A Scalable Machine-Learning Approach for Semi-Structured Named Entity Recognition

Utku Irmak
Yahoo! Labs
4401 Great America Parkway
Santa Clara, CA
uirmak@yahoo-inc.com

Reiner Kraft
Yahoo! Inc.
701 First Avenue
Sunnyvale, CA
reiner@yahoo-inc.com

ABSTRACT

Named entity recognition studies the problem of locating and classifying parts of free text into a set of predefined categories. Although extensive research has focused on the detection of person, location and organization entities, there are many other entities of interest, including phone numbers, dates, times and currencies (to name a few examples). We refer to these types of entities as *semi-structured named entities*, since they usually follow certain syntactic formats according to some conventions, although their structure is typically not well-defined. Regular expression solutions require significant amount of manual effort and supervised machine learning approaches rely on large sets of labeled training data. Therefore, these approaches do not scale when we need to support many semi-structured entity types in many languages and regions.

In this paper, we study this problem and propose a novel three-level bootstrapping framework for the detection of semi-structured entities. We describe the proposed techniques for phone, date and time entities, and perform extensive evaluations on English, German, Polish, Swedish and Turkish documents. Despite the minimal input from the user, our approach can achieve 95% precision and 84% recall for phone entities, and 94% precision and 81% recall for date and time entities, on average. We also discuss implementation details and report run time performance results, which show significant improvements over regular expression based solutions.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Retrieval Models; I.2.7 [**Artificial Intelligence**]: Natural Language Processing

General Terms

Algorithms, Experimentation

Keywords

Boostrapping algorithm, weakly-supervised learning, NER

1. INTRODUCTION

There is a significant amount of work in the field of named entity recognition (NER), where the goal is to locate and classify parts of free text into a set of predefined categories. These categories include names of persons, organizations, locations and also expressions of phone numbers, date and time entities, monetary values etc. NER has been studied extensively as it is quite important for a large number of applications including question answering, user-centric entity detection systems, content de-identification and information retrieval (e.g., web search engines).

In this paper we focus on a subset of named entities, which we refer to as *semi-structured named entities (SSNE)*. We formally define SSNE as those whose entity string may syntactically contain only: (1) digits and white spaces; (2) a finite set of non-letter characters; (3) a finite set of domain-specific terms (tokens), if applicable. In other words, we define a named entity type to be semi-structured, if its instances (of interest) conform to these restrictions. Although the person, location and organization entities do not conform to them, phone or fax numbers, date, time, monetary, weight, length, percentage entities do follow such syntactical properties naturally. However, these properties are usually not well-defined and change greatly from language to language, and even from region to region for the same language. As we will describe shortly, traditional approaches for SSNE recognition require significant amount of manual effort, either in the form of handcrafting rules or in the form of labeling examples, and therefore they do not scale when many entity types, languages and regions need to be supported.

Our goal in this work is to provide a scalable solution for the detection of semi-structured named entities for many entity types, languages and regions. To achieve this, we propose a novel three-level framework: The framework employs text mining techniques and runs a novel two-step bootstrapping algorithm on a large corpus of text documents, and feeds the output to a machine learning algorithm to create the final detector. Our approach significantly reduces the manual effort and requires very little domain knowledge. In fact, in some cases, the input to the system can be provided by non-speakers of the target language. We describe our approach in detail for the detection of phone or fax numbers and also date and time entities. Note that the techniques can be extended for the detection of other SSNE types. Below, we describe some applications that can benefit from our approach, discuss existing solutions and their limitations, and finally specify our contributions.

1.1 Applications

User-centric entity detections systems: These systems typically detect various types of entities in web pages, email messages, content on mobile devices and create intelligent hyperlinks to relevant applications [12, 31]. For phone numbers, most user-centric entity systems offer shortcuts to address books or online phone services, and for date and time entities they offer calendar applications. So these systems can immediately benefit from a scalable solution.
Web search engines: Search engines can leverage semi-structured entities recognized in the web pages to improve the overall relevancy for certain types of queries. For example, if the query intent is to find contact information, those pages that contain phone numbers can be boosted for a higher relevancy. Also, a number studies

including [3, 4, 16] use phone numbers (more specifically the country and area codes) as features for the goal of extracting geographic information from web pages. Web search engines can use date and time entities to improve results on time sensitive queries.
Data extraction, integration and classification: Detection of semi-structured entities in text can greatly help question answering systems (e.g., [13]), as well as systems focusing on information integration over the Web (e.g., [5]). Many comparison shopping engines need to identify money (price), time, and phone number (contact) information on target web pages; scalable non-site specific techniques can help to boost confidence in extraction. Also, semi-structured named entities can be quite useful features for the classification systems. For example, the detection of currency entities might be a good indicator that the content is about finance.
Content de-identification systems: Another application that would immediately benefit from this solution is the de-identification of documents. This is quite crucial for the health research community: No medical record can be released or shared with others unless all the private data, including phone numbers, dates and times are removed [10, 30].

Before proceeding, we would like to note that for some applications the recognition of date and time entities alone is not sufficient, and a second step, which understands the entity and creates a standard or machine understandable representation, is required. This step is usually referred as normalization or resolution of the temporal information [24], and is crucial for event ordering. In this paper, we focus on the recognition task which is a prerequisite for the normalization. Some studies, such as [25], propose techniques to make the normalization process language independent.

1.2 Existing Solutions and Their Limitations

We will have a detailed description of the related work in the rest of the paper, however, we give an overview of the existing solutions here, as their limitations motivate this work.
Rule-based approaches: Since SSNE have some syntactical properties by definition, rule-based solutions work to some extent. For this, regular expressions are commonly used, however, they usually require a significant manual effort to generate. Since production-quality detectors need to handle many cases, the expressions can become more and more complicated. To give the reader some idea, the regular expression used for phone number detection in Y! Mail (for English in US via [31]) contains more than 600 characters as it is required to have large coverage and handle some ambiguous patterns.

Clearly, the phone number conventions in US are different than in Sweden, but also in the UK. So a different regular expression needs to be developed for every target language and region. (These differences, which arise due to local customs and other reasons are also discussed in [15, 16].) Imagine an application needs a phone number detector for German in Germany, or Turkish in Turkey. In practice, the following steps are taken for this task: (1) Editorial study: People with the domain knowledge collect instances and identify patterns that are commonly used; (2) Development: Programmers develop regular expressions for the patterns specified by editors; (3) Maintenance: The first two steps usually cannot be done perfectly initially, so a number of iterations usually follow as editors and developers identify problematic cases, such as missing or ambiguous patterns. So supporting many languages and regions with regular expressions does not scale (even for phone numbers, which may be considered as a simple type), since the solution requires significant amount of coordinated effort.

A rule-based approach for date and time detection is described in [27]. Date and time detection is a more difficult problem as it involves many more patterns, and also due to ambiguity problems, especially with simple patterns: "1/18" can be a fraction, or a date. The following contexts contain month and year entities: "We will meet this may", "2000 is a nice year". But the same patterns fail in the following contexts: "This may not be ok", "Windows 2000 is an operating system". To avoid such mistakes, rule-based solutions are usually optimized for the most obvious and common cases, at the cost of coverage. For the implementation, regular expressions may be infeasible for some applications, as they are computationally expensive. Many programming languages, such as C++, Java, Perl, provide date and time libraries, however, their purpose is to provide infrastructure for calendar operations (parsing, comparison, interval computation of dates, etc.) but not directly NER capabilities.
Machine-learning approaches: Due to shortcomings and efforts required as described above, machine learning approaches are highly preferred. The supervised approaches (e.g., [6, 11, 24]) usually require a large annotated corpus, and in most cases, they rely on Part-of-Speech tags, as important features (which again requires another training). These aspects make supervised techniques undesirable from the scalability perspective, considering that many entity types, languages and regions need to be supported. So we follow a weakly-supervised learning (bootstrapping) approach to address these limitations, and develop text mining techniques for SSNE recognition. In Section 2, we review the bootstrapping algorithm in detail, and use it as our baseline in our evaluation.

1.3 Contributions

In this paper, we study the problem of detecting semi-structured named entities (such as phone numbers, date and time entities etc.) in text, and propose a weakly-supervised learning approach by developing new bootstrapping and text mining techniques. Our major contributions are listed below:

(1) We propose a novel three-level bootstrapping framework, whose main advantage is to allow attacking the recall and precision aspects separately, whereas, traditional bootstrapping algorithms try to balance them at the same time, or require additional resources. The last level employs machine learning techniques to create the final model, which generalizes the extraction rules further, and can be used on new content for real time detection.

(2) We adopt the framework for semi-structured entity detection, and propose a two-step bootstrapping algorithm, which can learn both regular expressions and contextual rules at every iteration.

(3) We evaluate the proposed techniques extensively on English, German, Polish, Swedish and Turkish documents for phone, date and time entities. Despite its minimal input, our approach can achieve 95% precision and 84% recall for phone entities, and 94% precision and 81% recall for date and time entities, on average.

(4) We discuss implementation details for the real-time detection of semi-structured entities. We report performance test results for phone number detection; the results show that the proposed approach is an order of magnitude faster than a production-quality regular expression based solution.

In the remainder of the paper, we review the traditional bootstrapping techniques in Section 2, describe our three-level framework in Section 3, and adopt the framework for the semi-structured entity detection in Section 4. We then present our experimental results in Section 5. We review implementation issues and performance results in Section 6.

2. BOOTSTRAPPING ALGORITHM

In this section, we review the baseline bootstrapping algorithm, which is an iterative algorithm, and usually runs on a large collection of text documents. The algorithm has been studied extensively and employed in a number of applications. Important applications include: (1) The extraction of the semantic lexicons, where the term semantic lexicon refers to a dictionary of words labeled with semantic categories, such as vehicles, animals, events etc. [22, 28]; (2) The recognition of named entities such as persons, organizations or locations [7]; (3) The extraction of pair-wise named entities that share a certain relation, where the relation can be *organization-hasheadquarters-in* for organization and location pairs [1], *book-written-by* for book title and author pairs [2], and *person-bornin-year* for person and year entities [19].

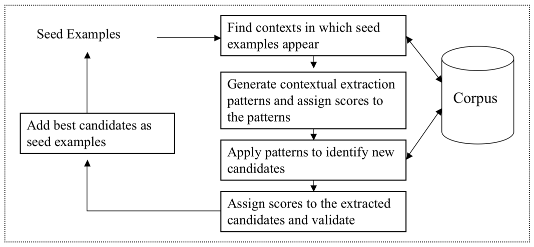

Seed Examples
Find contexts in which seed examples appear
Generate contextual extraction patterns and assign scores to the patterns
Corpus
Add best candidates as seed examples
Apply patterns to identify new candidates
Assign scores to the extracted candidates and validate

Figure 1: Bootstrapping algorithm

Although the proposed solutions have variations, as will be discussed shortly, the main idea can be summarized as follows, and illustrated in Figure 1: The system starts with a small number of seed examples, which are provided by the user. The system then finds occurrences of these examples in a large set of documents. By analyzing these occurrences, the system generates contextual extraction patterns (rules) and assigns confidence scores to the patterns. After this step, the system applies the extraction patterns to the documents and extracts new candidates. Based on some validation mechanism, the system assigns scores to the extracted candidates, and chooses the best ones to add to the seed set. Then the system starts over to perform many similar iterations, and at every iteration it learns more patterns and can extract more instances.

Clearly, the success of the bootstrapping algorithm depends on how the patterns are generated, and also the validation mechanism used for choosing the best candidates, which are then used as seed examples in the next iteration. The work in [2] tries to address these issues by assigning a specificity score to the patterns, and proposes to require that candidates are generated by multiple patterns before moving them to the seed set. For efficient computation, the specificity of a pattern is defined to be the length of the pattern itself and those patterns that have too low scores are rejected.

In [22] the authors study the problem of extracting semantic lexicons on a text corpus, for a number of different semantic categories, such as locations, person titles, companies, weapons. The system relies on *AutoSlog* [21] to extract the patterns, which are in the form of prefix and suffix strings. The score of an extraction pattern is computed using the *RlogF* metric [21], which employs the number of both unique lexicons and unique noun phrases produced by the pattern. The idea behind this metric is to achieve a good balance between precision and recall. In their multi-level approach, the authors propose to add very few number of examples to the seed set at every iteration, so that patterns are re-evaluated with a seed set that is growing more conservatively. The score of a candidate is computed based on the scores of its matching patterns.

The work in [1] focuses on the extraction of organization and location pairs that have *has-headquarters-in* relation. The authors propose to use named-entity tags (for organization and location entities) in their pattern representation, and for this, they rely on a named-enity tagger, MITRE Corporation's Alembic Workbench [1]. To compute the score of a pattern, the authors adapt a metric similar to *RlogF*. The confidence of a candidate tuple is based on the scores of its matching patterns. The authors also introduce a parameter to control the learning rate of the system, so that the system trusts new examples less while creating patterns.

In [19], the authors propose a novel bootstrapping approach to achieve large-scale and fast iterative progression; the algorithm avoids the use of syntactic parsers, named entity recognizers and gazetteers, etc., but relies on the availability of distributionally similar words [14], which needs to be extracted in advance for the target language. The proposed system generates basic extraction patterns similar to above systems, however, with the help of pre-generated set of similar words, it generalizes these basic patterns at every iteration, allowing higher coverage and faster progression. For the validation, the authors check if the candidates are distributionally similar to the seed examples. They also use PMI score [29] and a completeness score.

3. THREE-LEVEL FRAMEWORK

In this section, we describe our three-level bootstrapping framework, which is the basis of our approach for semi-structured named entity detection. The framework is illustrated in Figure 2.

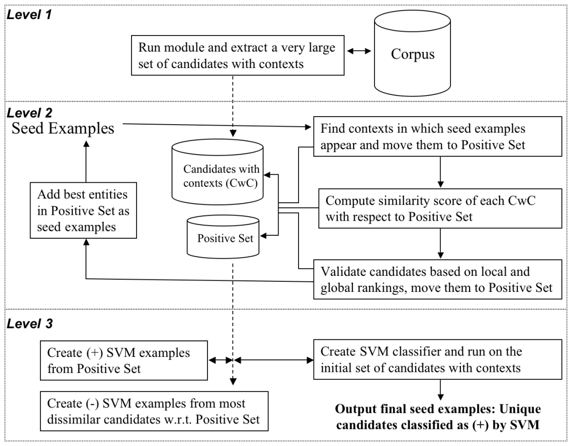

Level 1
Run module and extract a very large set of candidates with contexts
Corpus
Level 2
Seed Examples
Find contexts in which seed examples appear and move them to Positive Set
Candidates with contexts (CwC)
Add best entities in Positive Set as seed examples
Compute similarity score of each CwC with respect to Positive Set
Positive Set
Validate candidates based on local and global rankings, move them to Positive Set
Level 3
Create (+) SVM examples from Positive Set
Create SVM classifier and run on the initial set of candidates with contexts
Create (-) SVM examples from most dissimilar candidates w.r.t. Positive Set
Output final seed examples: Unique candidates classified as (+) by SVM

Figure 2: Three-level framework

3.1 First Level

As described in Section 2, the traditional bootstrapping algorithm learns extraction rules by analyzing the occurrences of the seed examples in the corpus, which in return enables the extraction of more seed examples. Since the discovery of new examples depends on the extraction rules learned, they have to be general enough. However, too general rules introduce noisy examples, causing a catastrophic effect in the next iterations.

The main motivation for the proposed framework is to provide a mechanism to avoid the problem of balancing this trade-off. To achieve this, the first level requires the implementation of a module that can extract a very large set of candidates from the corpus. The main goal here is to have high recall rates and not worry about the precision. The implementation clearly depends on the task at hand. (For SSNE detection, such modules can be implemented fairly easily due to their syntactical properties, which come by definition).

In the first level, the module is run on the corpus and a very large set of candidates are extracted with their contexts. With context, we refer to a window that surrounds the candidate (in practice the window size could be twenty words). This usually reduces the total data size significantly, since most parts of the corpus are eliminated.

3.2 Second Level

The first level of the framework is expected to generate a very large set of candidates with their context windows. The goal of the second level is to identify the target entities in this set with very high precision. To achieve this goal, we employ a bootstrapping algorithm, where the input to the system is only a small number of seed examples. Clearly, one could conservatively use the traditional algorithm described in Section 2, which learns contextual extraction rules iteratively and validates the candidates based on the specificity of the rules that extracted them. However, we propose a different approach: At every iteration, (1) The candidates with their contexts are moved to a *Positive Set* if the candidate matches a seed example, (2) The confidence score of each remaining candidate is computed based on the similarity of its contexts to the positive set and valid ones are moved to the positive set, (3) The positive set is analyzed and the qualifying entities are included in the seed set, based on some statistical data. So compared to the baseline, the validation does not rely on the extraction rules but on the contextual similarities between the candidates and the positive set. We define the similarity functions used in this paper below:
Cosine Similarity: This metric uses the bag of words model and assigns scores to matching terms based on their *tf*idf* scores, where *tf* stands for the term frequency and *idf* stands for the inverse document frequency [23]. If the contexts of a given candidate share many significant terms with the positive set, then the final similarity score of the candidate will be high.
Cosine Similarity with Distances: This is similar to above, but it also uses the position of the context terms in order to boost the scores of the closer terms. The *tf*idf* scores are divided by the *log(distance+1)*, where the distance represents the number of white spaces to the candidate (e.g., the distance of the term next to the candidate is 1 and it increases by 1 for the following terms).
String Length: This identifies the longest prefix and suffix strings shared by the candidate and positive set items and uses the sum of both string's lengths as the similarity score.

We compute the score of a candidate c_i in context C, by comparing C to the contexts seen in positive set:

$$score(c_i, C) = \frac{\sum_{k=1}^{N} similarity(Context_k, C)}{N} \tag{1}$$

Note that our approach does not necessarily require a pairwise comparison of all candidates and the positive set. After every iteration, depending on the similarity function used, a hash table or some other efficient structure can be constructed from the positive set to allow efficient computation of the similarity score. We validate candidates based on two rankings and the qualifying ones are moved to positive set:
Local Ranking: This simply ranks all the candidates seen in the current iteration based on their similarity scores, and identifies the top candidates (say top 1%, or equally ranking threshold of 0.01).
Global Ranking: This keeps track of all the scores computed in previous iterations and makes sure the best candidates are also ranked high globally (say top 2%, or equally ranking threshold of 0.02).

At every iteration, the positive set is analyzed and the best entities are added to the seed set. We describe the details in Section 4, for now, it is sufficient to say the qualification is based on statistical data, such as the frequencies of the entities and seed examples. We stop the bootstrapping algorithm when the positive set growth slows significantly (e.g., if it grows by less than 1%). This eventually happens as the similarity scores decrease after many iterations and ultimately very few qualify according to the global ranking.

3.3 Third Level

The first level creates a very large set of candidates with high recall and the second level identifies more of the seed examples with high precision. The goal of the third level is to create a final model using machine learning techniques to balance and further improve the overall precision and recall rates. Below we describe the feature spaces and models used.
Feature space: The first representation follows a bag of words model and uses individual terms as features. In practice, very rarely occurring terms may be excluded from the feature space. As a second feature space, we generate multi-term phrases using pointwise mutual information [29] and include them in the bag of words model. The third feature space we employ does not follow a bag of words model, but uses prefix and suffix strings that immediately precede or follow the candidates (or seeds) as features. Each unique delimiter found is represented by a unique feature in the space (we call this approach delimiter-based feature space).
Model creation: In our approach, we employ an implementation of support vector machine (SVM) to create the models. An open source library for SVM models is also available in *SVM*^{light} [1]. [9] describes an open source library [2] for large-scale linear classification without kernels. For the model creation, we can follow two approaches: (1) We can use the positive set alone and create a one-class model [26], or (2) In addition to the positive set alone, we can also find most dissimilar candidates with respect to the positive set, according to Equation 1, and use them as negative examples. Once the SVM model is created, it is run on all the candidates created in the first level and unique entities classified as positive are output as the final set of seed examples. This model can also be used on new documents in the future for a real time detection.

3.4 Discussion

We adopt this framework in Section 4 for SSNE recognition. In our case, it is quite trivial to implement the first level by taking advantage of the syntactical properties of SSNE. However, a number of large scale relation extraction tasks involve named entities (see Section 2), and the first level can be fairly easily implemented for those cases as well, by leveraging off-the shelf named entity tagging tools. In fact, some approaches, such as [1], already employ named entity tagging tools. So we discuss the advantages of the framework below as it can be used for such tasks.

Instead of trying to balance the precision and recall at the same time, the framework targets for the highest recall and precision values at the first two levels, and explicitly separates these two tasks. The second level acts conservatively to build a set of examples with high precision, and can take advantage of all the information and global statistics made available at the first level. More specifically, the validation of the candidates can be done more reliably, because now all the occurrences (contexts) of the candidate is available for this decision. In other words, the score of a candidate can be computed not only based on the patterns that extract it, but also other contexts that the candidate appears in. Some approaches such as [8] try to achieve a similar goal by using search engine results. The third level employs machine learning techniques and uses the output generated by earlier levels. At this level the problem is reduced to a classification problem, and the model learned goes beyond simple extraction rules that the bootstrapping algorithm can potentially generate. Also, this model can be used later on new documents for real time detection when the implementation of the first level is incorporated.

[1]http://svmlight.joachims.org
[2]http://www.csie.ntu.edu.tw/~cjlin/liblinear

Level 1	**Input**
Tokens	–
Pattern	7 to 13 digits separated by -+()./[]_* and space
Level 2	**Input**
SeedPattern	`(ddd)ddd-dddd`
SeedTerms	{phone, telephone, contact, fax, call}

Table 1: Input for phone detection in English (US)

4. SSNE RECOGNITION

In this section, we adopt the three-level framework for semi-structured named entity detection, and propose a novel two-step bootstrapping algorithm, which is employed at the second level of the framework. For clarity, we describe our approach for phone numbers in Section 4.1 as a case study, and adopt the approach for date and time detection in Section 4.2.

4.1 Phone Number Detection

First Level: The implementation of the first level is quite straightforward for SSNE as they follow certain syntactical properties by definition (see Section 1 for the definition). The module for phone number detection simply finds sequences of digits that may be separated by space character and following characters: -+()./[]_* (no tokens applicable in this case). For any sequence found, the module requires that the number of digits is between seven and thirteen, and also the number of non-digit characters does not exceed ten. This module gives a large set of candidates which contains possibly all phone numbers, and it can be used for many languages.

Second Level: A two-step algorithm: We now describe our two-step bootstrapping algorithm and illustrate it for phone number detection (Figure 3). The second level input from the user is one seed regular expression and optionally a small number of seed terms (Table 1). The algorithm follows a two-step approach while moving the candidates to the *Positive Set*. In the first step, the algorithm runs the regular expression on all the candidates, and considers any candidate that matches the expression to be a positive match, only if its context window contains at least one of the seed terms. The reason for using the optional seed terms is to improve our confidence and avoid false positives. In the second step, the algorithm computes similarity scores for each candidate with respect to the existing positive set, and adds the best candidates to the set. As described in Section 3.2, the algorithm uses local and global rankings to determine the best candidates. Once these two steps are completed, the algorithm analyzes all the examples in the positive set, and may add new seed regular expressions or new seed terms.

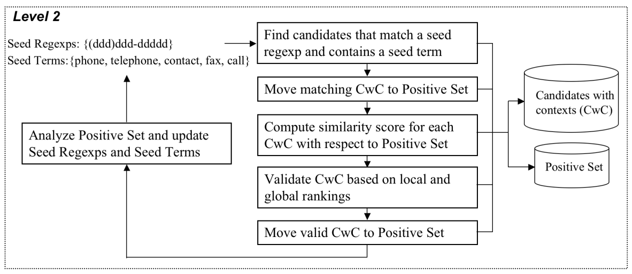

Level 2
Seed Regexps: {(ddd)ddd-ddddd}
Seed Terms: {phone, telephone, contact, fax, call}
Find candidates that match a seed regexp and contains a seed term
Move matching CwC to Positive Set
Compute similarity score for each CwC with respect to Positive Set
Validate CwC based on local and global rankings
Move valid CwC to Positive Set
Analyze Positive Set and update Seed Regexps and Seed Terms
Candidates with contexts (CwC)
Positive Set

Figure 3: Two-step bootstrapping algorithm

Adding new regular expressions: The algorithm makes a pass over the positive set, and replaces the digits found in the phone numbers with a special character. Then it identifies the most frequent two or more non-seed expressions to be added as seed regular expressions. However, to ensure that no incorrect expressions are added, the algorithm requires a minimum number of occurrences. This number

Level 1	**Input**
LongWeekdays	{monday, tuesday, ..., sundays}
ShortWeekdays	{mon, tue, ..., sun}
LongMonths (LM)	{january, february, ..., december}
ShortMonths (SM)	{jan, feb, ..., dec}
LongTimeZone	{alaska daylight, ..., pacific time zone}
ShortTimeZone	{akdt, ..., pt}
GeneralKeywords	{yesterday, today, ..., nights}
PreText	{this coming, due, next, ..., past}
PostText	{am, a.m, ..., p.m.}
Connectors	{from, at, on, -, –, ..., until}
NumbersWithText	d*{am, a.m, ..., st, nd, rd, th}
Pattern	digits separated by ()+-.,:/#
Level 2	**Input**
SeedPattern	`(SM|LM) D(D)?(th|st|nd|rd)?,?  dddd`
SeedPattern	`D(D)?:DD (a|p)(.)?m(.)?`
SeedTerms	–

Table 2: Input for date and time detection in English (US)

can be obtained based on the occurrences of the most frequent and the least frequent seed expressions, (f_{max} and f_{min} respectively), such as $max(f_{max}/c_1, f_{min}/c_2)$, where c_1 and c_2 are constants. For phone number detection, we used values of 5 and 1, for c_1 and c_2, respectively. Clearly, other functions as well as some fixed numbers could be used for this.

Adding new seed terms: The algorithm learns and adds new seed terms by identifying frequently occurring context terms in the positive set. To achieve this, it computes *tf*idf* values of all the context terms seen and identifies the top ones. Then it uses the same function as above to ensure they are not noisy and adds the qualifying terms to the seed terms set.

Third Level: Once the bootstrapping algorithm stops, we follow the approach described in Section 3.3 and create a model using machine learning techniques. For the classification model, we create negative examples by identifying the most dissimilar candidates, using Equation 1. But now, since the semi-structured entities follow certain syntactical formats, the similarity functions can also leverage these expressions, in addition to the contextual similarities discussed. More specifically, the score in Equation 1 is now multiplied by the ratio of positive examples that have the same pattern as the candidate to all positive examples found. Similarly, each unique pattern seen in the training set is used as a feature in the model.

4.2 Date and Time Detection

We now apply the proposed techniques for the detection of date and time entities. Before proceeding, we would like to mention that our focus is to detect explicit entities (e.g., “1/1/2009”, “on Monday”, “4/15”), implicit references (e.g., “last Monday”), intervals for both (e.g., “2-3pm”, “mon to tue”) and also periodic expressions (e.g., “on Fridays”), but not durations (e.g., “10 minutes”) or event anchored expressions (e.g., “two days after the accident”). Clearly, a large number of expressions, or labeled training data would be required to capture all these target entities with the existing solutions, and putting the same effort for every language may not be feasible.

First Level: Date and time entities (differently than phone numbers) use a limited set of tokens. We split this set into logical subsets and show the input needed for the first level implementation, in Table 2. Each input field runs separately on a given page and the detections are merged to represent a single entity if they are separated by white space characters. The fields for weekday, month, general keywords and time zones are self-explanatory. *PreText* simply lists a set of keywords to be detected if they precede another detection, and similarly *PostText* defines terms that may follow another detection. *Connector*s help to get two other detections merged into one when they appear in the middle. *Pattern* is similar to the one used for phone detection and used for all languages: It simply matches any digits that are separated by ()+-.,:/#. Clearly, this matches a large number of digit sequences that possibly contain all the numerical date and time expressions. Lastly, *NumbersWithText* defines some patterns that involve both digits and letters that are not separated by white space characters, if used in the language. Although constructing this input might initially seem difficult, it is significantly simpler than enumerating all possible date and time patterns, and developing them via regular expressions or other languages. Our users (native speakers of the target languages) found the task quite easy, but online dictionaries or thesauruses could also be used either to automate this process or to assist the users.

This module can not only successfully capture the numerical date and time entities (e.g., "1:10", "11.15.2005", or "11/15"), and formally written entities (e.g., "Thu Apr 30 11:33:52 PDT 2009"), but also long entities (e.g., "on monday april 6th from 5:15pm to 6:15 pm pst"). This is possible as it merges contiguous detections to be a single entity, if they are separated by white space characters. This ideally captures all the date and time entities of interest based on the user input, which in fact, expresses the desired detection capabilities in a very flexible manner. Clearly, this flexibility causes many ambiguous and noisy data to be extracted as candidates as well, including all the numerical values (such as "13/13", "100", "15-15.9999") and text pieces (such as in contexts "Usa Today", "Jan sings well", "sun is shining"). However, the next levels in the framework are designed particularly to handle these cases.

Second and Third Levels: These work exactly same as in the phone number detection. The input for the second level is shown in Table 2: We use two seed regular expressions (one for date and one for time entities), and we do not require any seed terms, since these expressions are already strong and contain some keywords.

As mentioned above, the contiguous detections are merged into a single date and time entity, if they are separated by space characters. Although this approach significantly improves coverage and simplifies the input from the user, it may cause some over-selection problems (i.e including extra text as part of the entity). For example, in context "... in year 2002 3rd place was ...", the approach would merge 2002 and 3rd into a single entity. We address this issue by leveraging the distribution of the patterns: When the algorithm stops, we compute the frequencies of all the patterns found in the corpus. Then for each entity found, we require its pattern to have a minimum frequency and trim the surrounding text until the requirement is satisfied. This effectively adjusts the entity boundaries based on the pattern frequencies observed in the corpus.

5. EVALUATION AND RESULTS

We now present our experimental setup and results for phone and date-time entity detection in English, German, Polish, Swedish and Turkish, for the regions US, Germany, Poland, Sweden and Turkey, respectively. For each language, we use a corpus of 1GB, which contains about 200K random web pages from a large crawl. We run the proposed algorithms on this corpora; due to the infeasibility of examining every detected entity we use a sample set. In our case, the first level identifies all the target entities of interest by definition, in addition to the other possibly large noisy data. We sample this output, and for each language we use 250 and 300 instances for phone numbers and date-time entities, respectively. Each instance is then judged by a native speaker of the language: "*No*": This is not a target entity; "*Yes*": This is a target entity and boundaries are correct; "*Over-Selection*": This is a target entity, but some unrelated text is included; "*Under-Selection*": This is a target entity, but some part of the entity is not included. We evaluate the algorithms using these judgments (boundary problems reported separately, if any). Our key metrics in the evaluation are precision and recall. By definition, *Precision = TP/(TP+FP)* and *Recall = TP/(TP+FN)*, where *TP*, *FP* and *FN* represent true positives, false positives and false negatives, respectively. We focus on two target behaviors in the evaluation: (1) Highest possible precision with as high recall as possible, as many target consumer applications require, (2) Highest F-measure score, where F-measure is the weighted harmonic mean of precision and recall, defined as $F_\beta = (1+\beta^2).precision.recall/(\beta^2.precision + recall)$. For the parameter β, we use the value 1, which weighs the precision and recall equally. In our experiments, we parse the html tags, lower case all the characters and remove the surrounding punctuation characters in the context window. We tried the alternatives for all the algorithms and the results were either worse or comparable.

5.1 Phone Number Detection

We first review the bootstrapping algorithms below, and perform some initial experiments to choose the best termination settings, as well as to get some insights on various feature sets and models used in the three-level framework. For these initial experiments, we use a collection of English (US) web pages randomly chosen from a large crawl. The size of the collection is again 1GB and it contains 207K unique documents. We use the input shown in Table 1.

Baseline: We adopt the bootstrapping algorithm described in Section 2 as our baseline algorithm. Clearly, providing individual phone numbers as seed examples would not achieve the desired behavior; the numbers may not even exist in the corpus. So we use the following approach: We run the seed regular expression on the corpus and require occurrence of at least one seed term. This generates more than 1000 examples (positive set) in this corpus. We provide them to the baseline algorithm to be used in the creation of contextual extraction patterns. Recall that the contextual extraction patterns are based on prefix and suffix context strings of the entities identified. The confidence score is computed based on the length of the pattern, as was done in [2]. So the candidates which are generated by longer patterns get higher scores at every iteration. Since we run the baseline algorithm on the contexts created by the first level of our framework, they are ensured to contain phone candidates. For fair comparison, we also adopt the local and global ranking mechanism, described in Section 3.2, which defines the terminating condition. Recall that at every iteration, the candidates are sorted based on their scores, and those candidates that get ranked high locally (specified by the local ranking threshold) are moved to seed set only if their scores are also ranked high globally (specified by the global ranking threshold). We test with all pairwise combinations of local ranking threshold values of {0.0005, 0.001, 0.005, 0.01, 0.02, 0.03}, and global ranking thresholds values of {0.005, 0.01, 0.02, 0.10, 0.15}; this is done for all algorithms.

Similarity-based bootstrapping algorithm: We keep everything same as the baseline algorithm, but replace the candidate scoring mechanism with the similarity functions described in Section 3.2. We test all three functions, and the best results are obtained with the longest string length similarity function, so we use this one.

Two-step bootstrapping algorithm: We employ the two-step bootstrapping algorithm (Section 4.1), which moves the candidates to the positive set not only based on the contextual similarities, but also when they are matched with seed regular expressions and terms.

Three-level framework: In the third level, we create an SVM model using the positive set generated by the two-step algorithm (Section 3.3): This can be one-class SVM model (using only positive examples), or C-support vector classification (using both positive and negative examples, where the latter are obtained from the

Algorithm	Setting	Precision	Recall	F-measure
Baseline	0.01 and 0.05	1.00	0.08	0.15
Sim-based	0.03 and 0.025	1.00	0.32	0.23
Two-step	0.02 and 0.05	1.00	0.73	0.84
Baseline	0.03 and 0.15	0.77	0.50	0.61
Sim-based	0.01 and 0.10	0.81	0.58	0.68
Two-step	0.03 and 0.10	1.00	0.73	0.84
Algorithm	**Setting**	**Precision**	**Recall**	**F-measure**
Three-level (C-svm)	BoW	1.00	0.90	0.95
	Regexp	0.96	0.97	0.96
	BoW+Regexp	1.00	0.91	0.95
Regexp	–	1.00	0.85	0.92

Table 3: Initial results for phone number detection for the best precision, best f-measure, three-level framework and regexp

most dissimilar 25% of all non-positive candidates). In the experiments, C-svm performed significantly better and we do not include one-class SVM results here due to space constraints. For the SVM kernels, we test with linear, rbf and poly kernels with the default settings, and choose to use the linear model since its results were either comparable or better. We experimented with the following contextual feature spaces: bag of words (BoW), BoW with multi-term phrases, and delimiter-based feature space. The delimiter-based feature space alone performed significantly worse than the others, and including multi-term phrases did not improve BoW. These observations are inline with our intuition and due to space constraints we do not include the results here. We also performed experiments to understand the effect of contextual and regular expression features; the combined set performs best, as expected. With C-svm model, a probability estimate (confidence score) is returned for each instance. In the initial experiments, we used the default threshold value of 0.5, however, a higher value can be used for a better precision at the cost of a reduced recall, or vice versa.
Regexp: We include the results of a regular expression based solution for English (US), which is currently being used in a production system [31] on Yahoo! Mail. The expression is manually tuned carefully over time and contains more than 600 characters.

We first perform experiments to find the termination settings that yield the best precision and best f-measure values for each algorithm (Table 3). As we can see, the two-step bootstrapping algorithm greatly improves the baseline and similarity-based algorithms, which only employ contextual extraction rules. The three-level framework, which internally employs the best precision two-step bootstrapping algorithm, performs significantly better than the baseline approach. Compared to the regexp approach, which requires a significant amount of effort, three-level framework achieves a comparable perfect precision and improves the recall.
Experiments in Five Languages: We obtain a new collection for English (US) and collections for German, Polish, Swedish and Turkish, as described earlier. We use the parameters according to the initial experiments performed above. The input (one seed regular expression and about 5 seed terms, as shown in Table 1) is obtained as follows: We check the websites of three international companies (Microsoft, Citibank, and IBM in our case) and identify one common phone expression for each language; during this process we favor patterns that have the most non-digit characters. For the seed terms, we use online dictionaries for the translations. We would like to note that these tasks were done by a user who does not speak German, Polish or Swedish, and the judgments are performed by native speakers of the target languages.

In Table 4, we present the precision and recall values obtained for the two settings of interest: best precision (left) and best f-measure (right). We see similar trends as in the initial experiments, not only for English, but other languages as well. As mentioned earlier, the SVM model in the three-level framework produces a probability estimate (confidence score). Although we used the default value of 0.5 in our experiments, we can use different threshold values to boost either precision or recall, at the cost of the other. Next, we test various threshold values and report the precision and recall values in Figure 4. The best values are also included in Table 4 to illustrate the potential improvements.

Lang	Algorithm	P1	R1	F1	P2	R2	F2
English	regexp	1.00	0.79	0.88	1.00	0.79	0.88
	baseline	1.00	0.07	0.13	0.69	0.58	0.63
	sim-based	1.00	0.23	0.37	0.62	0.56	0.59
	2-step	1.00	0.52	0.69	0.98	0.65	0.79
	3-level (0.50)	1.00	0.77	0.87	1.00	0.77	0.87
	3-level (best)	1.00	0.89	0.94	0.99	0.95	0.97
German	baseline	1.00	0.05	0.09	0.62	0.60	0.61
	sim-based	0.95	0.22	0.35	0.98	0.74	0.84
	2-step	0.96	0.30	0.45	0.99	0.76	0.86
	3-level (0.50)	1.00	0.66	0.80	1.00	0.66	0.80
	3-level (best)	1.00	0.78	0.88	0.99	0.84	0.91
Polish	baseline	0.86	0.21	0.34	0.38	0.49	0.43
	sim-based	1.00	0.42	0.59	0.31	0.46	0.37
	2-step	0.96	0.42	0.59	0.20	0.49	0.29
	3-level (0.50)	0.95	0.66	0.78	0.95	0.66	0.78
	3-level (best)	0.95	0.66	0.78	0.87	0.85	0.86
Swedish	baseline	1.00	0.07	0.14	0.35	0.54	0.42
	sim-based	1.00	0.24	0.39	0.73	0.56	0.63
	2-step	1.00	0.74	0.85	0.29	0.85	0.44
	3-level (0.50)	0.92	0.90	0.91	0.92	0.90	0.91
	3-level (best)	0.92	0.90	0.91	0.91	0.94	0.92
Turkish	baseline	1.00	0.10	0.18	0.47	0.62	0.54
	sim-based	1.00	0.19	0.32	0.49	0.62	0.55
	2-step	1.00	0.33	0.50	1.00	0.57	0.72
	3-level (0.50)	0.98	0.67	0.79	0.98	0.67	0.79
	3-level (best)	1.00	0.64	0.78	0.87	0.92	0.90

Table 4: Phone number detection results in five languages

To get a sense on the phone patterns used in each language, we list the most frequent 10 patterns in Table 5. We also present the frequency of the patterns along with the total number of unique patterns (that occur at least three times). As we can see, the phone numbers used in English (US) follow well-accepted conventions, however, this does not seem to be the case for other languages. For example, the number of phone patterns used in German is quite large, and the most frequent one only has a frequency of about 4.5%. Based on these statistics, we would not expect regular expressions for other languages to be as successful as in English (US).
Detection in Multi-Language Documents: Since some documents or messages are authored by multi-lingual people, it is possible that the authors follow the conventions of one language although they are writing in another one. To address this edge case, we employ the three-level framework, and create a model using only the contextual features. Under this setup, the system could use two models: one with the full feature set to capture the common case as usual, and another one with only contextual features but with a higher confidence threshold. So the second model would cover the edge case with less recall. To test this, we rerun the experiments with only contextual features with a confidence level of 0.75, and report the results in Table 6. As we can see, the solution can detect phone numbers (that may have unexpected formats) with high precision, by just using the contextual features.

5.2 Date and Time Detection

We use the same setup and metrics as we do for phone number detection, which is described above. Below, we perform our initial experiments to find the best settings. In Table 7, we present the

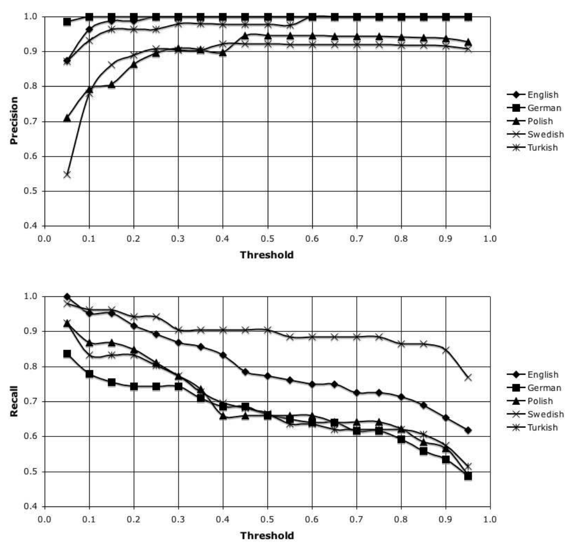


Figure 4: Phone detection results with three-level framework

Patterns (English, 245)	%	**Patterns** (German, 2290)	%
ddd-ddd-dddd	39.4	ddddd/dddddd	4.46
(ddd) ddd-dddd	35.7	ddddd-dddddd	3.92
d-ddd-ddd-dddd	6.55	ddddd dddddd	2.55
ddd.ddd.dddd	5.01	ddddd/ddddd	2.15
(ddd)ddd-dddd	2.83	dddd-ddddddd	1.82
ddd-dddd	2.25	dddd/ddddddd	1.78
ddd/ddd-dddd	0.80	dddd ddddddd	1.51
d.ddd.ddd.dddd	0.65	ddddddddddd	1.48
dddddddddd	0.63	ddddd-ddddd	1.20
ddd ddd dddd	0.56	ddddd / dddddd	1.08
Patterns (Polish, 691)	%	**Patterns** (Swedish, 502)	%
ddddddd	12.6	dddd-dd dd dd	18.8
ddd ddd dd dd	11.9	ddd-ddd dd dd	18.4
d-ddd ddd ddd	5.15	dd-ddd ddd dd	12.0
(ddd) ddd dd dd	4.50	dddd-dddddd	4.53
dddddddddd	4.33	dddddddddd	4.02
+dd dd ddd dd dd	3.35	ddd-ddddddd	3.88
ddd/ddddddd	3.18	dddd-ddd ddd	2.59
ddd dd dd	3.05	ddd - ddd dd dd	2.21
ddd-dd-dd	2.71	ddd-ddd dddd	1.95
d ddd ddd ddd	2.64	dd-ddd dd ddd	1.68

Patterns (Turkish, 344)	%
dddd ddd dd dd	20.3
d ddd ddd dd dd	9.08
(ddd) ddd dd dd	8.25
ddd dd dd	7.69
+dd ddd ddd dd dd	5.78
(dddd) ddd dd dd	3.88
dd-ddd-ddddddd	3.57
+dd (ddd) ddd dd dd	3.06
ddddddd	2.78
(ddd) ddd dddd	2.21

Table 5: Common phone patterns in each language, their frequencies and total number of unique patterns discovered

Language	Precision	Recall	F-measure
English	0.99	0.83	0.90
German	1.00	0.64	0.78
Polish	0.94	0.60	0.74
Swedish	0.90	0.83	0.86
Turkish	0.95	0.64	0.76

Table 6: Phone number detection results when only contextual features are used to support multi-language documents

Algorithm	Setting	Precision	Recall	F-measure
Baseline	0.01 and 0.01	1.00	0.10	0.19
Sim-based	0.01 and 0.01	1.00	0.13	0.28
Two-step	0.03 and 0.02	1.00	0.19	0.32
Baseline	0.03 and 0.15	0.76	0.23	0.35
Sim-based	0.02 and 0.15	0.70	0.45	0.55
Two-step	0.02 and 0.15	0.81	0.50	0.62
Algorithm	**Setting**	**Precision**	**Recall**	**F-measure**
Three-level (C-svm)	BoW	0.85	0.55	0.67
	Regexp	0.91	0.80	0.85
	BoW+Regexp	0.95	0.76	0.84

Table 7: Date and time detection results for the best precision, best f-measure scores, and three-level framework

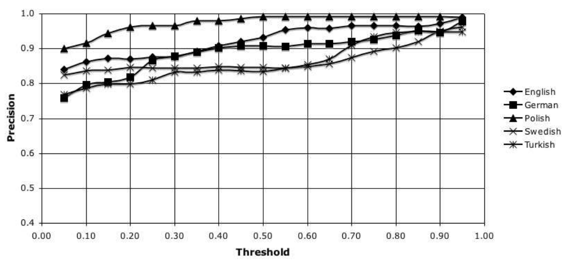
Lang	Algorithm	P1	R1	F1	P2	R2	F2
English	baseline	1.00	0.10	0.18	0.69	0.27	0.38
	sim-based	1.00	0.14	0.25	0.72	0.57	0.64
	2-step	1.00	0.23	0.37	0.76	0.64	0.69
	3-level (0.50)	0.93	0.76	0.84	0.93	0.76	0.84
	3-level (best)	0.99	0.61	0.76	0.84	0.88	0.86
German	baseline	0.82	0.12	0.20	0.51	0.23	0.32
	sim-based	1.00	0.05	0.10	0.47	0.38	0.42
	2-step	0.96	0.15	0.26	0.48	0.38	0.42
	3-level (0.50)	0.91	0.83	0.87	0.91	0.83	0.87
	3-level (best)	0.98	0.56	0.72	0.93	0.82	0.87
Polish	baseline	0.88	0.19	0.31	0.70	0.38	0.49
	sim-based	1.00	0.21	0.35	0.69	0.53	0.60
	2-step	1.00	0.31	0.48	0.69	0.52	0.59
	3-level (0.50)	0.99	0.86	0.92	0.99	0.86	0.92
	3-level (best)	0.99	0.86	0.92	0.96	0.93	0.95
Swedish	baseline	0.95	0.11	0.20	0.80	0.32	0.45
	sim-based	0.89	0.04	0.08	0.61	0.36	0.45
	2-step	0.94	0.17	0.29	0.64	0.42	0.50
	3-level (0.50)	0.85	0.87	0.86	0.85	0.87	0.86
	3-level (best)	0.96	0.53	0.69	0.85	0.90	0.87
Turkish	baseline	0.87	0.13	0.22	0.69	0.34	0.46
	sim-based	0.96	0.15	0.25	0.65	0.51	0.57
	2-step	0.84	0.23	0.36	0.67	0.54	0.60
	3-level (0.50)	0.84	0.90	0.87	0.84	0.90	0.87
	3-level (best)	0.95	0.84	0.89	0.95	0.89	0.92

Table 8: Date and time detection results in five languages

best precision (upper part) and best f-measure (middle part) values achieved by each algorithm. Next, we run experiments with C-svm model (lowest part): As expected and observed in Section 5.1, combined set of BoW and regular expressions performs best.

Experiments in Five Languages: The input for English (US) has already been shown in Table 2. For other languages, we obtain the translations and the seed patterns from our users. Although online dictionaries could be leveraged for assistance, this was not necessary as our users found the task trivial. We run the algorithms on the five collections with the best settings found above. In Table 8, we present the precision and recall values for the best precision and best f-measure settings. Similar to the results obtained in phone number detection, we see significant improvements with the proposed techniques. Recall the SVM model in the three-level framework produces a probability estimate. Next, we test with various threshold values and report the precision and recall values in Figure 5. The best values are also included in Table 8 to illustrate the potential improvements.

In Table 9, we report the boundary detection problems observed. The left part of the table shows over and under selection cases in terms of percentages, when the first level candidates are directly used in the final output. The right part of the table shows the results when the boundaries are adjusted based on the frequencies of the



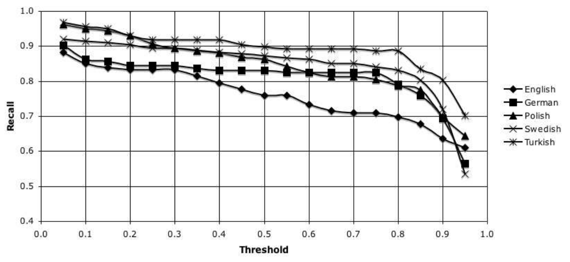


Figure 5: Date and time detection results with three-level framework using various threshold values

Language	Over (%)	Under (%)	Over (%)	Under (%)
English	3.25	0.00	1.62	0.00
German	1.56	0.00	0.00	0.00
Polish	3.62	1.45	2.90	1.45
Swedish	0.60	0.60	0.60	0.60
Turkish	2.11	0.70	0.70	1.40

Table 9: Selection problems when 1st level output is used (left), when boundaries adjusted based on pattern frequencies (right)

patterns, as described in Section 4.2. Based on our observations, the reasons for under selection cases include character problems (use of non-white space characters as separators), missing input from the users for the first level module, and not-optimal settings.

5.3 Statistical Significance Tests

We compute *SampleError = (FP+FN)/SampleSize* for the baseline and three-level algorithms using all the judgments in five languages, and compute the true error intervals at 99% confidence level. For phone number detection, baseline: 0.218 ± 0.033, and three-level: 0.048 ± 0.016. For date and time detection, baseline: 0.458 ± 0.033 and three-level: 0.139 ± 0.023.

We also perform a Cohen's Kappa test to see the agreement rate between the judgments and the two approaches. For the phone detection, the kappa scores and the agreement strengths are baseline: 0.341 *(fair)* and three-level: 0.855 *(very good)*. For date and time detection, baseline: 0.100 *(poor)* and three-level: 0.702 *(good)*.

6. PERFORMANCE

For some applications, the running time performance of the SSNE detector can be a crucial factor. For example, user-centric entity detection systems perform real-time detection and certain response time requirements apply. Web search engines process large collections of crawled web pages and even small improvements can result in big reductions in the overall processing time. Therefore, we review some implementation details here and present performance test results for phone number detection, as a case study.

We implement the proposed approach in C++ and use a linear SVM model based on [9]. The integration of the final detector, which is created by the three-level framework, is illustrated in Figure 6. The first level module in the framework is re-used and it

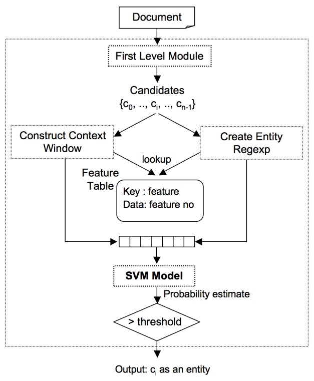


Figure 6: Implementation overview

Algorithm	Run Time	Processing Rate
regexp	3.813 sec	0.975 MB/sec
level1regexp	0.842 sec	4.417 MB/sec
svmModel	0.082 sec	45.158 MB/sec

Table 10: Running time performance results

generates the candidates. The module can run quite efficiently, as it simply finds sequences of digits in the string which may be separated by phone characters (Section 4.1). For these candidates, we first create features based on the terms found in the context window. Next, we replace the digits in the candidate with a special character and obtain a regular expression feature. We then make lookups into the feature table, and create an SVM vector to run with the model. Based on the probability estimate, we output the candidate as an entity. We call this approach as *svmModel*, and compare it to a production-quality regular expression, which is currently used in [31] and tested in Section 5.1. This is based on the PCRE library [20] and called via a C++ wrapper library. We call this solution as *regexp*, which can simply output phone entities found in a given document. However, one could argue that it does not have to run on the whole document, but only on the candidates generated by module. We test this solution as well and call it as *level1regexp*.

For the run time results, we perform the following experiment on a Linux machine with Dual Core AMD Opteron Processor 275 (1808 MHz) with enough main memory and 1MB cache. In the test, we used 1493 randomly chosen documents with an average size of 2.5KB. The corpus contained 152 phone numbers according to the *svmModel* detector. The total running time and processing rates of the above solutions are shown in Table 10. As we can see, the proposed approach is an order of magnitude faster than the production quality regular expression solution.

7. RELATED WORK

Much of the relevant work was already reviewed in the first three sections, so we briefly describe here the most closely related results and differences in our approach. The first category of related work involves bootstrapping algorithms, such as [1, 2, 19, 22], which have been successfully employed for the extraction of relations, semantic lexicons and named entities on large collections of text documents. In [17, 18], bootstrapping algorithms are run on web search engine query logs for the goal of extracting class attributes (such as *side effects* or *generic equivalent* for drugs). We are unaware of any previous studies that employed bootstrapping techniques for the recognition of semi-structured named entities. To provide a scalable solution for this problem, we propose a new framework and a two-step bootstrapping algorithm that can learn not only contextual rules but also regular expressions.

Another category of related work includes supervised learning techniques developed for the recognition of temporal expressions, including [11, 24]. In addition to our target entities (Section 4.2), these approaches focus on more complex expressions (e.g., "the same period a year ago", "Christmas Day"). To achieve this, they usually require a large training data. For example, above studies use the ACE2004 (automatic context extraction) corpus, which contains 767 documents and more than 8000 labeled temporal expressions. Obtaining such training data for many languages is a difficult task. Additionally, the final model depends on the nature and style of the training data. So if a small training set is used, we would not expect that model to perform well on web content as it is generated by many authors from different backgrounds. Therefore, we focus on the scalability aspect and propose a weakly-supervised technique as many languages need to be supported for the target entities, as specified by the users.

8. CONCLUSIONS

In this paper, we study the problem of detecting semi-structured named entities (such as phone numbers, date and time entities etc.) in text, and propose a scalable three-level bootstrapping framework to support many entity types, languages and regions. The framework identifies a large set of candidates in the first level by running a simple module, and employs a novel two-step bootstrapping algorithm in the second level to obtain an accurate set of positive examples. The algorithm can learn both regular expressions and contextual rules at every iteration. In the third level, the framework employs machine learning techniques and creates a model using the examples generated in the first two levels. This model, which generalizes the extraction rules further, can be used on new content for real time detection.

We evaluate the proposed techniques extensively on English, German, Polish, Swedish and Turkish documents for phone, date and time entities. Although the input required from the user is minimal, our approach can achieve 95% precision and 84% recall for phone entities, and 94% precision and 81% recall for date and time entities, on average. We also discuss some implementation issues and report our performance test results for phone number detection. The results show that the proposed approach is an order of magnitude faster than a production-quality regular expression based solution. In our future work, we are planning to incorporate active learning techniques to the framework and investigate capabilities for the extraction of other named entities and relations.

Acknowledgment

We are grateful to Vadim von Brzeski and Omid Rouhani-Kalleh for their help in the experimental evaluation. We would like to thank John Thrall and Kun Liu for their helpful suggestions.

9. REFERENCES

[1] E. Agichtein and L. Gravano. Snowball: extracting relations from large plain-text collections. In *Proc. of the 5th ACM Conf. on Digital Libraries*, 2000.

[2] S. Brin. Extracting patterns and relations from the world wide web. In *Proc. of the 1st Intl. Workshop on the Web and Databases*, 1998.

[3] O. Buyukkokten, J. Cho, H. Garcia-Molina, L. Gravano, and N. Shivakumar. Exploiting geographical location information of web pages. In *Proc. of the WebDB*, 1999.

[4] Y.-Y. Chen, T. Suel, and A. Markowetz. Efficient query processing in geographic web search engines. In *Proc. of the 2006 ACM SIGMOD Intl. Conf. on Management of Data*, 2006.

[5] T. Cheng and K. C.-C. Chang. Entity search engine: Towards agile best-effort information integration over the web. In *Proc. of the 3rd Conf. on Innovative Data Systems Research (CIDR)*, 2007.

[6] H. L. Chieu and H. T. Ng. Named entity recognition with a maximum entropy approach. In *Proc. of the 7th Conf. on Natural Language Learning*, 2003.

[7] M. Collins and Y. Singer. Unsupervised models for named entity classification. In *Proc. of the Joint SIGDAT Conf. on Empirical Methods in Natural Lang. Processing and Very Large Corpora*, 1999.

[8] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Unsupervised named-entity extraction from the web: an experimental study. *Artificial Intelligence*, 165(1):91–134, 2005.

[9] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[10] R. Farkas, G. Szarvas, S. Ivan, K. Andras, and R. Busa-Fekete. An iterative method for the de-identification of structured medical text. In *Proc. of AMIA I2B2NLP workshop*, 2006.

[11] K. Hacioglu, Y. Chen, and B. Douglas. Automatic time expression labeling for english and chinese text. In *Proc. of Conf. on Intelligent Text Processing and Computational Linguistics*, 2005.

[12] U. Irmak, V. von Brzeski, and R. Kraft. Contextual ranking of keywords using click data. In *Proc. of the 25th Intl. Conf. on Data Engineering (ICDE)*, 2009.

[13] A. Ittycheriah and S. Roukos. Ibm's statistical question answering system-trec 11. In *TREC*, 2002.

[14] D. Lin. Automatic retrieval and clustering of similar words. In *Proc. of the 17th Intl. Conf. on Computational Linguistics*, 1998.

[15] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[16] K. S. Mccurley. Geospatial mapping and navigation of the web. In *Proc. of the 10th Intl. Conf. on World Wide Web*, 2001.

[17] M. Paşca. Organizing and searching the world wide web of facts – step two: harnessing the wisdom of the crowds. In *Proc. of the 16th Intl. Conf. on World Wide Web*, 2007.

[18] M. Paşca, B. V. Durme, and N. Garera. The role of documents vs. queries in extracting class attributes from text. In *Proc. of the 16th ACM Conf. on Information and Knowledge Management*, 2007.

[19] M. Paşca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. Organizing and searching the world wide web of facts - step one: the one-million fact extraction challenge. In *Proc. of Natl. Conf. on AI*, 2006.

[20] Perl Compatible Regular Expressions. http://www.pcre.org.

[21] E. Riloff. Automatically constructing a dictionary for information extraction tasks. In *Proc. of Conf. on Artificial Intelligence*, 1993.

[22] E. Riloff and R. Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proc. of Conf. on Artificial Intelligence*, 1999.

[23] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[24] E. Saquete, O. Ferrández, S. Ferrández, P. Martínez-Barco, and R. Munoz. Combining automatic acquisition of knowledge with machine learning approaches for multilingual temporal recognition and normalization. *Inf. Sci.*, 178(17):3319–3332, 2008.

[25] E. Saquete, R. Munoz, and P. Martinez-Barco. Event ordering using terseo system. *Data Knowledge and Engineering*, 58(1):70–89, 2006.

[26] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Comput.*, 13(7):1443–1471, 2001.

[27] M. K. Stern. Dates and times in email messages. In *Proc. of the 9th Intl. Conf. on Intelligent User Interfaces*, 2004.

[28] M. Thelen and E. Riloff. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proc. of the 2002 Conf. on Empirical Methods in Natural Lang. Processing*, 2002.

[29] P. Turney. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proc. of European Conf. on Machine Learning*, 2001.

[30] O. Uzuner, Y. Luo, and P. Szolovits. Evaluating the state-of-the-art in automatic de-identification. *Journal of the American Medical Informatics Association*, 14, 2007.

[31] V. von Brzeski, U. Irmak, and R. Kraft. Leveraging context in user-centric entity detection systems. In *Proc. of the 16th ACM Conf. on information and knowledge management*, 2007.