlation coefficients in that sub-sample?

[37] Since the entire evaluation set is used for testing, the process is deterministic, unlike with the other baselines.

3. Are risky activities correlated with movement, given that they are both believed to be correlated with a human trafficking signal?

4. Are certain indicators more correlated with bigger cities than others?

In some cases, such as the last question, indicators have to be combined with explicitly extracted entities like locations. Although any such findings must be treated with caution, especially without support from a field-level sociological undertaking, we hope that the *structure* of the questions and study presented herein still illustrate the promising benefits of modeling and semi-automatically mining implicit semantic tags like indicators from large corpora in socially consequentially and difficult domains like illicit sex advertising. In ongoing work, along with improving FlagIt and using it for analysis, we are also extending it to automatically mine promising indicators in other investigative domains, such as securities fraud.

Summary Evaluated on five diverse indicators against a range of competitive (adaptive and non-adaptive) baselines, FlagIt outperforms the best baseline on the F1-Measure metric by 2-13% on four of the five indicators and is always the best of all adaptive baselines. Finally, we also demonstrate the potential of FlagIt for conducting large-scale computational social science studies involving indicators in the illicit sex ad domain. Although the study or its results is not the main focus of this paper and we only briefly describe its structure, to the best of our knowledge, an indicator correlation study of this scale (encompassing millions of sentences and hundreds of thousands of webpages) has not been possible before in the online illicit sex ad domain, which provides a proof-of-concept illustration that FlagIt can provide promising insights both to computational social scientists and investigators.

3.3 Investigative Search

Interacting with knowledge graphs (KGs) like Google Knowledge Graph [62], Microsoft Satori [58], Freebase [10], DBpedia [9], and YAGO[64] has become a daily experience for the public through tools like web search (Google, Bing) and personal assistants (Siri, Alexa). These KG interfaces fall on a spectrum from keyword-based entity search to structured query languages to natural language question and answer. Keyword search is eminently usable after users have been exposed to it for almost two decades, but it lacks the expressiveness needed to achieve high precision results. Structured query languages provide expressiveness but require training and their recall is brittle in the face of even small ambiguities and differences in vocabulary. Natural language question and answering is also very usable, but it is difficult to implement in practice and often falls back to mapping to a structured query language. There's still much room for improvement on query precision and recall for all approaches along this spectrum.

These approaches' precision and recall are further diminished in the face of noisy KGs. All KGs have some noise, usually in the form of inaccurate, inconsistent, stale, or missing facts. KGs built primarily using information extraction techniques, which are less than perfect, face noise of another magnitude, in particular over novel domains where it can't be compensated for or corrected using outside sources. The last chance to address this noise is while evaluating queries. For query strategies over these noisy KGs to achieve high precision and high recall, they must be able to compensate for noise in novel ways, because the noise only aggravates keyword search's imprecision and structured query languages' recall brittleness.

To overcome the limitations of these two approaches: strict pattern matching in SPARQL[56] and imprecise keyword search over a noisy KG, we propose methods for hybridizing their strengths beyond existing entity search and structured search engines. The process starts with a hybrid KG (HKG) that combines each entity's triples, enriched with their extraction provenance, and the source documents it was derived from. From there

- We create an entity-based search engine from an HKG by indexing each entity's triples with text of its source document in a NoSQL document store.
- We present a scheme for structured search to map triples to a set of indexes that represent the product of predicates and categories that capture extraction provenance.
- We show how to combine and weight matches across different extraction provenance category indexes to efficiently incorporate measures of relevance and confidence for extractions without modifying the document store's engine.
- We provide query relaxation, rewriting and expansion strategies for translating simple structured SPARQL queries in to the document store's native syntax to create complex hybrid structured and keyword search queries over these indexes.

We first present the desiderata of robust systems and query rewriting strategies and then present details of our approach, including the strategies for rewriting queries and indexing the data in Elasticsearch[13], the NoSQL store we use in our implementation. Our evaluation uses a benchmark developed to model questions a law enforcement analyst might ask a KG built from escort ads crawled on the internet for the human trafficking domain. The benchmark is used in two ways. First, we perform an ablation study to understand the compound effects of our strategies on query performance in terms of Mean Average Precision (MAP), index size, index time, and query time. Second, we show that our approach outperforms another similar hybrid approach and a learn to rank approach, achieving a MAP of 85%.

Desiderata for Robust Systems Systems deploying robust strategies should share the following desiderata at both index time and query time.

- Usability: A system should be immediately useful with minimal user configuration. It should not require a software engineer to operate. Someone with appropriate domain knowledge should be able to write queries without having to understand the strategies employed.
- Traceability: It should provide the user with the supporting decision context and query plan to explain why documents were retrieved over others. A document's rank without the ability to drill down into how the score was calculated is insufficient. This is important when building trust and reliability in a system, especially under uncertainty [24].
- Flexibility: As the system is applied to new domains, it faces new challenges. Few, if any, strategies will fall into the winner takes all category across domains. Giving control over strategies the system uses will allow users to overcome unforeseen limitations. The system should support an extensible, pay as you go library of strategies, allowing users to add, remove, enable and disable strategies as needed.

Desiderata for Robust Strategies The strategies employed by a system for querying a noisy KG should have one or more of the following properties.

- Robust to missing extractions: KGs built from information extraction techniques won't perfectly extract all knowledge from its source material. Natural language processing efforts like named entity recognition and other machine learning techniques for extracting information face challenges when training data is hard to come by, language models are irregular, information is obfuscated, or the knowledge is domain specific. The source material is still a rich source of information at query time. Adding keyword search for matching against both extracted entity labels (New York vs New York City) and the source document can help bridge the gaps between information extraction, entity resolution and aligning search terms against the KG. Introducing keyword search is not without its own set of challenges.

- Robust to over-constrained queries: KGs incorporating new information operate under the open world assumption. SPARQL queries usually follow a closed world assumption. As users express all their information requirements in SPARQL, their queries can quickly become over-constrained. Requiring matches satisfy all the information requirements under the closed world assumption can exclude potential matches because the matching information wasn't extracted or keyword search fails. To improve the match recall, some requirements can be relaxed. Instead of requiring the user to express the complexity of which requirements can be relaxed and how, query strategies should automatically assist the user to reasonably relax constraints to retrieve the best results possible. Users should also have control over the relaxation. Unless expressly marked as not relaxable by the user, required clauses should be relaxed to optional clauses. For all clauses in a query, a document should be a match if at least one of those optional clauses is satisfied. Additional matching clauses should aggressively contribute to the document's score and rank.

- Robust to search term confusion: The greater specificity of structured query languages allows for users to ascribe semantic meaning to values and their relationships limiting the need for the system to infer the same. Introducing keyword search and ranking without careful consideration can quickly defeat that specificity by matching against the document for values that appear in the range of many predicates. Some clauses should not support keyword search at all, such as numeric values like weight. Others, after applying to tokenization to their constraints, should have additional stop words to avoid inappropriate partial matches like emails for their domains, e.g. 'com' 'org', or should weight much higher exact term matching.

- Robust to information irrelevance: Facts in the KG can be irrelevant to a query for many reasons. Information extracted from boiler plate forms, e.g. headers and footers, around the content of a web page are usually less relevant than the content. Take a blog post. The mailing address of the company hosting the site found in the footer is less relevant than the people, places, and things mentioned in the blog post. This means query strategies should understand there is an implicit relevance hierarchy over the zones of the web page. The information extracted is often of decreasing relevance from title, to body, to the rest of the page, so structured and keyword matches should be weighted appropriately. Furthermore, there is a hierarchy of relevance over the type of information, and a query strategy should

weight it appropriately. Matches against hard identifiers like phone numbers, emails, and full names should be scored higher than soft categorical identifiers like hair and eye color when ranking results.

- Robust to poor quality extractions: Different information extraction techniques have different precision and recall trade-offs and serve different roles. Some may be chosen for their high recall and simplicity like dictionaries and regular expressions. Others are chosen for their high precision, like spaCy rules and the Stanford NER, but take time to write or train. When high precision, high confidence extractions are present, a query strategy should favor them over lower precision, lower confidence extractions. Some high quality extractions also include additional metadata like units or context. Query strategies should account for alternate formulations and synonyms by performing query expansion and unit conversions to increase a likelihood of a match.

A system that can implement a set of strategies that achieve these properties while maintaining usability, traceability and flexibility should be able to overcome the noise found in a KG to rank and retrieve documents with higher precision and recall while building sufficient trust in its users.

Approach Our approach is called Pinpoint. The three building blocks that make up Pinpoint are: hybrid KGs (HKG), indexing strategies, and query reformulation strategies. To meet the desiderata described above, Pinpoint takes as input a HKG, consisting of source documents and triples enriched with extraction provenance information. It then specifies how to map the documents and triples according to their provenance to a set of indexes for efficient ranking and retrieval in a NoSQL document store. Finally, it reformulates user queries written in SPARQL by automatically performing query expansion, relaxation, hybridization and translation for execution against the NoSQL document store.

Rich Triples and HKGs Pinpoint takes as input a HKG where triples are grouped by entity, enriched with their extraction provenance information, and paired with the source documents they were extracted from. We illustrate this for an example advertisement in JSON in Listing 1. The JSON describes the provenance of a triple for an email mentioned in the advertisement. For an advertisement, there can be multiple emails mentioned and each email can have multiple provenance records, one for each time it was extracted. The triple has a searchable label value and a key which maps to an ID like a URI. For this email, they're the same, but, for values like city names, they can differ. Each provenance record captures its source, including the document ID, the segment or zone of the document the information was found, i.e. title, body, and, optionally, any additional context. The provenance record also captures the extraction method and a confidence score. Triple provenance information can be captured many ways, i.e. reification, quads, w3c prov. This format is designed for convenient storage in document stores.

Efficient Indexing of Rich Triples, HKGs Maintaining this extraction provenance is most important for strategies to limit the effects of irrelevant and/or poor quality extractions. It allows Pinpoint to differentiate between facts derived from high recall and high precision extractors, relevant web page body content and boilerplate, weighting them appropriately. To make sure Pinpoint

can efficiently make these comparisons, it encodes the information by indexing them in multiple indexes.

```
{"knowledge_graph":{
   "email": [{
    "confidence": 1,
    "key": "websupport@eccie.net",
    "value": "websupport@eccie.net",
    "provenance": [
      {
        "source": {
          "segment": "title",
          "document_id": "ABCDEFGHIJK",
          "context": {
            "text": " Address : websupport@eccie.net ; mis",
            "obfuscation": false
          }
        },
        "confidence": { "extraction": 1.0 },
        "method": "regex"}]}
]}}
```

Listing 1: Example Email Rich Triple

In the previous email rich triple example, its provenance recorded that it was extracted by a regular expression extractor. Regular expressions can have high precision, but low recall, especially if the email deviates even slightly from the valid syntax for an email address. Since the provenance also indicates the email was found in the title of the web page it was extracted from, it is more likely to be relevant to this advertisement.

As the number of predicates and categories grows, however, it can lead to an expensive, sharp increase in the number of indexes required. To help limit the number of predicates, Pinpoint supports mapping synonyms and related predicates to the same set of indexes, trading some specificity in the process. When configuring Pinpoint, we attempt to limit the number of segment and extractor categories. In practice, we find only a few merit distinguishing. The segments, supported extractors, and predicate mappings are all user configurable categories that must be set at indexing time. To support graceful degradation of the system at run time, any unrecognized predicate in a user query is mapped to a set of indexes created for the text segments.

In the JSON document in Listing 1 that represents our advertisement, the value for the extracted email will be added to an indexable field named `indexed.email.other_method.title.value`. The key will be indexed similarly under `indexed.email.other_method.title.key`, but without tokenization to ensure Pinpoint can do an exact lookup. During tokenization, the email value will also be subject to stop word removal, i.e, `com`, `org`. As described above, partial matches against common, low information terms like these would otherwise confuse our structured search approach by ranking irrelevant documents much higher. Even partial matches against structured search clauses like emails and phone numbers are weighted heavily because they are powerful

identifiers. The indexable subsection of the document that will be sent to the NoSQL document store for the email looks like Listing 2

```
{
  "indexed":{
    "email": {
      "other_method": {
        "title": {
          "key": "websupport@eccie.net",
          "value": "websupport@eccie.net",
      }
    }
  }
}
```

Listing 2: Example Indexed Email Rich Triple

The document stored in the search engine will also contain the section of the KG and raw provenance information for the advertisement along with the entire source document. This information will not be directly indexed but preserved for client applications to retrieve, display and reason about. This has an added storage expense, but we've found it necessary to build trust in user facing search applications.

Query Reformulation Pinpoint aims to increase precision and recall by reasoning about user queries and making appropriate choices without extensive user input. To this end, for a balance of usability of expressiveness, Pinpoint supports a subset of SPARQL instead of keyword search or natural language question and answer. It also limits queries to match against trees instead of the full graph, which is amenable to a form-driven UI as illustrated in Figure 15 and its indexing scheme. Pinpoint does not enforce any particular schema. If the schema is unfamiliar to the user or the user attempts to use predicates unknown to Pinpoint, Pinpoint gracefully degrades by falling back to keyword search as user queries become less expressive.

For application developers, Pinpoint provides the flexibility of a pluggable architecture for adding, removing, enabling, and disabling strategies. For traceability, Pinpoint also provides the user with feedback, including the rewritten SPARQL query, the translated ES, and the relevant matching clauses for each document, all with generated IDs for traversing back to the original SPARQL query.

We will now discuss the strategies we implemented to rewrite SPARQL queries, translate them into hybrid structured and free text queries, and then further rewrite and optimize them against the indexes built for the KG. In Listing 3, we provide a sample query generate-able by the UI in Figure 15.

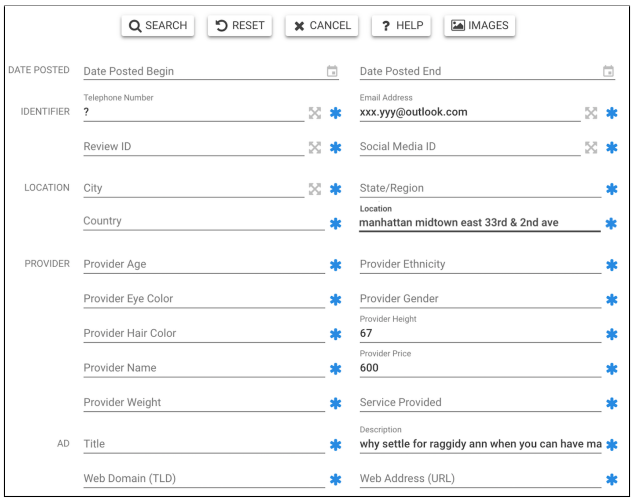

Figure 15: Example App Form To Generate SPARQL Query

```
PREFIX qpr:<http://istresearch.com/qpr>
SELECT ?ad ?phone
WHERE
{
  ?ad a qpr:Ad ;
      qpr:email 'xxx.yyy@outlook.com' ;
      qpr:price '600' ;
      qpr:height '67' ;
      qpr:street_address 'manhattan midtown east 33rd & 2nd ave' ;
      qpr:phone ?phone ;
      qpr:content ?content .
      FILTER CONTAINS(LCASE(?content), 'why settle for raggidy ann when
          you can...')
}
```

Listing 3: Example Benchmark SPARQL query

Preprocessing Queries Query rewriting begins with preprocessing the query's SPARQL semantics and constraints. Pseudo-code has been provided. Preprocessing is outlined in Listing 4. For traceability, Pinpoint tags each clause and reformulation with a unique id, but that process has not been documented for brevity.

```
QueryPreprocess(Q, ontology, expandersByType, consistencyByType)
  (s, w) := Q.select, Q.where
  C := w.clauses
  if queryRelaxationEnabled
    C := QueryRelaxation(C)
  varTypeMap := ComputeVarTypeMap(C, ontology, {})
  C := QueryExpansion(C, varTypeMap, expandersByType)
  C := QueryConsistency(C, varTypeMap, consistencyByType)
  w.clauses = C
```

Listing 4: SPARQL Query Preprocessing

Pinpoint relaxes required clause constraints and filters by introducing OPTIONALs as shown in Listing 5 to protect against over constrained SPARQL queries. The default behavior is to relax all required constraints to maximize recall, but, in systems that need additional query expressiveness and precision, this can be disabled. Intuitively, the scoring mechanism ranks documents higher that match these required clauses by virtue of matching more clauses. As more clauses are added, however, scoring becomes complex and matching many non-essential clauses could outscore matching just essential clauses. Giving users flexibility as they refine their searches is important to avoid frustration.

```
QueryRelaxation(C)
  C' = ()
  for c in C
    if isOperator(c) and !isFilterNotExistsOperator(c)
      c := QueryRelaxation(c)
    if !isOptional(c)
      c := OPTIONAL(c)
    C'.push(c)
  return C'
```

Listing 5: SPARQL Query Relaxation

After relaxing, Pinpoint maps variables to types in Listing 6. This can be derived from predicate domain and ranges in the KG ontology, but also a user provided dictionary. It is done breadth first so Pinpoint can type variables before recursing into FILTERs, sub graphs, and other parts of the SPARQL query, SELECT, GROUP BY where type information may not be inferable.

```
ComputeVarTypeMap(C, ontology, varTypeMap)
  for c in C
    if isTriple(c)
      (s, p, o) := (c.subj, c.pred, c.obj)
      if isVariable(s)
        varTypeMap(s) := ontology.domain(p)
      if isVariable(o)
        varTypeMap(o) := ontology.range(p)
    elif isOperator(c)
       varTypeMap :+ ComputeVarTypeMap(c, ontology, varTypeMap)
    return varTypeMap
```

Listing 6: SPARQL Variable Type Mapping

Based on the predicate range types in triple clauses and the variable types in FILTER operators, Pinpoint applies a series of query expansions to object literal and FILTER argument constraints as shown in 7. This includes, but is not limited to, user-provided dictionaries of synonyms and unit conversions. This helps overcome the brittleness of SPARQL over an incomplete KG that is missing extractions or is not aligned to the user's query vocabulary. In the example query shown in Listing 3, a height of 67 describes a person measured in inches, but not explicitly. In case the knowledge graph extractors only produced facts containing height in centimeters, the system can rewrite the query by introducing a UNION of the clauses as in Listing 8. Pinpoint also supports applying user defined functions for cleaning up search terms with the QueryConsistency operation, but the details are not important.

Parameterizing Queries Parameterizing queries occurs without rewriting the SPARQL explicitly. The parameters affect how a query is further translated and rewritten against the NoSQL document store. These parameters allow the system to enable or disable strategies like differentiating between extractors, extraction zones, and the minimum number of optional clause matches for a given query. By default, differentiating between extractors and zones is enabled and the minimum number of optional clause matches is 1, unless a required clause is present, then it is 0.

```
QueryExpansion(C, ontology, varTypeMap, expandersByType)
  C' = ()
  for c in C
    if isTriple(c)
      (s, p, o) := (c.subj, c.pred, c.obj)
      E := expandersByType()
      expandedC := ()
      for e in E
        expandedC.pushAll(e.expand(s, p, o))
      c := UNION(c ++ expandedC)
    elif isOperator(c) and isInFilter(c) and isComparison(c)
      (f, A) := (c.function, c.arguments)
      expandedC := // function and operator specific reasoning that
         relies on varTypeMap
      c := OR(c ++ expandedC)
    elif isOperator(c)
      c := QueryExpansion(c, ontology, varTypeMap, expandersByType)
    C'.push(c)
  return C'
```

Listing 7: SPARQL Query Expansion

```
?ad a qpr:Ad ;
{
  { ?ad qpr:height '67' . }
  UNION
  { ?ad qpr:height '170 cm' . }
}
```

Listing 8: SPARQL Query Expansion Units UNION

Generating Queries Generating the final queries involves three stages, mapping and weighting clauses to indexes or fields, translating SPARQL into native NoSQL document store query syntax, and rewriting the translated query to support hybrid structured and keyword search. For our approach we chose Elasticsearch (ES), but analogous capabilities exist in other document stores like Apache Solr. The strategies for mapping, weighting, and hybridizing queries are applicable to the other engines as well. To be consistent on terminology in the section, an ES index is actually a collection of inverted indexes called fields for a set of documents categorized by type. ES's structured search is implemented by matching clauses against one or more fields and combining the set of documents in the search results from each field.

```
// User defined or Pinpoint derived inputs available to all generation
   steps
DATA := varTypeMap, predFieldMap, fieldWeightMap, predQueryMap,
   typePredMap

GenerateESQuery(Q)
  w := Q.where
  C := w.clauses
  return SEARCH(GenerateESClauses(C))
```

Listing 9: SPARQL to ES Query Generation

```
GenerateESClauses(C)
  (musts,mustNots,shoulds,filters):=(),(),(),())
  for c in C:
    if isFilter(c)
      c = GenerateESFilter(c)
      if isNotExistsFilter(c)
        mustNots.push(c)
      elif !isTextFilter(c)
        filts.push(c)
      elif isOptional(c)
        shoulds.push(c)
      else
        musts.push(c)
    else
      if isTriple(c)
        c = GenerateESClause(c)
      else
        c = GenerateESClauses(c)
      isOptional(c)
        shoulds.push(c)
      else
        musts.push(c)
  if isUnion(C)
    return DISMAX(shoulds)
  msm := |musts| = 0 ? 1 : 0
  return BOOL(musts,mustNots,shoulds,filters,msm)
```

Listing 10: Compound SPARQL Clauses to ES

After preprocessing, Pinpoint reformulates a SPARQL query as described in GenerateESQuery in Listing 9. As input, it expects the query's clauses have been mapped to a tree. For simplicity, the pseudo code assumes each tree node is either a triple graph pattern clause, a filter clause, or a SPARQL algebraic operator like UNION or FILTER which has children. Handling graph pattern joins is out of the scope of this paper. Pinpoint expects the application developer has configured it with a predicate to field map, a field to weight map, a predicate to ES query type map, and a type to predicate map. The variable to type map is computed during preprocessing.

Pinpoint visits each clause in the WHERE, as outlined in the GenerateESClauses method in

Listing 10, recursively rewriting them in a depth first manner. When it reaches a triple clause with a literal object, it translates it into a match according to GenerateESClause in Listing 10. Pinpoint starts by mapping the predicate to a set of fields that encode predicate, type, extractor, and zone information. Pinpoint doesn't task ES to take into account the extraction confidence or relevance score when scoring queries, only weight according to these categories encoded into the field names.

Some predicates are additionally mapped to the text extracted from the source documents, but predicates with numeric and date ranges are not. If the predicate or variable matches only against the extracted text fields like `title`, `description` or `content`, like the `FILTER CONTAINS` clause in Listing 3, those get an additional boost because those fields are not being relied on as a fall back when structured fields don't match.

Pinpoint then translates clauses by field to ES syntax. Most clauses are mapped to the ES `match` query. Depending on specificity, Pinpoint can use ES's `match phrase` functionality to impose an order on the terms, like in the `title`, and score more heavily search terms in close proximity in the `body`. For clauses with types like `email` that map to fields like `indexed.email.other_method.title.key` that index values without tokenization, ES implicitly treats them as `term` queries, which match exactly, because ES applies the same (lack of) tokenization at query time to search terms as indexed terms.

```
GenerateESClause(c, isTextFilter:=False)
  (s, p, o) := (c.subj, c.pred, c.obj)
  if isLiteral(c.obj)
    M := ()
    for f in predFieldMap(p)
      w := fieldWeightMap(f)
      q := predQueryMap(p)
      terms := |o.tokenize()|
      msm := |terms| > 5 ? |terms| / 2 + 1 : max(1, |terms| / 2)
      if isMatch(q)
        if isTextFilter
          w := w * 5
         m := MATCH(f, o, w, msm)
      elif isMatchPhrase(q)
        if isTextFilter
          w := w * 10
        m := MATCHPHRASE(f, o, w, msm)
      M.push(m)
    return DISMAX(M)
  else
    // optionally add EXISTS for fields to support binding variables
    shoulds = ()
    for f in predFieldMap(p)
      shoulds.push(EXISTS(f))
    return BOOL(shoulds)
```

Listing 11: SPARQL Triple Clause To ES Match Query

Pinpoint combines these match queries into one sub query for ES to evaluate. When translating SPARQL to ES, Pinpoint must consider the ranking semantics. ES's ranking semantics interpret

should like you would keyword search: more is better. To achieve this, ES divides any should match scores by the number of clauses. This makes reasoning about weighting fields complicated because Pinpoint doesn't map each predicate to the same number of should clauses. For example, the street address clause in Listing 3 could be mapped to more than twenty fields, based on city, state, address, extracted text, extractor type and zone, but the height clause is only mapped to four extraction fields. Unless the street address constraint matches every field, it will get penalized. The alternative is to use Lucene's Disjunction Max query over the fields, so that only the best match for a clause contributes its full score to the ranking. The case could be made that a match against a fact derived many times by multiple extractors or in multiple zones should boost the score, but this is accomplished in part by TF/IDF by the value appearing multiple times in the document field according to ES. For simplicity, we just choose the best match.

Pinpoint translates FILTERs according to GenerateESFilter in Listing 12. There are two considerations. If the filter operates on text, Pinpoint treats it like matching a constraint. If anywhere in a filter, text is matched, Pinpoint must add the whole filter to should clauses. Otherwise the match will not contribute to the document's score, because ES doesn't score bool filter clauses.

```
GenerateESFilter(F)
  (musts,mustNots,filters,shoulds):=((),(),(),())
  if isOperator(F)
    for f in F
      f = GenerateESFilter(f)
      if isNotExistsFilter(f)
        mustNots.push(f)
      elif !containsTextFilter(f)
        filters.push(f)
      elif isOptional(f)
        shoulds.push(f)
      else
        musts.push(f)
    msm := |musts| = 0 ? 1 : 0
    return BOOL(musts, filters, shoulds, should_nots, msm)
  else
    f := F
    if isTextFilter(f)
      v := findVar(f.arguments)
      l := findLit(f.arguments)
      p := typePredMap(varTypeMap(v))
      c := GenerateESClause((nil, p, l), true)
    else
      c := GenerateESRangeQuery(f)
    return c
```

Listing 12: SPARQL FILTER to ES Filter Query

After mapping the clauses to ES queries, the GenerateESClauses adds them to an ES bool, unless it's translating a UNION. Required clauses are added to the must of the bool, OPTIONAL clauses to should, NOT EXISTS clauses to must not, and FILTER to filter, unless FILTER has been relaxed to OPTIONAL. The default behavior for ES doesn't preserve Boolean semantics

precisely. It requires that for a document to match a `bool` query, $\forall m \exists s(m \in must \wedge s \in should \wedge satisfies(doc, m) \wedge satisfies(doc, s))$. This translates to all required clauses must match and, in addition, at least one optional clause must match. To ensure the semantics are equivalent with SPARQL, the minimum should match option for the `bool` query should be set to 0.

Pinpoint must also consider the Boolean semantics and ranking semantics when translating algebraic operators like `UNION`. Naively translating a SPARQL `UNION` into a `bool` query with a `should` clause satisfies Boolean semantics. However, if we introduce a `UNION` during query expansion and the document matches only the original inch `height` clause in Listing 8, the match will only contribute half of its score (less if ES's query coordination is enabled) and the document would be penalized in its final ranking even though the clause was properly satisfied. To keep weighting consistent, we use the Disjunction Max query so that only the best match contributes its full score to the ranking.

Executing Queries The generated query is then sent to ES for evaluation. The top-n scored documents are returned along with their query plan and highlighted matches structured and labeled according to the original SPARQL query using the annotation capabilities of ES. It is not included in the scope of this paper to describe how variables are bound to values in the documents.

Evaluation *Use Cases and Metrics:* The KGs built for evaluation consist of information extracted from 90,000 webpages selected from web domains relevant to human trafficking. 101 point fact queries were written in English by subject matter experts from DARPA MEMEX government and NGO partners. These queries were then translated in to SPARQL by a a contractor in charge of the evaluation. The webpages and queries are considered sensitive data and are not available outside of the DARPA MEMEX program. For evaluation, a system is allowed to return up to 500 ranked document IDs per query. The IDs are evaluated using Mean Average Precision (MAP) against ground truth generated by subject matter experts as well.

Experimental Setup: The software used to build, index and query the KG is available on GitHub and is written in Python. This includes a framework for extracting information from webpages and Pinpoint. The document store used was an Elasticsearch 2.4.1 cluster of 5 machines with 32GB of RAM for heap and 32 cores each. The 90,000 webpages were indexed using 5 shards without replicas.

Internal Baselines: We will perform an ablation study over the query rewriting strategies and indexing methods of our system, Pinpoint. We will demonstrate their efficacy by introducing them iteratively, refining a baseline system to create the following new systems.

- STRICT: Strict SPARQL semantics, i.e. exact predicate term and filter matching, required respected
- HYBRID: Relaxing strict SPARQL semantics and hybridizing literal matching with free text search and scoring
- OPTIONAL: Relaxing strict SPARQL semantics and rewriting required predicates from required to optional
- RELEVANCE: Weighting predicate value matches based on information content and relevance (scoring identifiers like phone nos. and emails higher than hair or eye color)

- ZONE: Indexing predicate values by extraction from the web page zone (i.e. title, body), weighting matches separately
- EXTRACTOR: Indexing predicate values and weighting matches separately by which system extracted the value (i.e. regex, spacy, landmark)
- EXPANSION: Relaxing strict literal matching for numeric values by allowing matches between ranges and measurement systems (metric vs. imperial)

The STRICT system approximates a SPARQL query engine. Predicate value matches do not contribute to the document's score in ranking, only whether an entity is included in the result set. For ranking, it implements the SPARQL CONTAINS function similar to Virtuoso's bif:contains, albeit with Elasticsearch's ranking of term matches. Each subsequently named system in the list adds the appropriately named feature and builds on all features added to previous systems. EXPANSION will correspond to the fully featured system under evaluation that is compared with the other teams. For comparison, we will also present the additional overhead for indexing time, index size, and query execution time required to support each feature.

Pinpoint Index Design: For the experiment, there were thirty different predicates we needed to support for structured search. STRICT had an index for each predicate to match exactly by term. HYBRID introduced an other index for free text matching against predicate values along with an index for matching against the source document. ZONE also categorized and indexed the document segments in four ways in descending relevance for each predicate:

- Title: the title of a web page.
- Landmark Description: the text of a web page based on rules derived by Landmark[6], a regex-based tool that learns to identify semi-structured "landmarks" in web pages. Trained by domain
- Strict Content: the text of a web page has extracted by the python Readability package with the strict option
- Relaxed Content: the text of a web page has extracted by the python Readability package with the relaxed option.

EXTRACTOR introduced indexing the KG according to the extractor, so we introduced three types of extractor categories with corresponding indexes:

- Landmark: the same regex-based tool above applied to identified values associated with other "landmarks" like those found in forms. A user must label the values output by the Landmark rules with a semantic type. Trained by domain
- Other: A catch-all for extractors including ad hoc ones using dictionaries and hand written regular expressions

After accounting for zones and extractor categories, the Pinpoint system created almost 400 indexes, known as fields in Elasticsearch. The typical predicate has at most 24 indexes, computed by taking the product of |zones| X |extractors| X |value, key|. Commonly, only eight of those indexes are

created, because if no Landmark or spaCy extraction rules exist for the predicate, they aren't necessary. Some predicates, like those that map to text-only fields or web page top level domain, have fewer. In practice, many of these indexes are still only sparsely populated and Elasticsearch can efficiently combine matches across all of them. The overhead of creating and querying them will be illustrated in the Pinpoint ablation study below.

Table 16: Performance of Pinpoint Systems with Ablation

	STR.	HYB.	OPT.	REL.	ZONE	EXT	EXP.
MAP	0.21	0.37	0.74	0.76	0.79	0.81	0.85
Ex Time	58s	91s	197s	201s	376s	464s	486s

Pinpoint System Ablation Evaluation: Table 16 shows that the STRICT system is brittle and performs poorly. The HYBRID system gains considerably over the STRICT system by matching data missed by information extractors in content fields and allowing for partial predicate value matches. Relaxing required predicates in OPTIONAL performs better by giving the system flexibility to return documents that don't match all constraints. Adding ZONE, RELEVANCE, EXTRACTOR, and EXPANSION helps address structural, semantic, data quality, and similarity issues to achieve our best result. This has overhead. ZONE and EXTRACTOR add the most to overhead because they multiply the number of field indexes in Elasticsearch for each category introduced. At query time, this also multiplies the number of fields the query must match against. OPTIONAL increases the query evaluation the most because it rewrites and executes the query N times where N is the number of predicates in the query.

External Baselines: For comparison, we have also included two other systems from the DARPA MEMEX program that were submitted for evaluation. In the results section, Pinpoint corresponds to the EXPANSION system. Team B applied similar structured search techniques using Elasticsearch. Team C applied a learn to rank approach.

External System Evaluation:

Table 17: Point Fact Query MAP

	Pinpoint	Team B	Team C
MAP	0.85	0.78	0.71

Table 17 shows our system outperformed all others on MAP, returning the relevant document in the first position for 81 queries. In a post hoc analysis of the five questions where we failed to return a relevant document, the crawled web pages either had a misspelled query term, were missing relevant information in the body, or even lacked a body. For 16 queries, Pinpoint received less than 1.0 MAP. 85% of queries were in the top 2 and and 91% were in the top 4.

The 101 point fact queries were completed in under 10 minutes. On average, the queries take 5 seconds. 89% of queries are executed within 10 seconds, however, which is within the guidelines laid out in Nielsen's Usability Engineering [50] for optimal user experiences.

Results The DIG search strategies are robust in particular to KGs built from noisy information extractions derived from web pages with potentially irregular language models. All extractions are represented in these KGs as rich triples which capture their extraction provenance. Pinpoint is user configurable and has a pluggable interface for its strategies. It has been used to query KGs for

many domains including human trafficking, illicit weapons, counterfeit electronics, patent trolls, and research trends in material science and autonomous systems. Pinpoint has also been used to index and serve a KG extracted from 100 million web pages. During an evaluation against other systems designed to query a KG built for the human trafficking domain, Pinpoint outperformed them all, achieving a MAP of 85%. As part of our evaluation we also performed an ablation study that characterized the additional overhead associated with supporting Pinpoint's index and query strategies.

Error Analysis We did a careful error analyses of the point fact questions where the correct entity was *not* retrieved as the *top* result. Table 18 provides a condensed breakdown of this analysis, including counts and potential solutions that may be integrated into the next prototype. More details are provided below. We note that, while some errors have relatively simple solutions (e.g. implementing phonetic analyzers for name misspellings), others require more sophisticated engineering of the schema functional and semantic strategies.

Importance of pseudo-identifiers: At least two error categories (1 and 6 in Table 18) involved pseudo-identifier attributes like phone numbers and email addresses. The first category is an extraction-centric problem because the queried phone number did not get extracted (even imprecisely) from the relevant page and further analysis showed that the phone number (in any format) was not present in the extracted text. This rendered all strategies, including the keyword strategy, ineffective. An inspection of the raw HTML content showed that the phone numbers in such pages were cleverly hidden and got filtered out by the text extractors.

While improvements in obfuscation detection and higher-recall extractions are the ideal solutions for this category of issues, it is unlikely that they will ever be achieved in the near-term, especially given the illicit and ever-changing nature of the human trafficking domain. Instead, a key lesson learned here is that, even with text extractors and scrapers, the raw HTML cannot be discarded but may have to be retained (despite the native difficulties in storing and indexing large HTML corpora), perhaps as part of backup infrastructure or in a secondary index, for robust query-centric performance on the knowledge graph.

In contrast, error category 6 is not due to faults in the extraction but due to the entity-centric search engine. Despite higher weights being assigned to pseudo-identifiers, the engine still made mistakes in recognizing these as highly relevant, especially when the query contained other attributes that were rare enough to be recognized as significant by the NoSQL tf-idf-based indexer[38]. A short-term solution to this problem is listed in the table, which is to continue improving weight assignments for all attributes to achieve the best empirical tradeoff. However, a longer-term solution is adaptive re-ranking on the client side, possibly with the help of user interaction.

Other related problems do not involve pseudo-identifiers but occur because a match on some field was not given enough importance by the search engine. A particular instance is error category 3, when using tf-idf scoring with the bag-of-words representation of the text extraction was too weak a signal to be useful. Given the simplicity and speed of tf-idf on the vast majority of queries that used the text field, or even queries that did not use the text field but had to fall back on the keyword strategy, we do not believe that abandoning the tf-idf for a more complex measure is the correct solution. Instead, we are modifying the search to draw not only on unigram representations, but also on bigram and higher-order n-gram bag-of-words representations. Similar to *smoothing*

[38]Because of the idf term, rare terms and extractions can sometimes have such unintended consequences.

Table 18: A condensed overview of our error analyses exercise. More detailed explanations are provided in the text. The error count reports the number of questions on which the error occurred.

Error ID	**Category of Error**	**Example of occurrence**	**Potential Solution(s)**	**Error Count**
1	Missing pseudo-hard identifiers (phone, email) in the 'ground-truth' answers	Searching for (but not finding) an ad that contains the phone 15102142367, has ethnicity Asian and location Reno	(1) Use raw HTML in the search along with text extractions; (2) improve extractors to have better recall	8
2	Spelling Differentiation	Asheerah (in ad) vs. Asheera (in query)	Phonetic analyzers on *Name* extractions	2
3	Score due to bag-of-words is too weak	Searching for (but not finding) an ad that contains the post date September 6 2016, the text string 'Lacy funsize' in the title, and the text string 'Im available for incall during the day' in the ad text	Higher-order bag of words such as bigrams, trigrams or phrase matching	6
4	Difficult disambiguations	What is the age provided in the ad that contains the name Cindy, with height 61 inches?	More robust normalizations	3
5	Systems-level problems	Text extractor did not return anything	Some recourse to original HTML document	4
6	Pseudo-identifier match not weighted highly enough	An age match ended up with more weight than a phone number match	Better weight assignment strategies to pseudo-identifiers	4
7	Difficult ad disambiguation (wrong but closely related, thus relevant)	Two escorts sharing a phone; we return the 'wrong' one at the top	More post-processing	1
8	Multiple correct/relevant answers	Multiple postings for an escort	Not really an error but entity resolution could help with the evaluation	9

models in the NLP literature, the search algorithm will assign higher scores to higher-order n-gram matches for a given query term than to unigram-only matches.

Misspellings: Misspellings constitute a common class of problems in the search engine literature. Many solutions have been proposed, including n-grams and modeling-based solutions (e.g., indexing by explicitly modeling the noise introduced by mistakes like typing, OCR etc.). We noted in our analysis (error category 2 in Table 18) that spelling errors almost always caused problems in our engine when they occurred due to differing name representations in the actual query and the document. For example, an investigator wants an ad that contains the name 'Asheera', when in reality, the spelling (on the relevant ad) is 'Asheerah'. Without significant tuning and engineering, character n-grams are not a practical solution to this problem (even higher-order n-grams like 4-grams would result in indexing of commonly occurring phrases like 'Ashe'). A more feasible modeling based solution is alternate indexing of such fields. For example, Elasticsearch offers custom analyzers and indexers for non-text data such as phonetic representations, geolocations, and vectors. In particular, *phonetic* analyzers work well for the name field, as Asheera and Asheerah have the same representation in various phonetic frameworks. Other better-known examples of such phonetic equivalences include name pairs like *Katheryn* and *Catherine*, and *John* and *Jon*.

Disambiguation: Error categories 4, 7 and to a limited extent, 8, involve different categories of disambiguation issues. An important case is unit disambiguation (e.g., height in inches vs. feet and inches), which could have been resolved through normalization if the extractions were not noisy. Unfortunately, it is very difficult to cover the space of all possible representations for a given attribute without large amounts of manually annotated training data. A second option is to modify the query itself to take into account various representations. For example, given that a height of 61 inches is being queried for (as illustrated in the table), we could potentially run 61 inches through a transformation program that converts inches to various commonly used units (for height). Although promising, writing such programs is still manually intensive; furthermore, because of strategies like the keyword strategy we could end up introducing noise during the retrieval. A full empirical study is required to map the cost-benefit tradeoff of this solution.

Error category 7 arises due to document-level ambiguity. For example, it is commonly known that escorts in the sex trafficking world often share phone numbers and sometimes copy text from each others' ads with minor changes. Sometimes, it is difficult to distinguish the relevant ad from a sufficiently similar wrong ad with the same pseudo-identifier, using the NoSQL index alone. We are currently investigating finer-grained client-side re-ranking solutions to this problem.

Error category 8 is potentially not an error category *per se* but is still problematic for actual users. The problem usually arises because there are multiple postings for an escort, typically with different time stamps or slightly different locations (e.g. Miami vs. Fort Lauderdale). Taking down (and subsequently re-posting) ads is also a common strategy employed by sex providers. It is important to note that, while such 'document sets' do not cause problems for point fact queries (assuming the set describes versions of the ad that was actually queried for), they can skew answers significantly for aggregate queries. For this reason, and also to provide a better experience to analysts in our GUI, we are investigating *entity resolution* solutions to deal with this problem [21].

3.4 Image And Face Search

Columbia University developed a large scale image and face search tool that was integrated in the frontend tools developed by other partners including the DIG. This tool can find similar images

based on visual content over large image repositories containing hundreds of millions of images.

The Columbia Image and Face Search tool is able to return visually similar and near duplicate images and faces for any given query image. The tool uses deep learning based image features, namely a deep model train to recognize the Sentibank concepts [12] which are Adjective-Noun Pairs (ANPs) with strong correlation to visual sentiment and emotions. Thanks to these semantic features, the content-based retrieval method is able to match human subjects based on hairstyle, physical characteristic, dress, environment and pose. The face detection and recognition models are from the DLib[39] open source library. In order to handle the extremely large image database and support fast search response, it exploits a quantization approach [29] that enables an approximate search of the database in less than a second.

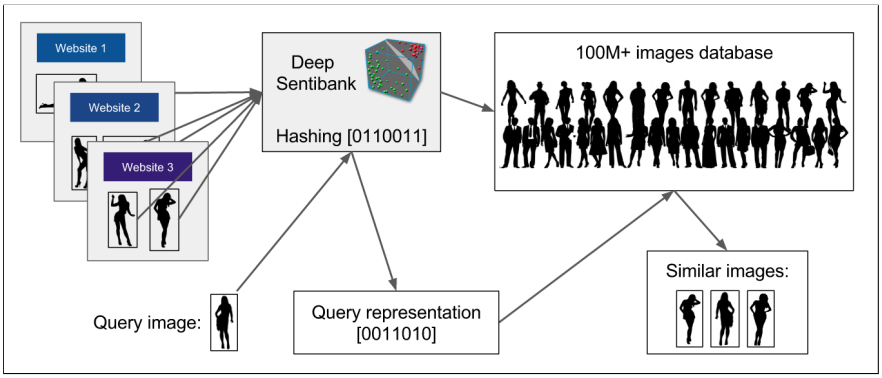

Figure 16: Overview of the Columbia Image Similarity Search tool.

The image similarity service (Figure 16) deployed in MEMEX currently indexes around 100 million images and 35 million faces of the human trafficking domain. It supports incremental update and the indexed images are up-to-date within a few minutes. Querying a previously unseen image takes about 2 seconds, of which about one second is for computing the image or face feature and the rest for the search process.

The image similarity search (Figure 17) tool has been open sourced[40] and was an important component of both the DIG Search Engine and the Tellfinder application from Uncharted during the MEMEX project. The image and face search tool is part of the MEMEX transition effort, and continues to be used daily by law enforcement agents across the country to help the fight against human trafficking online.

3.5 User Interface

Search in DIG is supported in a variety of ways. Users start by filling out some fields (or just free text) on a search form. As DIG retrieves results, facets on the side get populated so that users get a sense of how the data is distributed (e.g., if results indicate that many hits seem to be spatially localized). Users can browse any document in the ranked list of documents, obtain provenance for the extractions, open the cached webpage associated with each document, and click on the document to see temporal and geospatial information. Users can continue exploring by directly

[39] https://github.com/davisking/dlib,http://dlib.net/

[40] https://github.com/ColumbiaDVMM/ColumbiaImageSearch

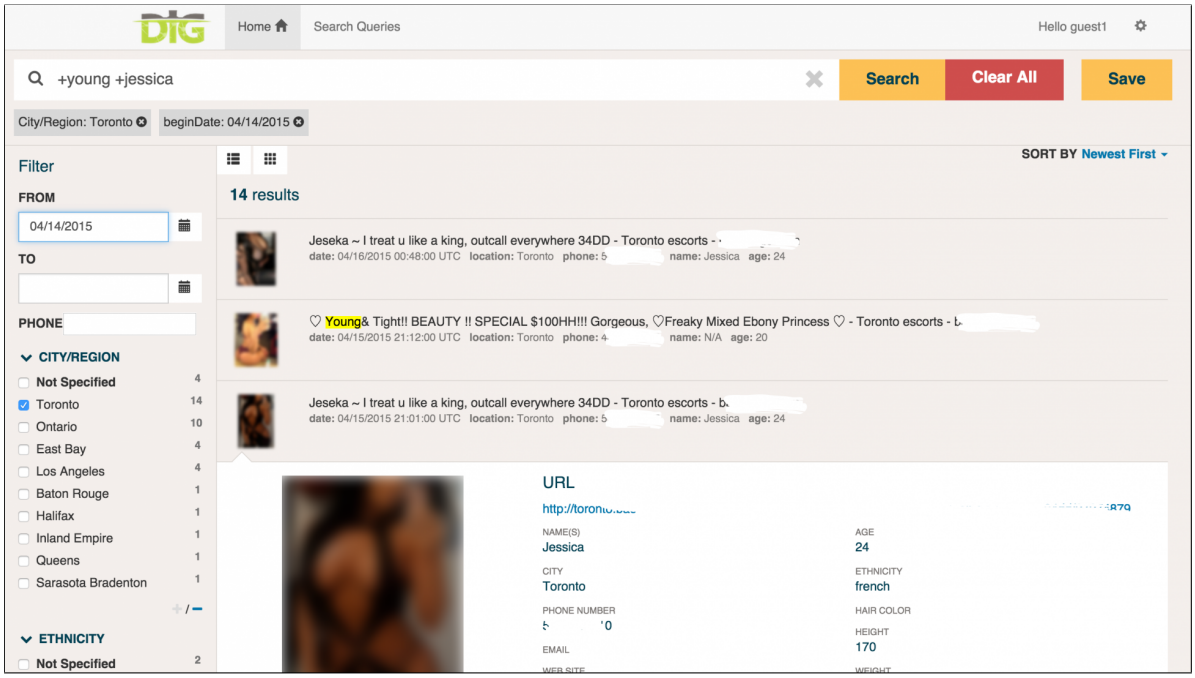

Figure 17: DIG search engine.

manipulating facets (by clicking on the tick boxes next to a facet item) and thereby making the search narrower or updating the original search form. We received direct feedback from users that having flexible exploratory capabilities did much to foster trust in DIG, and the underlying AI components. DIG does not require installation as it is hosted in the cloud and is accessible through a browser.

DIG supports both faceted and entity-centric search, as Figures 18 and 19 illustrate. For example, DIG can be used to jump-start an investigation starting from a vague, under-specified search query. For example, a user searches for a hispanic escort in Chicago who is known to provide certain sex services in her ads. By exploring a ranked list of ads retrieved by the system, along with images, the user is able to drill down, in an entity-centric fashion, on important details like phone numbers and emails that are promising avenues for field investigations. Acquiring such a list without the help of the system would ordinarily have taken months of field level investigations and online searches.

4 RESULTS AND DISCUSSION

4.1 Experience Building Applications

As research in machine learning and AI continues to progress, front-facing systems have become steadily more complicated, expensive and limited to specific applications like business intelligence (BI) [20], [52]. *Democratizing* technology is a complex issue that involves multiple stakeholders with different sets of abilities [67], [66]. A particular user whose needs are very real, but who is seldom addressed except in BI or military-specific situational awareness situations [38], is a domain expert with extremely limited technical abilities. In particular, such users do not know how

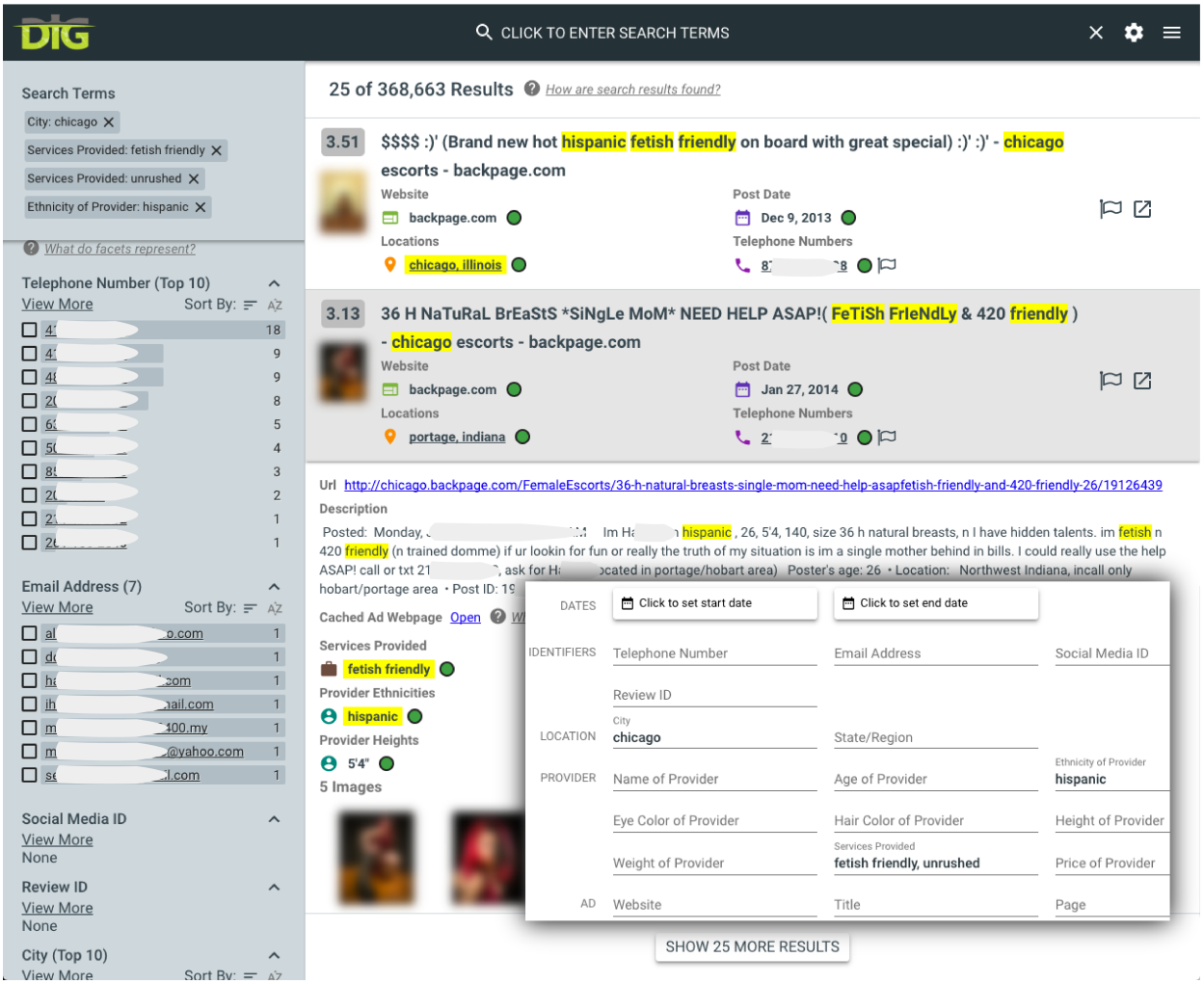

Figure 18: A case-study illustrating complex investigative search in DIG. Starting from vague details, a user is able to retrieve a set of highly specific ads that help her to narrow down on suspects.

to program, let alone cope with complex machine learning or deep learning algorithms, and cannot satisfy their information needs through simple Google search for several subsequently described reasons.

A real-world example of such a domain expert that we encountered in the securities fraud domain is an employee from the Securities and Exchange Commission (SEC) who is attempting to identify *actionable* cases of *penny stock* fraud [25]. Penny stock offerings in over-the-counter (OTC) markets are frequently suspected of being fraudulent, but without specific evidence, usually in the form of a false factual claim (that is admissible as evidence), their trading cannot be halted. With thousands of penny stock offerings, investigators do not have the resources or time to investigate all of them. One (technical) way to address this problem is to first crawl a corpus of relevant pages from the Web describing the domain, a process alternately known in the Information Retrieval (IR) literature as *relevance modeling* or *domain discovery* [39]. The latter term is more encompassing, as it involves not just relevance modeling but the actual crawling of the data.

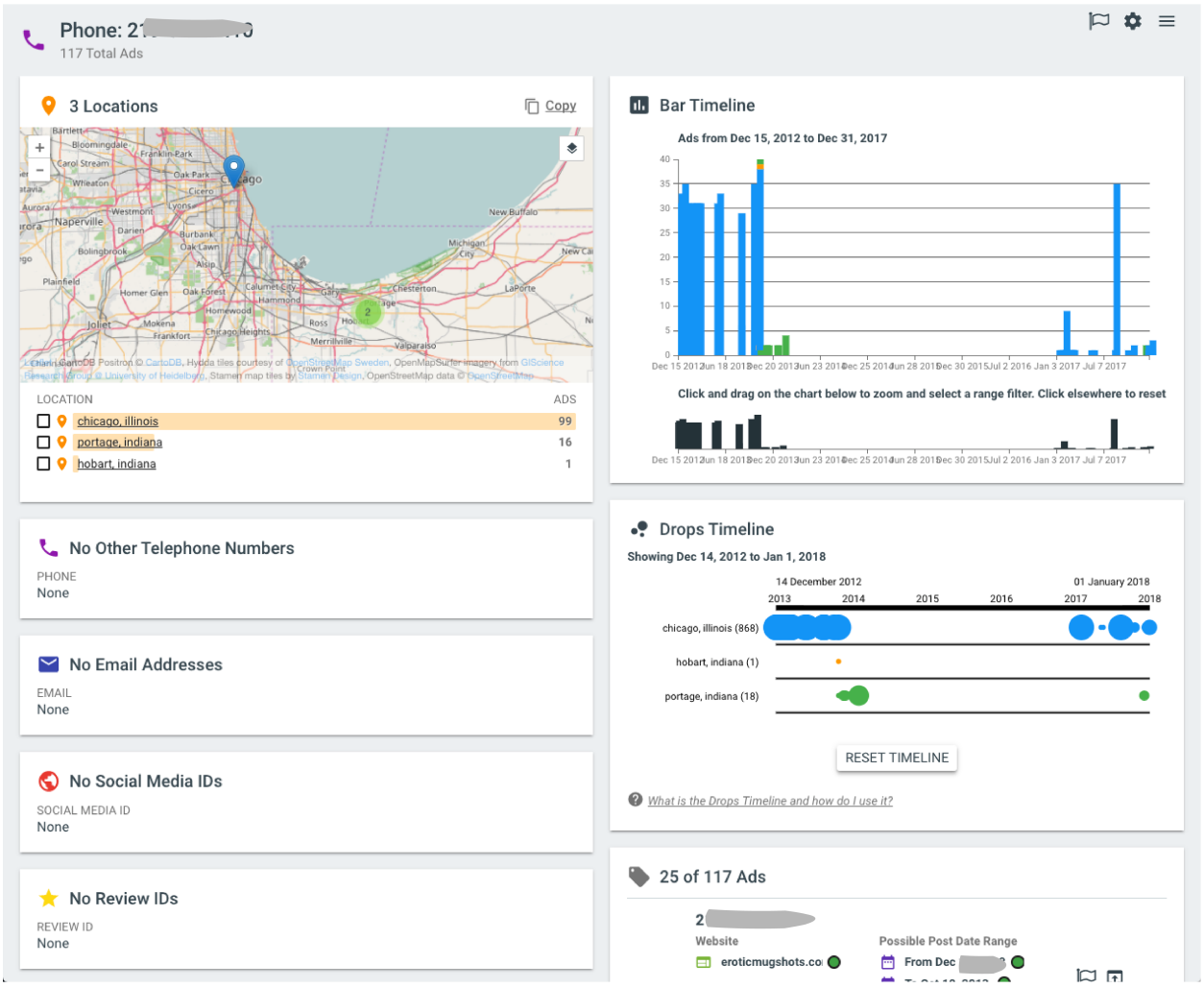

Figure 19: Continuing from Figure 18, the user is able to use entity-centric search facilities in DIG to collect a list of 117 ads, with details such as timelines and locations. These numbers can be used to conduct an investigation on the ground.

Once such a corpus is obtained, an expert in information extraction and machine learning would elicit opinions from the users on what fields (e.g., location, company, stock ticker symbol) are important to the user for answering domain-specific questions, along with example extractions per field. This sequence of *knowledge graph construction* (KGC) steps results in a graph-theoretic representation of the data (a *knowledge graph* or KG) where nodes are entities and field values, and the (directed, labeled) edges are relationships between entities, or assignments of field values to entities [57], [19]. Since the KG is structured, it is amenable to aggregations, and to both keyword and structured querying. With a good interface, for example, the domain expert can identify all persons and organizations (usually shell companies) associated with a stock ticker symbol, aggregate prices, or zero in on suspicious activity by searching for hyped-up phrases that indicate fraud.

In this section, we present our experience using DIG with domain experts in five different investigative domains, each of which involves significant technological potential. We describe the

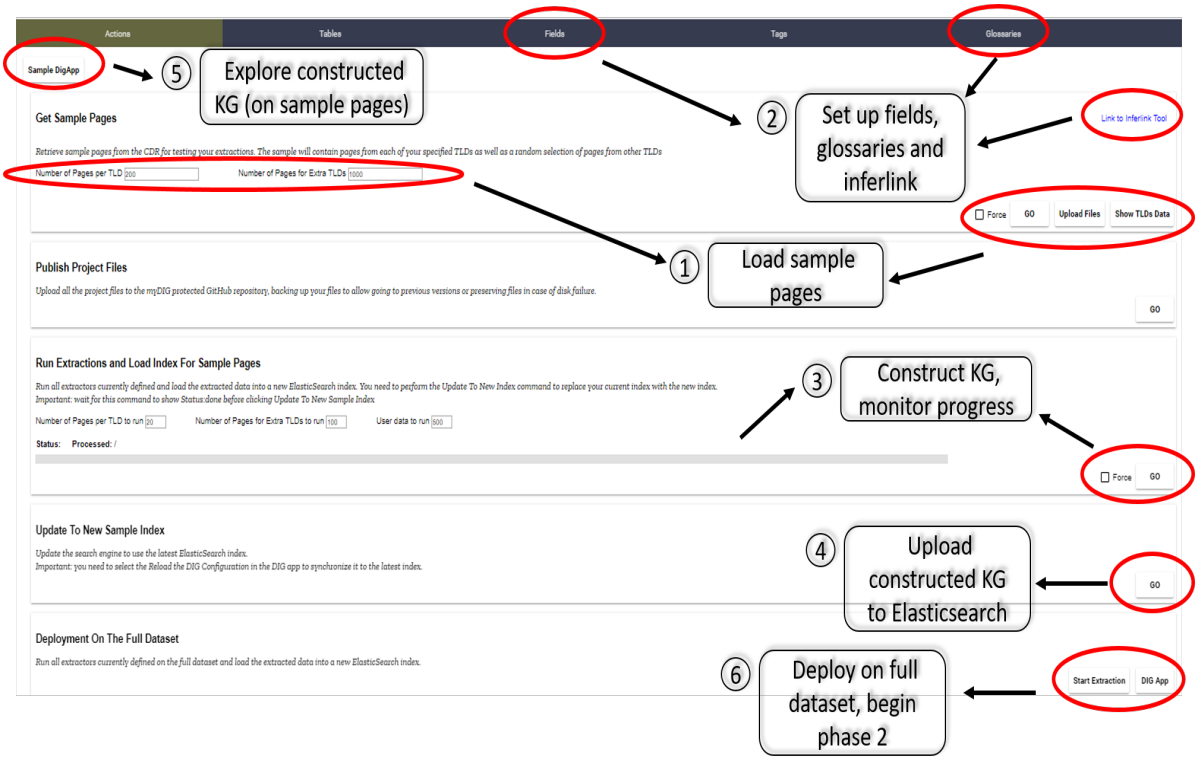

Figure 20: The DIG dashboard, with corresponding elements and actions.

dataflow and DIG components that allow users to construct and search knowledge graphs, and thereby discover new and relevant knowledge by conducting data-driven analysis. We also present qualitative and quantitative data collected from the five case studies to illustrate the real-world potential of DIG in making an advanced set of knowledge discovery technologies accessible to non-technical users. Using DIG, practicing, non-technical domain experts from the five case study domains were able to build a personalized domain-specific search engine over corpora containing more than a million raw webpages in 4-6 working hours.

4.1.1. System Overview and User Experience

The DIG action dashboard is illustrated in Figure 20. The input to the system is a corpus of webpages that is assumed to have been crawled by a domain discovery system prior to the user engaging with DIG. While domain discovery is, by no means, a solved research problem, significant advances have been made in recent years (including *deep crawls* of specific *top level domains* or TLDs like `backpage.com`) [14], [44], [45]. We make the loose assumption that a significant fraction of this corpus is relevant, but that there may be many irrelevant pages also. The DIG architecture is designed to be reasonably robust to such irrelevance, as we explain.

User experience in DIG can be separated into two phases. The first phase, *domain setup* and *curation*, involves setting up the domain with the goal of answering a certain set of questions that the user is interested in. This phase does not comprise a strictly linear set of steps, but can involve several interleaved steps (Figure 20). At a high level, the user loads a sample of the corpus to explore, followed by defining wrappers using the Inferlink tool, customize domain-specific fields, and add

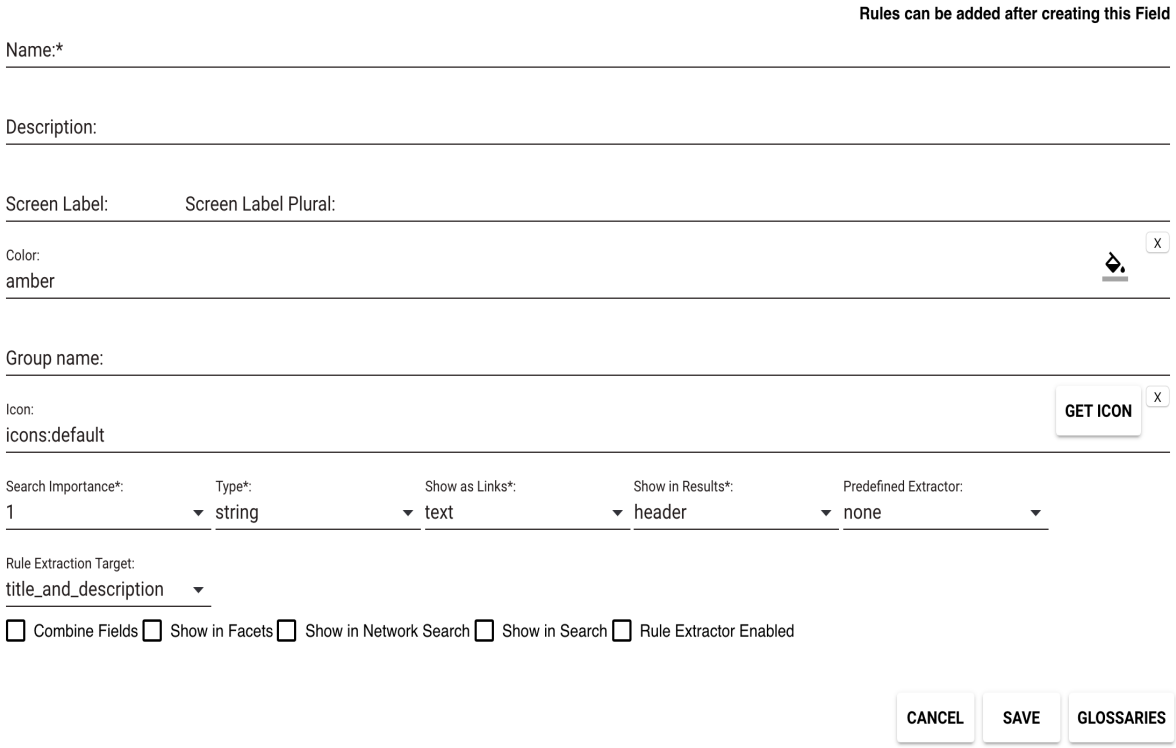

Figure 21: A form that allows the user to define/customize fields.

field-specific glossaries (if desired). Periodically, the user can crystallize a sequence of steps by running extractions (akin to constructing the knowledge graph) and uploading the knowledge graph to an index. Once the upload is complete, the user can 'demo' their effort by clicking on the *Sample DigApp* button in the upper left corner in Figure 20 to explore the knowledge graph using a search interface. The process is iterative: the user can always return to the dashboard to define or refine more fields, define more wrappers with Inferlink, or input more glossaries.

The second phase is the 'in-use' phase, when the system is actually being used to satisfy information retrieval needs. This phase tends to begin when the user is finished setting up the domain, and is ready to deploy all changes on the full dataset (the entire corpus). In subsequent sections, we describe the user experience and dataflow for both phases.

Setting up the Domain: The DIG system allows personalized setting up of a domain. Users can define and customize their own fields, choose what extractions to retain from webpages (using the Inferlink tool), add glossaries that might be available to them as investigators, and repeat the process till they are satisfied with domain setup.

The DIG system allows users to define their own fields, and to customize the fields in several ways that directly influence their use during the second phase (domain exploration). To define a field, users click on the *Fields* tab (Figure 20), which provides them with an overview of fields that are already defined (including pre-defined fields like location, detailed further below), and also allows them to add their own fields. A 'field form' is illustrated in Figure 21. In addition to customizing the appearance of a field by assigning it a color and icon, users can set the 'importance'

of the field for search (on a scale of 1-10), declare the field to represent an entity by selecting the 'entity' rather than the 'text' option in *Show as Links* (thereby supporting *entity-centric search*, described in *Exploring the Domain*), and assign it a pre-defined extractor like a glossary.

A powerful, interactive extractor that uses wrapper technology [37], [23] to extract *structured* elements from webpages is the Inferlink tool (described earlier). Inferlink operates in various steps. For convenient formalism, let us define a top level domain or TLD (e.g., `backpage.com`) T as a set $\{w_1, \ldots, w_n\}$of webpages (e.g., `backpage.com/chicago/1234`). As a first step, Inferlink uses an *unsupervised template clustering algorithm* to *partition* T into clusters, such that webpages in each cluster are structurally similar to each other. To provide some real-world intuition, one cluster could contain taxonomy webpages that contain lists of things, another cluster could contain webpages describing forum posts, while a third cluster could contain webpages containing long text. Inferlink presents the users with samples from these clusters, and allows users to select a 'relevant' cluster for curation. At this point, users see an illustration like the one in Figure 1 wherein Inferlink has extracted *common structural elements* from the webpages in the cluster and presents it to the users in a column layout, with one row per webpage and one column per structured element that is common to the webpages. Users can open a webpage by clicking on a link, delete a column, or assign it to a field that has already been defined, as in the figure. In the literature, this step is also called *semantic typing* of columns [59]. Users must separately type columns for each top level domain (TLD), since different Web domains share different structures in the webpages they contain. We also trained users to perform more advanced customizations with Inferlink in less than an hour of example-based demonstrations. For example, users could choose to curate and semantically type more than one cluster from a given TLD, if they felt it gave them added value over another cluster from a less important TLD.

In addition to the Inferlink took, DIG offers extraction methods suitable for information extraction from blocks of text or other content that is not delimited as structured HTML elements i.e. delimited using HTML tags. To ease user effort, some generic extractors are pre-trained and cannot be customized, but can be disabled. A good example is the location extractor, which extracts the names of cities, states and countries, using a machine learning model that was trained offline. However, DIG offers users the option to input a glossary (with incumbent options, such as whether the glossary terms should be interpreted case sensitively or not) for a given field. This option was popular with domain experts, as we describe in the *Qualitative Feedback* section. Although not evaluated herein, the latest version of DIG also offers users an intuitive rule editor for expressing and testing simple natural language rules or templates with extraction placeholders. For example, a name extractor can be set up with a pattern recognition rule like 'Hi, my name is [NAME]'.

4.1.2. Exploring the Domain

Once the domain has been set up, users begin the second phase, which involves exploring the domain to get the answers they need, or even to generate new leads and questions (an important concern in investigative domains where users are sometimes not sure quite what they are looking for). We describe the main components of a typical user experience.

Specifying Queries: Keywords and More: The DIG system supports both basic keyword search, as well as *structured search* wherein the user curating the domain can fill out values in a form containing fields that she has declared (during domain setup) as being *searchable* (Figure 22).

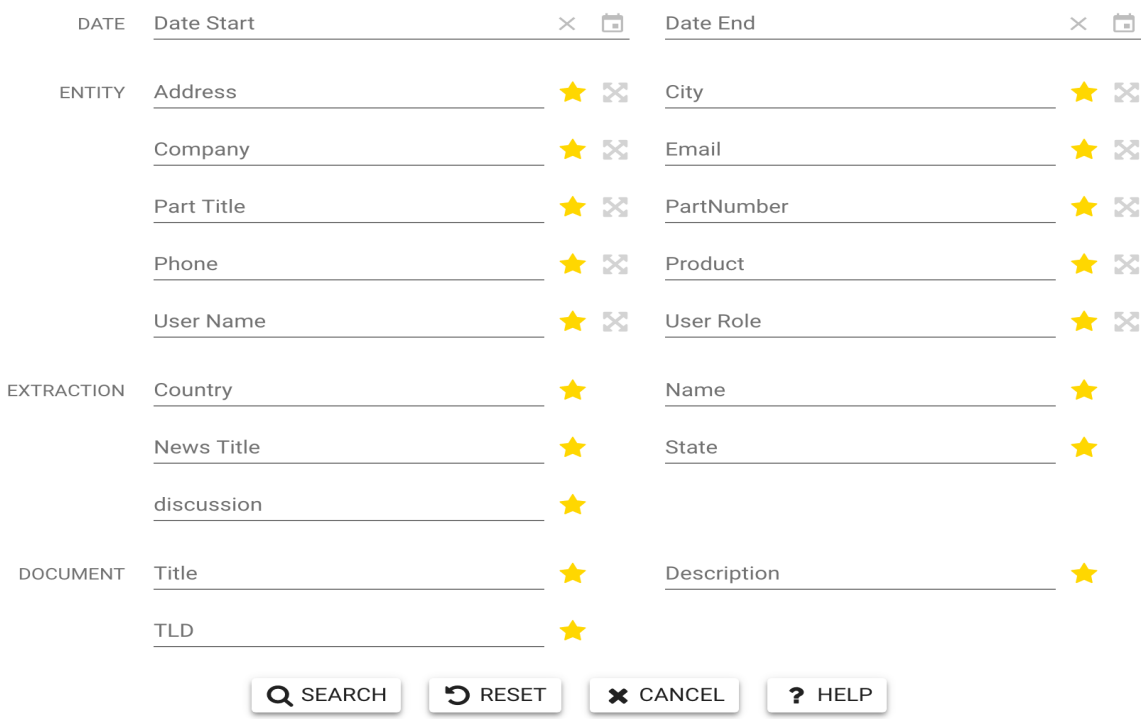

Figure 22: A form that allows the user to pose fine-grained, structured queries (*Counterfeit Electronics* domain).

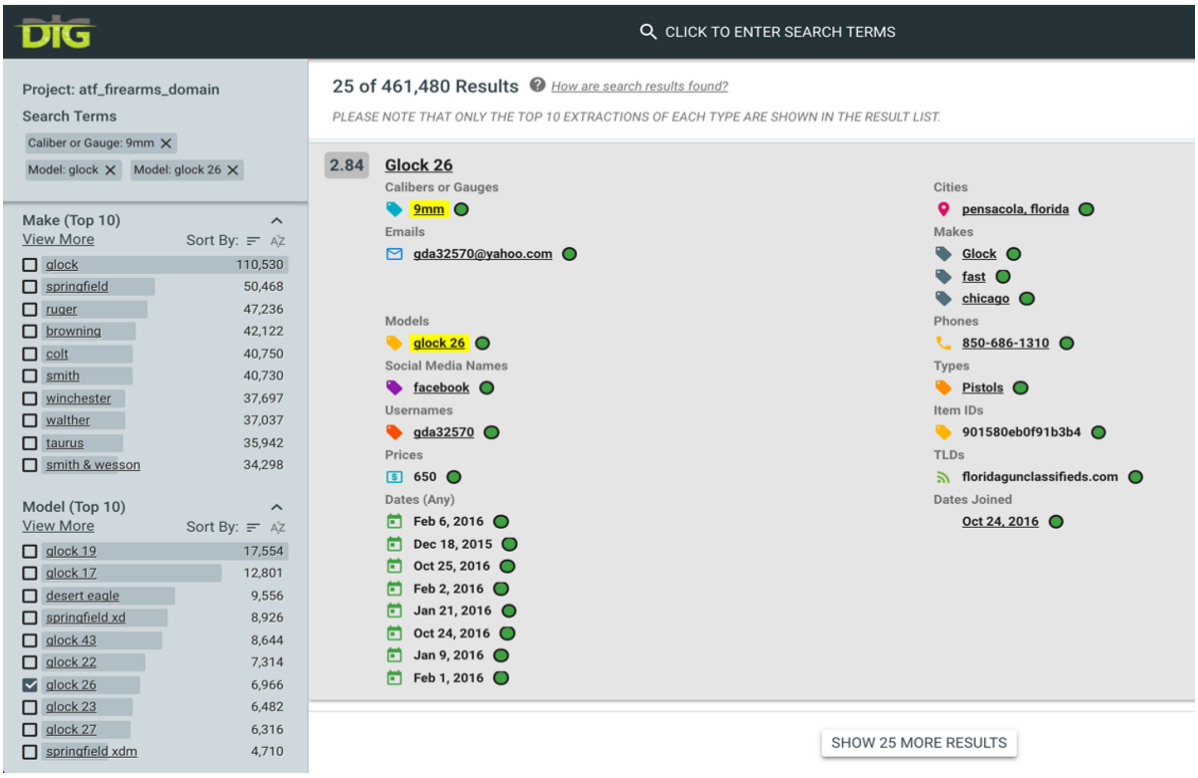

Figure 23: The search interface offered by DIG (*Illegal Firearms Sales* domain).

As described in *Algorithms and Technologies*, the search engine in DIG uses an advanced set of techniques from the IR literature to ensure that user intent is captured in a robust and high-recall manner. The DIG system also supports entity-centric search, which is described shortly. While keyword search is designed to be primarily exploratory, structured search allows a user to quickly hone in on pages containing certain key details that the user has specified in the form. Since DIG uses ranking and relevance scoring, satisfying more criteria on a form will lead to a page having higher ranking, compared to another page that satisfies fewer criteria.

Facets: As shown in Figure 23, DIG supports faceted search and filtering on select fields (e.g., Model or Make in the figure[41]) that the user can specify during domain setup. In all of our case studies, users made fairly intuitive choices: except for 'free-form' fields like text, descriptions or comments, they favored faceting over non-faceting. In addition to allowing more informed search and filtering, facets also help the user to see an overview of the search results. For example, in the figure, one can deduce that (among the *glock* models) models 19 and 17 occur far more often in the data than other models.

Entity-centric Search and Summarization: Entity-centric search (ECS) and summarization is an important argument that distinguishes the domain exploration facilities of DIG from more generic Google search. An example is illustrated in Figure 24 for the entity *glock 26*, which is presumably a firearm model that an investigator in the Illegal Firearms Sales (IFS) domain is interested in

[41]Taken from the *Illegal Firearms Sales* (IFS) domain.

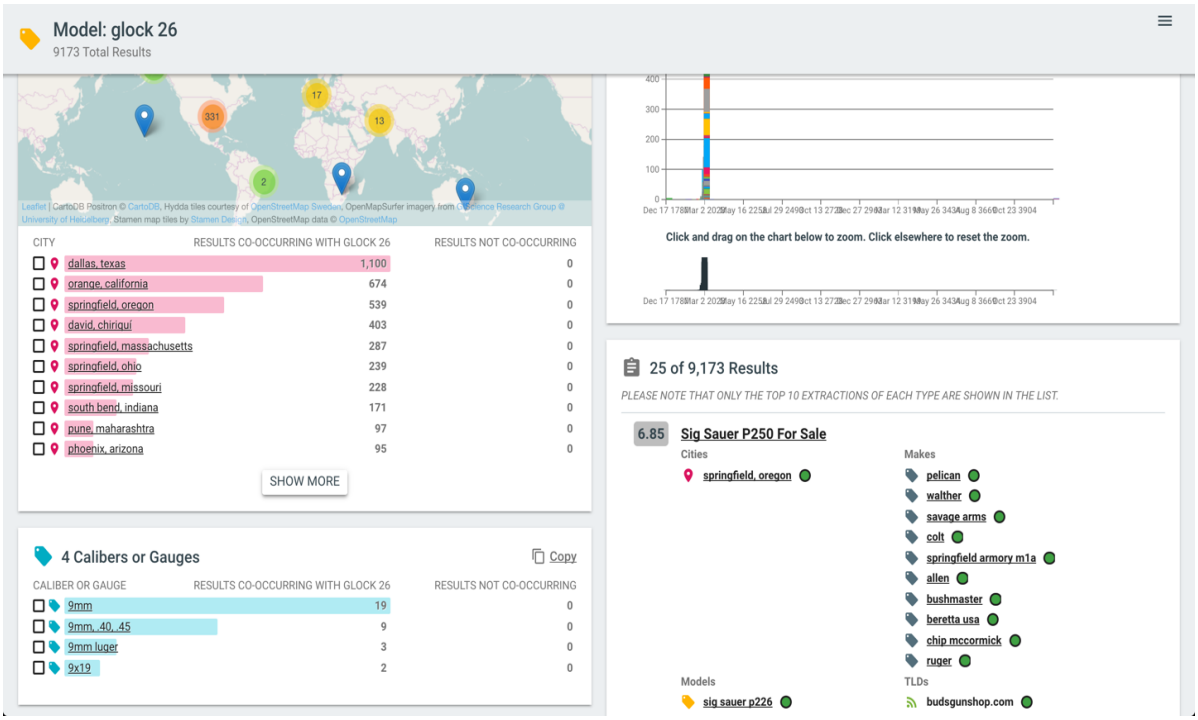

Figure 24: Entity-centric search (ECS) for the *Illegal Firearms Sales* domain in DIG.

further investigating for suspicious transaction-related activity. The ECS dashboard summarizes the information about this entity by providing a (1) timeline of occurrences, (2) locations extracted from webpages from which the entity was extracted, (3) other entities co-occurring with the entity (along with *non co-occurrence* information to enable intuitive significance comparisons), (4) relevant pages related to that entity. Users can declare the extractions of any field to be ECS-amenable by toggling the *Show as Links* option in the form in Figure 21.

Provenance: The user interface supports provenance both at the (coarse-grained) level of webpages and the (fine-grained) level of extractions. Concerning the latter, provenance information is obtained by clicking on the green circle next to an extraction (see Figure 23), which brings up the specific extraction method such as Inferlink, spaCy, glossary etc.(Figure 25), and in the case of context-based extractors that use methods like word embeddings, the *text* surrounding the extraction. Multiple provenances are illustrated, as shown in the figure, if applicable (e.g., glossary extractions from text that originated in different structures in a webpage). We also support webpage-level provenance by allowing the user to open the cached[42] webpage in a new tab.

4.1.3. Evaluations and Case Studies

DIG was evaluated by the DARPA MEMEX program in five different investigative domains, namely securities (specifically, penny stock) fraud, illegal firearms sales, illicit shipments via USPS mail,

[42]It is important to show the cached, rather than the 'live' webpage (which can also be shown by clicking on the pre-defined URL extraction that exists for every webpage in the corpus) since the webpage may have changed, or even removed, since domain discovery.

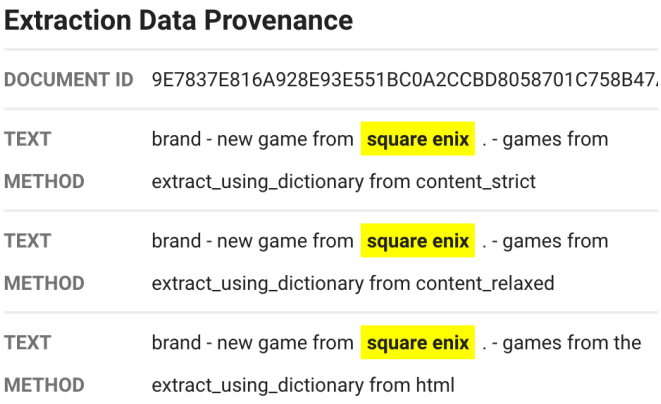

Figure 25: Provenance of glossary-based *organization name* extraction 'square enix' (*Securities Fraud* domain).

narcotics and counterfeit electronics. Each of these domains is extremely specialized, but has common characteristics that make it particularly amenable to analysis in DIG. When describing each domain, we focus on the types of investigative questions domain experts are looking to answer.

Concerning the evaluation protocol, to keep the user study as unbiased as possible, DARPA, NIST and a contracted private firm conducted all evaluations[43] described herein and released a select set of results both to inform future work, and to provide guidance on the current state and usability of the system. Based on a number of factors, the most important of which was the preference expressed by domain experts, not every result was released to our research group or can be published in the open research literature. By necessity, therefore, some evaluations are more detailed than others.

4.1.4. Users and Domains

The DIG system was evaluated by five pairs of users, each of which is affiliated with either a federal agency or a national organization. The users are practicing experts in their respective domains. Because they involve highly illicit activities, the domains described below may also have a significant Dark Web presence, but in this paper, all data discussed, released, analyzed or presented were crawled over the Open Web.

Concerning protocol, each pair of domain experts arrived on a separate day in the Washington D.C. facility reserved by DARPA for these evaluations. A member from our research team trained the users for 1-2 hours in navigating the system, setting up the domain and conducting search as we have described it in this paper. Users were then left to set up their domains (Phase 1, as described in *System Overview and User Experience*) in DIG. On a separate day (usually the next, but always within the week), users conducted actual knowledge discovery and search (Phase 2) by using the search interface for 2 hours. All feedback was collected by NIST and DARPA after Phase 2 had concluded.

[43]The one exception is a posthoc evaluation of *extraction quality* that was conducted by researchers in our group.

Securities Fraud (SF): Securities, particularly *penny stock*, fraud is a complex domain that falls under the direct authority of the Securities and Exchange Commission (SEC) in the United States. Penny stock fraud is unusual because much of the activity that *accompanies* fraudulent behavior, including hype and promotional activity, is legally permitted. Many of the actual actors involved may not be physically present in the US, but for regulatory reasons, 'shell' companies fronting such activity for promotional and legal purposes, have to be registered in the US to trade stocks legitimately in over-the-counter (OTC) exchanges. In addition to the longer term goal of investigating, and gathering information on, such shell companies and the people involved in them, investigators are also interested in taking preventive activity. This can happen when a penny stock company is caught actively engaging in *factually fraudulent* hype (for example, a false claim that a contract was just signed with a well-known customer firm), in which case trading can be halted or even shut down[44]. The DIG system supports these goals by allowing users to aggregate information (in the crawled corpus, which contains many Web domains) about suspicious penny stocks using the ECS facilities, and also to zero in on bourgeoning promotional activity.

Illicit Shipments via USPS Mail (USPS): Unlike the securities fraud domain, illicit shipments via USPS happen in the physical world but *communications* about the illicit shipment, particularly tracking numbers, happen *digitally* in specific forums and typologies that investigators in this domain understand well. Like the *Narcotics* domain, and unlike the other three domains, the USPS domain is an 'under the cloak' domain about which investigators have not shared much information. After training, however, these investigators were able to set up the domain and use it on their own despite having no technical or programming abilities. While the actual search methodology, and the questions that investigators were seeking to answer, are strictly confidential and cannot be revealed, we were allowed to assess (and report on) user effort in setting up the domain, and subsequent KG quality.

Illegal Firearms Sales (IFS): In the US, firearms sales are regulated in the sense that transactions cannot be conducted with arbitrary persons, or over arbitrary channels like the Internet. Investigators in the IFS domain are interested in pinpointing activity that, either directly or indirectly, provides evidence for illicit sales that leave some digital trace. The domain is similar to the SF domain (and dissimilar to the CE domain, described below) for the important reason that investigators limit their focus to domestic activities.

Counterfeit Electronics (CE): Despite what the name suggests, investigators in the counterfeit electronics domain are interested, not in consumer electronics, but in microchips and FPGAs that form the computational backbones of more complex 'application' devices. The FPGAs may resemble an FPGA from a genuine contractor, but are fakes, and may have malicious modifications at the hardware level. Certain countries, companies and devices are more relevant to this kind of activity than others. We note that there is an obvious national security component to these investigations, and just like with the other described domains, domain expertise plays a crucial role both in setting up the domain, and in the knowledge discovery itself.

[44]This is why the step is *preventive*; trading is shut down before unwary investors 'buy in' and subsequently end up losing their savings.

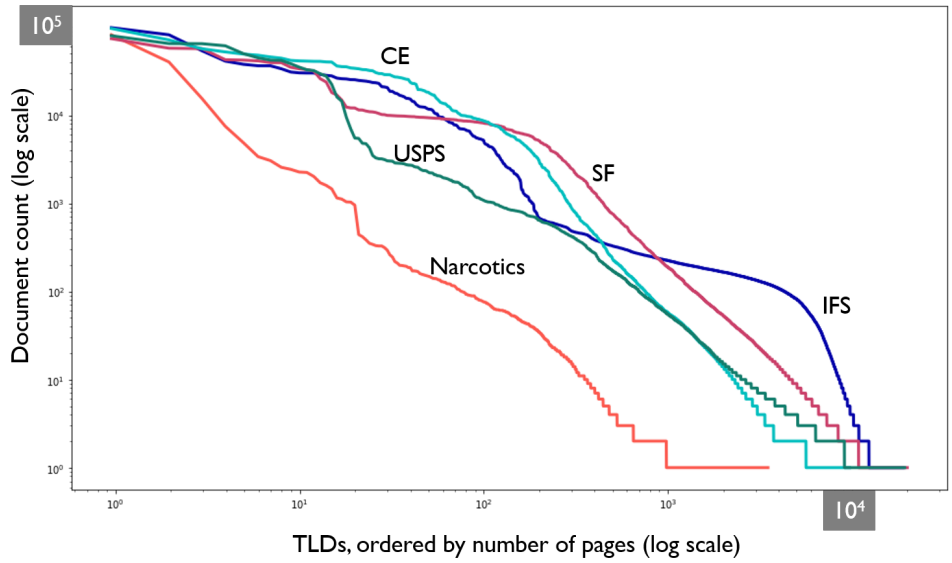

Figure 26: Document counts per TLD (Y-axis) against ranked list of TLDs (ordered by document count on X-axis).

Narcotics (N): Due to confidentiality constraints, there is little that we can reveal about the narcotics domain, except the self-explanatory implication in the name itself. Like the other domains, it is important to note the caveat that investigators are specifically interested in activity that has some digital component that can guide further investigation. That is, it does not purport to cover all narcotics activity. Among all five domains, this domain is featured most prominently in Dark Web data.

4.1.5. Domain Discovery Corpora

Figure 26 characterizes the corpus for each domain. From the roughly *power-law* distributions seen for most of the domains (near-linearity on log-log scale), despite their diversity, we note that some TLDs in each distribution are much more heavily represented than others. In turn, this immediately suggests why a tool such as Inferlink (that requires users to curate on a per-TLD basis) could be useful. By focusing on the *largest TLDs*, users can end up curating a *large fraction* of webpages. On the other hand, there is a significant *long tail*, which means that Inferlink (which is best suited for the short tail or largest TLDs) cannot be the only solution for recall-friendly knowledge discovery. Glossaries, as well as pre-defined extractions, are required to take up the slack. There is a clear tradeoff between user effort and knowledge graph quality. In the evaluations described subsequently, we measure both.

4.1.6. Evaluation of User Effort

Tasks and Materials: We evaluate user effort in setting up the domain in DIG in two different ways. First, we evaluate the level of effort that was expended in each domain on using the *Inferlink* tool to curate one or more relevant (as judged by the users) clusters of webpages. Since Inferlink operates on a per-TLD (more specifically, a per-*cluster per TLD*), we refer to its extractions as *short*

Table 19: Quantifying user effort in terms of total numbers of fields and glossaries per domain, after the users had finished setting up their respective domains.

Domain	Number of Fields	Number of Glossaries	Examples (user-defined fields)
IFS	27	8	caliber_gauge, ffl_status
CE	21	6	PartNumber, Company
USPS	21	5	confirmations, Tracking_Number
N	43	5	used_bitcoin, vendor
SF	32	6	counsel, disclaimer

tail extractions. We plot the short tail against the long tail, by reporting the numbers of webpages for both cases per field such that the webpage had at least one extraction (from Inferlink for the short tail, and overall i.e. any extraction at all, for the long tail) for that field. Second, we evaluate the level of user effort in defining the domain by reporting the total number of fields for each domain, with some representative examples of fields that users defined from scratch. We also quantify the number of glossaries used per domain, since qualitative feedback by users indicated that glossaries were extremely popular in incentivizing users to set up custom fields.

Results: Figure 27 illustrates the distribution of the numbers of webpages with at least one extraction (per field, as noted on the y-axis) both for the short and long tails. The figures illustrate two interesting aspects quantifying the importance of expending effort in the Inferlink tool vs. more all-encompassing methods like glossaries. In the IFS and SF domains, we find that Inferlink has made significant fractional contributions to some fields (at the bottom of the y-axis) compared to domains like Narcotics, where the long tail overwhelms Inferlink. On the other hand, we find that, for some fields, only Inferlink has yielded any extractions at all. For example, in the CE domain, fields like *UserRole* and *Product* had non-zero extraction numbers only because of Inferlink. In posthoc qualitative feedback, users expressed that using Inferlink gave them a means of expending effort with the confidence that the effort yields usable, interpretable results.

Results in Table 1 show that between 20-40 fields were retained (from the generic set of pre-defined fields, like city or state) or defined by users. Some domains are much finer-grained than others, an example being N. Users also made good use of the glossary facility, with the number of glossaries (8) being the highest for the IFS domain.

4.1.7. Evaluation of Knowledge Graph (KG) Quality

Tasks and Materials: The aim of this evaluation was to assess the quality of the KG; specifically, the precision of the extractions for fields in the KG. We conduct this task by sampling 25 random webpages per domain, and assigning each domain to an annotator from our research group. Each extracted element (for each field) reported by DIG was given a 0/1 score for correctness. Each annotator made this assessment by referring back to the original cached webpage to verify whether the answer was correct. We report these precision metrics, along with the total numbers of

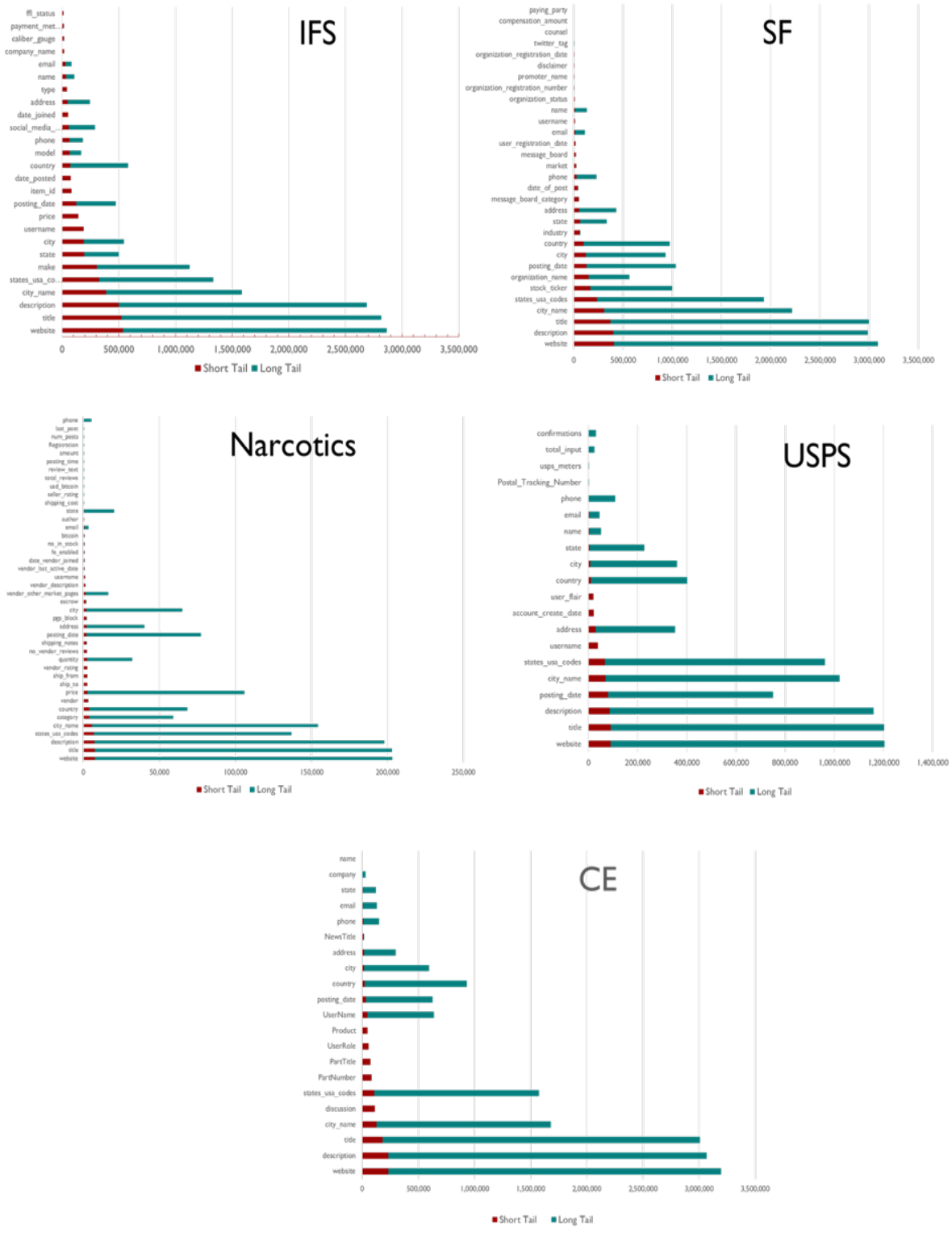

Figure 27: For all domains, a comparison of webpage count (x-axis) such that at least one value per field (y-axis) was extracted from that website (green bar/long tail), against the short tail/red bar wherein only Inferlink extractions are considered.

Table 20: Manually judged precision of field extractions stored in the knowledge graph for each domain.

Domain	Number of Extractions	Correct	Precision
IFS	168	125	74%
CE	85	75	89%
USPS	116	109	94%
N	211	156	74%
SF	67	48	72%

extractions evaluated for correctness, and comment on the results.

Results: Results in Table 20 show that DIG is able to achieve reasonable levels of precision for the collective sets of extractions. For some domains, like CE and USPS, precision is close to (or above) 90%, while for others it tends to be between 70-80%. While there is room for improvement, these precision numbers show that, on average, there is considerably more signal than noise in facets and extractions.

4.1.8. Qualitative Feedback and Analysis

Much feedback was elicited during the testing of DIG from users of all five domains. In the interests of space, and the contributions that we claimed in the introductory section of this paper, we focus on how DIG helped users obtain insights that are otherwise not achievable using generic search technology like Google.

Insights beyond Google: In response to the question *Did the tool ever help you achieve an insight that would have been otherwise difficult to identify without the tool? If so how?* posed by NIST evaluators, users had the following comments[45]:

[IFS] Yes, it allowed us to see possible connections to phone numbers, email addresses, locations, etc. that could not necessarily be verified/discovered using Google or site-led searches within Armslist, backpage, etc.

[CE] SAVES SO MUCH TIME. Aggregates data that would be searched using google.

Comments from SF users revealed, however, that the actual corpus crawled can sometimes be an important bottleneck:

[SF] Conceptually, maybe it would, but for the use cases proposed by SEC, there were no insights found; data extraction/crawling issues

Productivity: Users also felt that DIG improved their productivity and helped them to focus on relevant information. In response to the question, *Did the tool identify any new and relevant sources of information that you had not found through other methods previously? If so what were they?* users had the following comments:

[45]Comments are not syntactically modified in any way

[IFS] It was able to categorize postings by location which allowed us to focus our investigation.

[CE User 1] -Spartan6 xlinx (email addresses); -domain search and aggregation on email addresses; -filling in relevant domain endings to match[46]

However, once again, the crawled data (provided to DIG by the domain discovery teams) was noted to be an issue by both SF and CE User 2.

[SF] No ; index properly ; Investorshub was not indexed correctly – Some posts are crawled and scraped correctly, most aren't

[CE User 2] Was really unhappy with data that was pulled in by the tool. A better (more efficient) domain discovery process...

The indexing problems with Investorhub were due to DIG improperly parsing web pages containing posts. The pages in this web site contain lists with nested elements that the DIG tool could not interpret correctly. Despite the comments by SF users, we note that they explicitly indicated to us that they would like the system to be transitioned to their office, and that they would like to keep using DIG to curate their domain and applying it in actual on-the-ground investigations.

Useful Features: Users were also asked the following: *Were there any features within the tool that stood out as particularly useful?* Most users expressed that the glossary feature was very useful, and ECS was also singled out by some; specific responses are noted below:

[CE] -Really liked the glossary feature; -Can have multiple domains; -Inferlink integration is Really straightforward; -Status bar was helpful to know when a process is running vs. when it is not; -subset of TLDs helpful

[SF] -Being able to 'click' on the entity; -documents that relate to an address; find relevant information associated with a particular entity

4.1.9. User Study

Given the unusual nature of investigative search in illicit domains, we believe the evaluation and user study protocol decided upon by DARPA and NIST played an important role. A total of 16 questions (6 lead generation, 6 lead investigation and 4 operationally relevant i.e. generated in conjunction with the office of a state District Attorney based on its actual investigative needs during that time) were used to evaluate the assistive potential of DIG. A second system, TellFinder, which has a similar philosophy to DIG and has also been independently funded and developed under MEMEX, was also evaluated. These questions (with annotated answers) were independently derived by subject matter experts (SMEs) from a private research organization that had no part in building DIG, and that has been actively involved in training investigators in using search systems developed under MEMEX.

At the time of evaluation, DIG had indexed more than one hundred million sex ads collected over a period of more than two years through focused crawling systems also developed under MEMEX (by different teams). The purpose of the evaluation was not to collect a detailed set of quantitative results that could be put through statistical significance testing, but to understand the nature of investigative search itself, and the role that assistive systems like DIG and TellFinder could play, both through their exploratory UX facilities as well as the advanced AIs that are used to populate the knowledge base at scale. Fourteen users from multiple state and federal agencies

[46] While hard to interpret posthoc, we believe these indicate relevant information sets discovered by User 1.

volunteered their time to participate in the study. The user study itself was delegated to NIST by DARPA, with the stated evaluation goals being usefulness, usability and accuracy (compared to the SME-annotated answers), along with subsequent data collection and analysis.

The developers of DIG and TellFinder each had about an hour to brief the domain experts in small batches, illustrate key uses of the tool, and answer any questions they had. Following this training phase, each user was given the option of working either remotely or onsite. A facilitator from NIST took notes in the background without assisting or interfering in any way. A user was given a limit of 30 minutes to generate answers for each of the 4 operationally relevant questions, and 15 minutes for each other question. Except the facilitator, no one was present in the room during tool use. The order of assigned questions per user was random, and each question was exposed in turn i.e. right before the search session for that question commenced. Posthoc, the accuracy of users' answers was scored on a scale of 0-3, with the scale described in the sidebar. We do not comment on other metrics in this case study, but the operational impact of DIG and TellFinder (subsequently described) indicates significant uptake of both systems in the real world.

Feedback from investigative users shows that the system has potential for tackling some difficult questions, with the quality of domain discovery and data acquisition proving to be an important bottleneck that we are looking to address in future work. Most encouragingly, users (particularly in the Securities Fraud domain) have explicitly indicated to us that they would like to keep using the system, and would like the system to be transitioned to them.

The operational impact of both DIG and TellFinder has been widespread. Both are currently being used, often in complementary ways, by more than 200 law enforcement agencies in the US, specifically to combat sex trafficking. Outputs (particularly cached webpages, later taken offline) from these tools have led to evidence formally presented in court to arraign a sex trafficker.

5 CONCLUSIONS

There is a need to democratize and personalize machine learning and search technologies as they become ever more complicated, such that users and analysts in specific domains can interact with these tools without sacrificing the benefits of years of domain expertise. The DIG project was an effort funded under the DARPA MEMEX program to facilitate this high-level goal. User experience while using DIG can be roughly separated into two phases: setting up the domain, and conducting search. The DIG system does not require specifying vocabularies or making ontological commitments in advance, and is not specialized for a single domain. More specifically, in all five investigative domains where DIG was evaluated, users were able to build a personalized, domain-specific search engine over corpora that contained hundred of thousands, and in some cases, millions, of raw HTML webpages.

The software created in the DIG project is released publicly on Github (`https://github.com/usc-isi-i2/dig-etl-engine`) using the MIT license. The repository provides the source code , installation instructions, a user guide and sample projects.

In addition, standalone components that can be used separately are also released publicly on Github:

- `https://github.com/usc-isi-i2/etk`: information extraction
- `https://github.com/usc-isi-i2/rltk`: reccord linkage

- https://github.com/usc-isi-i2/dig-ui: user interface
- https://github.com/usc-isi-i2/dig-sandpaper: search engine
- https://github.com/inferlink/landmark-extractor: inferlink extractor
- https://github.com/ColumbiaDVMM/ColumbiaImageSearch: image and face search

Distribution Statement Further distribution only as directed by AFRL/RIEA, Rome NY 13441

Acknowledgements All data used in this research was provided by DARPA. This data is not delivered as part of this contract.

The USC Information Sciences Institute thanks its subcontractors, Inferlink[47], Next Century Corporation[48] and the Columbia University Digital Video and Multimedia Lab[49] for their contributions to DIG.

The Principal Investigator especially thanks his colleagues Mayank Kejriwal and Amandeep Singh for their intellectual and programming contributions to the project, and DARPA for supporting the project.

In addition, many members of the Center on Knowledge Graphs at USC ISI[50] contributed to the project, including researchers Craig Knoblock, Anoop Kumar and Linhong Zhu; research programmers Yixiang Yao, Dipsy Kapoor, Craig Milo Rogers, Runqi Shao, Dongyu Li and Andrew G. Philpot; Ph.D. students Majid Ghasemi-Gol and Jason Slepicka; and Master students Suresh Alse, Rajagopal Bojanapalli, Vinay Dandin, Akshay Ramesh Dani, Jiayuan Ding, Lidia Ferreira, Muthu Rajendran R. Gurumoorthy, Anika Jain, Patricia JimÃĺnez, Rahul Kapoor, Qian Liang, Qingyuandi Lin, Yike Liu, Sanmukh Lodha, Pulkit Manocha, Shrikanth Narayanan, Ankita Nargundkar, Subessware S K, Dhvanan Shah, Kaushal Shah, Preetam Shingavi, Sanjay Singh, Ashish Bharadwaj Srinivasa, Lingzhe Teng, Shreya Venkatesh and Chengye Yin.

References

[1] Inferlink r&d capabilities. http://www.inferlink.com/our-work#research-capabilities-section. Accessed: 2017-04-28.

[2] Isi extraction toolkit repository. https://github.com/usc-isi-i2/etk. Accessed: 2017-04-29.

[3] Readability text extractor. https://www.readability.com/. Accessed: 2017-04-28.

[4] spacy natural language package. https://spacy.io/, 2017. Accessed: 2017-09-19.

[5] Alvari, H., Shakarian, P., and Snyder, J. K. A non-parametric learning approach to identify online human trafficking. In *Intelligence and Security Informatics (ISI), 2016 IEEE Conference on* (2016), IEEE, pp. 133–138.

[47] http://www.inferlink.com
[48] https://nextcentury.com
[49] https://www.ee.columbia.edu/~sfchang/
[50] http://usc-isi-i2.github.io/home/

[6] Amanatullah, B. Landmark extractor. https://github.com/inferlink/extraction, 2016.

[7] Bauer, F., and Kaltenböck, M. Linked open data: The essentials. *Edition mono/-monochrom, Vienna* (2011).

[8] Bird, S. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions* (2006), Association for Computational Linguistics, pp. 69–72.

[9] Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., and Hellmann, S. Dbpedia-a crystallization point for the web of data. *Web Semantics: science, services and agents on the world wide web 7*, 3 (2009), 154–165.

[10] Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data* (2008), AcM, pp. 1247–1250.

[11] Bollegala, D., Maehara, T., and Kawarabayashi, K.-i. Embedding semantic relations into word representations. *arXiv preprint arXiv:1505.00161* (2015).

[12] Borth, D., Ji, R., Chen, T., Breuel, T., and Chang, S.-F. Large-scale visual sentiment ontology and detectors using adjective noun pairs. In *Proceedings of the 21st ACM International Conference on Multimedia* (New York, NY, USA, 2013), MM '13, ACM, pp. 223–232.

[13] BV, E. Elasticsearch. https://github.com/elastic/elasticsearch, 2016.

[14] Chakrabarti, S. *Mining the Web: Discovering knowledge from hypertext data*. Elsevier, 2002.

[15] Chang, C.-H., Kayed, M., Girgis, M. R., and Shaalan, K. F. A survey of web information extraction systems. *IEEE transactions on knowledge and data engineering 18*, 10 (2006), 1411–1428.

[16] Dai, A. M., Olah, C., and Le, Q. V. Document embedding with paragraph vectors. *arXiv preprint arXiv:1507.07998* (2015).

[17] Dalvi, B. B., Cohen, W. W., and Callan, J. Websets: Extracting sets of entities from the web using unsupervised information extraction. In *Proceedings of the fifth ACM international conference on Web search and data mining* (2012), ACM, pp. 243–252.

[18] Doan, A., Ramakrishnan, R., and Vaithyanathan, S. Managing information extraction: state of the art and research directions. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data* (2006), ACM, pp. 799–800.

[19] Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., and Zhang, W. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014), ACM, pp. 601–610.

[20] Elbashir, M. Z., Collier, P. A., and Davern, M. J. Measuring the effects of business intelligence systems: The relationship between business process and organizational performance. *International Journal of Accounting Information Systems 9*, 3 (2008), 135–153.

[21] Ferraram, A., Nikolov, A., and Scharffe, F. Data linking for the semantic web. *Semantic Web: Ontology and Knowledge Base Enabled Tools, Services, and Applications* (2013), 169.

[22] Finkel, J. R., Grenager, T., and Manning, C. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics* (2005), Association for Computational Linguistics, pp. 363–370.

[23] Flesca, S., Manco, G., Masciari, E., Rende, E., and Tagarelli, A. Web wrapper induction: a brief survey. *AI communications 17*, 2 (2004), 57–61.

[24] Freitas, A., Curry, E., Oliveira, J. G., and O'Riain, S. Querying heterogeneous datasets on the linked data web: challenges, approaches, and trends. *IEEE Internet Computing 16*, 1 (2012), 24–33.

[25] Goldstein, J. I., Ramshaw, P. D., and Ackerson, S. B. An investment masquerade: A descriptive overview of penny stock fraud and the federal securities laws. *The Business Lawyer* (1992), 773–835.

[26] Gupta, S., Kaiser, G., Neistadt, D., and Grimm, P. Dom-based content extraction of html documents. In *Proceedings of the 12th international conference on World Wide Web* (2003), ACM, pp. 207–214.

[27] Han, J., Haihong, E., Le, G., and Du, J. Survey on nosql database. In *Pervasive computing and applications (ICPCA), 2011 6th international conference on* (2011), IEEE, pp. 363–366.

[28] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759* (2016).

[29] Kalantidis, Y., and Avrithis, Y. Locally optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 2321–2328.

[30] Kapoor, R., Kejriwal, M., and Szekely, P. Using contexts and constraints for improved geotagging of human trafficking webpages. *arXiv preprint arXiv:1704.05569* (2017).

[31] Kapoor, R., Kejriwal, M., and Szekely, P. Using contexts and constraints for improved geotagging of human trafficking webpages. In *Proceedings of the Fourth International ACM Workshop on Managing and Mining Enriched Geo-Spatial Data* (2017), ACM, p. 3.

[32] Kejriwal, M., and Szekely, P. Information extraction in illicit web domains. In *Proceedings of the 26th International Conference on World Wide Web* (2017), International World Wide Web Conferences Steering Committee, pp. 997–1006.

[33] Kejriwal, M., and Szekely, P. Information extraction in illicit web domains. In *Proceedings of the 26th International Conference on World Wide Web* (2017), International World Wide Web Conferences Steering Committee, pp. 997–1006.

[34] Krishnamurthy, R., Li, Y., Raghavan, S., Reiss, F., Vaithyanathan, S., and Zhu, H. Systemt: a system for declarative information extraction. *ACM SIGMOD Record 37*, 4 (2009), 7–13.

[35] Kushmerick, N. *Wrapper induction for information extraction*. PhD thesis, University of Washington, 1997.

[36] Kushmerick, N. *Wrapper induction for information extraction*. PhD thesis, University of Washington, 1997.

[37] Kushmerick, N., Weld, D. S., and Doorenbos, R. Wrapper induction for information extraction.

[38] Lakkaraju, K., Yurcik, W., and Lee, A. J. Nvisionip: netflow visualizations of system state for security situational awareness. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security* (2004), ACM, pp. 65–72.

[39] Lavrenko, V., Allan, J., DeGuzman, E., LaFlamme, D., Pollard, V., and Thomas, S. Relevance models for topic detection and tracking. In *Proceedings of the second international conference on Human Language Technology Research* (2002), Morgan Kaufmann Publishers Inc., pp. 115–121.

[40] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S., et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web 6*, 2 (2015), 167–195.

[41] Lenci, A. Distributional semantics in linguistic and cognitive research. *Italian journal of linguistics 20*, 1 (2008), 1–31.

[42] Lerman, K., Minton, S., and Knoblock, C. A. Wrapper maintenance: A machine learning approach. *J. Artif. Intell. Res.(JAIR) 18* (2003), 149–181.

[43] Li, Y., Krishnamurthy, R., Raghavan, S., Vaithyanathan, S., and Jagadish, H. Regular expression learning for information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (2008), Association for Computational Linguistics, pp. 21–30.

[44] Liu, L., Peng, T., and Zuo, W. Topical web crawling for domain-specific resource discovery enhanced by selectively using link-context. *International Arab Journal of Information Technology (IAJIT) 12*, 2 (2015).

[45] Lopez, L. A., Duerr, R., and Khalsa, S. J. S. Optimizing apache nutch for domain specific crawling at large scale. In *Big Data (Big Data), 2015 IEEE International Conference on* (2015), IEEE, pp. 1967–1971.

[46] Maaten, L. v. d., and Hinton, G. Visualizing data using t-sne. *Journal of Machine Learning Research 9*, Nov (2008), 2579–2605.

[47] McCallum, A., and Li, W. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4* (2003), Association for Computational Linguistics, pp. 188–191.

[48] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (2013), pp. 3111–3119.

[49] Nadeau, D., and Sekine, S. A survey of named entity recognition and classification. *Lingvisticae Investigationes 30*, 1 (2007), 3–26.

[50] Nielsen, J. *Usability engineering*. Elsevier, 1994.

[51] Niu, F., Zhang, C., Ré, C., and Shavlik, J. W. Deepdive: Web-scale knowledge-base construction using statistical learning and inference. *VLDS 12* (2012), 25–28.

[52] Olszak, C. M., and Ziemba, E. Approach to building and implementing business intelligence systems. *Interdisciplinary Journal of Information, Knowledge & Management 2* (2007).

[53] Pantel, P., Crestan, E., Borkovsky, A., Popescu, A.-M., and Vyas, V. Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2* (2009), Association for Computational Linguistics, pp. 938–947.

[54] Pantel, P., Crestan, E., Borkovsky, A., Popescu, A.-M., and Vyas, V. Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2* (2009), Association for Computational Linguistics, pp. 938–947.

[55] Peng, F., and McCallum, A. Information extraction from research papers using conditional random fields. *Information processing & management 42*, 4 (2006), 963–979.

[56] Prud, E., Seaborne, A., et al. Sparql query language for rdf.

[57] Pujara, J., Miao, H., Getoor, L., and Cohen, W. W. Knowledge graph identification.

[58] Qian, R. Understand your world with bing, Mar 2013.

[59] Ramnandan, S. K., Mittal, A., Knoblock, C. A., and Szekely, P. Assigning semantic labels to data sources. In *European Semantic Web Conference* (2015), Springer, pp. 403–417.

[60] Ratner, A. J., De Sa, C. M., Wu, S., Selsam, D., and Ré, C. Data programming: Creating large training sets, quickly. In *Advances in Neural Information Processing Systems* (2016), pp. 3567–3575.

[61] Sahlgren, M. An introduction to random indexing. In *Methods and applications of semantic indexing workshop at the 7th international conference on terminology and knowledge engineering, TKE* (2005), vol. 5.

[62] Singhal, A. Introducing the knowledge graph: things, not strings. *Official google blog* (2012).

[63] Stevenson, M., and Wilks, Y. Word sense disambiguation. *The Oxford Handbook of Comp. Linguistics* (2003), 249–265.

[64] Suchanek, F. M., Kasneci, G., and Weikum, G. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web* (2007), ACM, pp. 697–706.

[65] Szekely, P., Knoblock, C. A., Slepicka, J., Philpot, A., Singh, A., Yin, C., Kapoor, D., Natarajan, P., Marcu, D., Knight, K., et al. Building and using a knowledge graph to combat human trafficking. In *International Semantic Web Conference* (2015), Springer, pp. 205–221.

[66] Tanenbaum, J. G., Williams, A. M., Desjardins, A., and Tanenbaum, K. Democratizing technology: pleasure, utility and expressiveness in diy and maker practice. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2013), ACM, pp. 2603–2612.

[67] Veak, T. J. *Democratizing technology: Andrew Feenberg's critical theory of technology*. Suny Press, 2012.

[68] Wheaton, E. M., Schauer, E. J., and Galli, T. V. Economics of human trafficking. *International Migration 48*, 4 (2010), 114–141.

[69] Wick, M., and Boutreux, C. Geonames. *GeoNames Geographical Database* (2011).

[70] Xu, J., and Chen, H. The topology of dark networks. *Communications of the ACM 51*, 10 (2008), 58–65.

[71] Zhu, X. Semi-supervised learning literature survey.

[72] Zulkarnine, A. T., Frank, R., Monk, B., Mitchell, J., and Davies, G. Surfacing collaborated networks in dark web to find illicit and criminal content. In *Intelligence and Security Informatics (ISI), 2016 IEEE Conference on* (2016), IEEE, pp. 109–114.

List of Acronyms

BoW Bag of Words

CE Counterfeit Electronics

CRF Conditional Random Field

DIG Domain Specific Insight Graphs

DQL Domain Specific Query Language

ECS Entity Centric Search

HKG Hybrid Knowledge Graph

HT Human Trafficking

IE Information Extraction

IFS Illegal Firearm Sales

KG knowledge graph

KGC Knowledge Graph Construction

LES Lightweight Expert System

MAP Mean Average Precision

NER Named Entity Recognition

NLP Natural Language Processing

NLTK Natural Language Toolkit

OCR Optical Character Recognition

OTC Over the Counter

POS Part of Speech

RI Random Indexing

RTE Readability Text Extractor

SWRL Semantic Web Rule Language

TLD Top Level Domain