Locating Complex Named Entities in Web Text

Doug Downey, Matthew Broadhead, and Oren Etzioni
Turing Center
Department of Computer Science and Engineering
University of Washington
Box 352350
Seattle, WA 98195, USA
{ddowney,hastur,etzioni}@cs.washington.edu

Abstract

Named Entity Recognition (NER) is the task of locating and classifying names in text. In previous work, NER was limited to a small number of predefined entity classes (*e.g.*, people, locations, and organizations). However, NER on the Web is a far more challenging problem. Complex names (*e.g.*, film or book titles) can be very difficult to pick out precisely from text. Further, the Web contains a wide variety of entity classes, which are not known in advance. Thus, hand-tagging examples of each entity class is impractical.

This paper investigates a novel approach to the first step in Web NER: locating complex named entities in Web text. Our key observation is that named entities can be viewed as a species of multi-word units, which can be detected by accumulating n-gram statistics over the Web corpus. We show that this statistical method's F1 score is 50% higher than that of supervised techniques including Conditional Random Fields (CRFs) and Conditional Markov Models (CMMs) when applied to complex names. The method also outperforms CMMs and CRFs by 117% on entity classes absent from the training data. Finally, our method outperforms a semi-supervised CRF by 73%.

1 Introduction

Named Entity Recognition (NER) is the task of identifying and classifying names in text. In previous work, NER was carried out over a small number of pre-defined classes of entities—for example, the named entity task definition from the *Message Understanding Conference* identified the three classes PERSON, ORGANIZATION, and LOCATION [Grishman and Sundheim, 1996]. However, the Web contains a variety of named entities that fall outside these categories, including products (*e.g.*, books, films), diseases, programming languages, nationalities, and events, to name a few.

NER requires first locating entity names in text. Some named entities are easy to detect in English because each word in the name is capitalized (*e.g.*, "United Kingdom") but others are far more difficult. Consider the two phrases:

(i) ...companies such as Intel and Microsoft...

(ii) ...companies such as Procter and Gamble...

The first phrase names two entities ("Intel" and "Microsoft"), whereas the second names a single entity ("Procter and Gamble"). However, this distinction is not readily apparent from the text itself. Similar situations occur in film titles ("Dumb and Dumber"), book titles ("Gone with the Wind"), and elsewhere (the "War of 1812").[1] The challenging problem is to precisely determine the *boundaries* of the entity names in the text. We refer to this problem as the entity *delimitation* problem. Since entity delimitation does not determine the entity's class, it is a subproblem whose solution is necessary but not sufficient for successful NER.

Another fundamental challenge for standard, supervised NER techniques is that the set of entity classes is not known in advance for many Web applications, including information extraction [Etzioni *et al.*, 2005], question answering [Banko, 2002], and search [Pasca, 2004]. Thus, it is impractical to hand tag elements of each entity class to train the supervised techniques. Instead, we are forced to create a training corpus where entities *of any type* are labeled "entity", and non-entities are labeled as such. This solution is problematic because NER techniques rely on orthographic and contextual features that can vary widely across entity classes. We refer to this problem as the *unseen classes* problem.

This paper introduces the LEX method, which addresses both the delimitation and unseen classes problems by utilizing n-gram statistics computed over a massive Web corpus. LEX does *not* address the entity classification problem. LEX is based on the observation that complex names tend to be *multi-word units* (MWUs), that is, sequences of words whose meaning cannot be determined by examining their constituent parts [da Silva *et al.*, 2004]; MWUs can be identified with high accuracy using *lexical statistics*, the frequency of words and phrases in a corpus [da Silva and Lopes, 1999].

[1] While uncapitalized words within entities tend to come from a small set of terms (e.g. "of," "and," etc.), it is not sufficient to simply ignore these terms as they often appear between distinct entities as well (as in "Microsoft and Intel" above).

LEX requires only a single threshold that is easily estimated using a small number of hand-tagged examples drawn from a few entity classes. However, LEX does require an untagged corpus that contains the relevant lexical items a handful of times in order to compute accurate lexical statistics. Thus, LEX is well-suited to the Web where entity classes are not known in advance, many entity names are complex, but a massive untagged corpus is readily available.

LEX is a semi-supervised learning method, so in addition to Conditional Random Fields (CRFs) [Lafferty *et al.*, 2001] and Conditional Markov Models (CMMs) [McCallum *et al.*, 2000], we compare it with CRFs augmented with untagged data using co-training and self-training. We find that LEX outperforms these semi-supervised methods by more than 70%.

Our contributions are as follows:

1. We introduce the insight that statistical techniques for finding MWUs can be employed to detect entity names of *any* class, even if the classes are unknown in advance.
2. We demonstrate experimentally that LEX, a surprisingly simple technique based on the above insight, substantially outperforms standard supervised and semi-supervised approaches on complex names rarely considered in NER such as the titles of films and books.[2]
3. We show preliminary results demonstrating that LEX can be leveraged to improve Web applications, reducing the error rate of a Web information extraction system by 60-70% in one case.
4. Based on an analysis of the limitations of LEX, we propose an enhanced algorithm (LEX++) which attaches common prefixes and suffixes to names, and uses additional lexical statistics to detect spurious collocations. LEX++ offers an additional 17% performance improvement over LEX.

Section 2 describes LEX. Section 3 presents our experimental results comparing LEX with previous supervised and semi-supervised methods. Section 4 presents an analysis LEX's limitations, along with an experimental evaluation of LEX++. Section 5 considers related work, and the paper concludes with a discussion of future work.

2 The LEX Method for Locating Names

The lexical statistics-based approach we use for locating names is based on the key intuition that names are a type of multi-word unit (MWU). Broadly, our algorithm assumes that contiguous capitalized words[3] are part of the same name, and that a mixed case phrase beginning and ending with capitalized words is a single name if and only if it is a MWU—that is, if it appears in the corpus sufficiently more frequently than chance. Capitalized words at the start of sentences are only considered names if they appear capitalized sufficiently often when *not* at the beginning of a sentence.

[2] All of the above techniques achieve close to 100% accuracy on simple names.

[3] Here, "capitalized," words are those beginning with a capital letter, and all other text (including punctuation and digits) is "uncapitalized."

More formally, the lexical method, denoted by LEX, is defined by two thresholds τ and δ, and a function $f(s_0, s_1, s_2)$ mapping three strings to a numerical value. The function $f(s_0, s_1, s_2)$ measures the degree to which the concatenated string $s_0 s_1 s_2$ occurs "more than chance" – several such collocation identifiers exist in the MWU literature, and we detail the two measures we experiment with below. The threshold τ is the value $f(s_0, s_1, s_2)$ must exceed in order for the string $s_0 s_1 s_2$ to be considered a single name. Lastly, if a capitalized word's appearances come at the beginning of a sentence more than δ of the time, we assume it is not a name. We heuristically set the value of δ to 0.5, and set τ using training data as described in Section 3.

Given a sentence S, a function $f(s_0, s_1, s_2)$, and thresholds τ and δ, LEX proceeds as follows:

1. Initialize a sequence of names $N = (n_0, n_1, \ldots, n_M)$ equal to the maximal contiguous substrings of S that consist entirely of capitalized words, where elements of N are ordered from left to right in S. If when the first word of S appears capitalized in the corpus, it is at the beginning of a sentence more than δ of the time, omit it from N.
2. Until all consecutive name pairs in N have been evaluated:
   (a) Choose the unevaluated pair of names (n_i, n_{i+1}) with minimum i.
   (b) If $f(n_i, w_i, n_{i+1}) > \tau$, where w_i is the uncapitalized phrase separating n_i and n_{i+1} in S, and w_i is at most three words in length,
   - replace n_i and n_{i+1} in N with the single name $n_i w_i n_{i+1}$.

2.1 Measures of Collocation

The function $f(s_0, s_1, s_2)$ is intended to measure whether a string $s_0 s_1 s_2$ occurring in text is a single name, or if instead s_0 and s_2 are distinct names separated by the string s_1. The functions f we consider are based on two standard collocation measures, Pointwise Mutual Information (PMI) and Symmetric Conditional Probability (SCP) [da Silva and Lopes, 1999]. Define the collocation function $c_k(a, b)$ for two strings a and b:

$$c_k(a, b) = \frac{p(ab)^k}{p(a)p(b)} \quad (1)$$

where $p(s)$ is the probability of the string s occurring in the corpus.

In the above, c_2 gives the SCP between a and b, and c_1 is (for our purposes) equivalent to PMI.[4] PMI and SCP are defined for only two arguments; a variety of techniques have been employed to extend these binary collocation measures to the n-ary case, for example by averaging over all binary partitions [da Silva and Lopes, 1999] or by computing the measure for only the split of highest probability [Schone and Jurafsky, 2001]. Here, we generalize the measures in the following simplistic fashion:

[4] More precisely, $PMI(a, b) = log_2 c_1(a, b)$. But in MWU identification we consider only the ordering of the collocation measure, so the monotonic *log* function can be ignored.

$$g_k(a,b,c) = \frac{p(abc)^k}{p(a)p(b)p(c)} \quad (2)$$

We take g_1 as our PMI function for three variables, and g_3 as our SCP function for three variables.

3 Experimental Results

This section describes our experiments on Web text, which compare LEX's performance with both supervised and semi-supervised learning methods. Experiments over the Web corpus can be tricky due to both scale and the diversity of entity classes encountered, so we begin by carefully describing the experimental methodology used to address these difficulties.

3.1 Experimental Methodology

Many classes of named entities are difficult to delimit precisely. For example, nested entities like the phrase "Director of Microsoft" could be considered a single entity (a job title), two entities (a job title and a company), or both. To make evaluation as objective as possible, we performed our experiments on entities from four classes where the boundaries of names are relatively unambiguous: (`Actor`, `Book`, `Company`, and `Film`).

The LEX method requires a massive corpus in order to accurately compute its n-gram statistics. To make our experiments tractable, we approximated this corpus with 183,726 sentences automatically selected as likely to contain named entities from the above four classes.[5]

We formed a training set containing a total of 200 sentences, evenly distributed among the four classes. In these training sets, we manually tagged all of the entities,[6] resulting in a total of 805 tagged examples. We formed a test set by hand-tagging a sample of 300 entities from each class; because the lexical approach requires estimates of the probabilities of different phrases, we ignored any entities that appeared fewer than five times in our corpus. To ensure that this was a reasonable restriction, we queried Google and found that over 98% of our original test entities appeared on at least 5 distinct Web pages. Thus, the requirement that entities appear at least five times in the corpus is a minor restriction. Given the massive size of the Web corpus, a lexical statistics method like LEX is likely to be effective for most entities.

We further separated our test set into "easy" and "difficult" cases. Precisely, the easy cases are those that are correctly identified by a baseline method that simply finds maximal substrings of contiguous capitalized words. The difficult cases are those on which this simple baseline fails. Our final test set contained 100 difficult cases, and 529 easy cases, suggesting that difficult entity names are quite common on the Web.

In the test phase, we ran each of our methods on the sentences from the test set. *Recall* is defined as the fraction of test set entities for which an algorithm correctly identifies both the left and right boundaries. *Precision* is defined as the fraction of putative entities suggested by the algorithm that are indeed bona fide entities. Because our test set contains only a proper subset of the entities contained in the test text, for the purpose of computing precision we ignore any entities identified by the algorithm that do not overlap with an element of the test set.

[5] We downloaded sentences from the Web matching patterns like "actors such as..." or "...and other films," *etc.*

[6] We also constructed training sets tagging *only* those entities of the target classes (Actor, Book, Company, and Film) – however, this approach decreased the performance of the supervised approaches.

	F1	Recall	Precision
LEX-PMI	0.38	0.43	0.34
LEX-SCP	**0.63 (66%)**	**0.66**	**0.59**

Table 1: **Performance of LEX using collocation measures PMI and SCP.** SCP outperforms PMI by 66% in F1.

3.2 PMI versus SCP

Our first experiment investigates which of the candidate collocation functions (PMI or SCP) is most effective for use in the LEX method. In these experiments and those that follow, the threshold τ is set to the value that maximizes performance over the training set, found using simple linear search.

The performance of each metric on difficult cases is shown in Table 1. The SCP metric outperforms PMI by a considerable margin. This performance difference is not merely a consequence of our method for acquiring thresholds; the performance discrepancy is essentially unchanged if each threshold is chosen to maximize performance on the test set. Further, although we are focusing on difficult cases, the SCP metric also outperforms PMI on the non-difficult cases as well (0.97 vs. 0.92 in F1).

The large performance difference between LEX-PMI and LEX-SCP is a more extreme version of similar results exhibited for those measures in MWU identification [Schone and Jurafsky, 2001]. In our case, PMI performs poorly on our task due to its well-known property of returning disproportionately low values for more frequent items [Manning and Schütze, 1999]. PMI fails to correctly join together several of the more frequent names in the test set. In fact, of the names that PMI fails to join together, 53% appear in the corpus more than 15 times – versus 0% for SCP.

In the remainder of our experiments the lexical method employs the SCP function and is referred to simply as LEX.

3.3 Comparison with NER approaches

We experimented with the following NER methods:

- **SVMCMM** – a supervised Conditional Markov Model trained with a probabilistic Support Vector Machine.
- **CRF** – a supervised Conditional Random Field approach.
- **CAPS** – a baseline method that locates all maximal contiguous substrings of capitalized words.
- **MAN** – an unsupervised name recognizer, based on manually-specified rules. This method is taken from the name location subcomponent of the KnowItAll Web information extraction system [Etzioni *et al.*, 2005].

Both Conditional Random Fields (CRF) [Lafferty *et al.*, 2001] and Conditional Markov Models (CMM) [McCallum *et al.*, 2000] are state of the art supervised approaches for

	F1	Recall	Precision
SVMCMM	0.42	0.48	0.37
CRF	0.35	0.42	0.31
MAN	0.18	0.22	0.16
LEX	**0.63 (50%)**	**0.66**	**0.59**

Table 2: **Performance on Difficult Cases** LEX's F1 score is 50% higher than the nearest competitor, SVMCMM.

	F1	Recall	Precision
SVMCMM	0.96	0.96	0.96
CRF	0.94	0.94	0.95
MAN	0.97	0.96	0.98
LEX	0.97	0.97	0.97
CAPS	**1.00 (3%)**	**1.00**	**1.00**

Table 3: **Performance on Easy Cases** All methods perform comparably near the perfect performance of the CAPS baseline; CAPS outperforms LEX and MAN by 3%.

text delimitation tasks. Our CRF and CMM implementations were taken from the MinorThird text classification package [Cohen, 2004]. We obtained settings for each algorithm via cross-validation on the training set. We chose to train the CMM using a Support Vector Machine (SVMCMM) rather than with the maximum entropy approach in [McCallum *et al.*, 2000] because the SVMCMM substantially outperformed the maximum-entropy classifier in cross-validation. The supervised classifiers create feature vectors for each word; these features include the lower-case version of the word, the orthographic pattern for the word (e.g. "Xx+" for "Bill" or "9+" for "1995", etc), and analogous features for tokens in a small window around the word (of size selected by cross validation).

By definition, the CAPS baseline performs perfectly on easy entity names, but does not correctly delimit any difficult entity names. CAPS is equivalent to Step 1 in the LEX algorithm in Section 2. Although LEX includes CAPS as an initial step to handle easy names, the two algorithms will differ on easy names when the lexical statistics, computed in Step 2 of LEX, indicate that two easy names delimited by CAPS ought to be concatenated. Consider, for example, the phrase "...as Intel and Microsoft have...". CAPS will always delimit this phrase's two names correctly, but LEX could potentially delimit them incorrectly (by, for example, outputting the phrase "Intel and Microsoft" as a single name). In practice, this situation occurs infrequently (see Table 3).

The performance of the different methods on difficult cases is shown in Table 2. Overall, LEX's F1 score of 0.63 is 50% higher than that of its nearest competitor (SVMCMM, at 0.42). The precision differences between LEX and SVMCMM are significant at $p < 0.01$, and the recall differences are significant at $p < 0.02$ (Fisher Exact Test). Of course, performance on the easy cases is also important. As shown in Table 3, all methods perform comparably well on easy cases, with F1 scores in the 0.95 and higher range.

The performance of LEX is fairly robust to parameter changes. A sensitivity analysis revealed that altering the learned value of τ by a factor of 2 in either direction resulted in LEX still outperforming SVMCMM by at least 40% on difficult cases. Also, altering the value of δ to either 0.4 or 0.6 resulted in LEX outperforming SVMCMM by at least 48% on difficult cases.

	F1	Recall	Precision
SVMCMM	0.29	0.34	0.25
CRF	0.25	0.31	0.21
MAN	0.18	0.22	0.16
LEX	**0.63 (117%)**	**0.66**	**0.60**

Table 4: **Performance on Unseen Entity Classes (Difficult Cases)** LEX outperforms its nearest competitor (SVMCMM) by 117%.

	F1	Recall	Precision
SVMCMM	0.93	0.92	0.94
CRF	0.94	0.93	0.95
MAN	0.97	0.96	0.98
LEX	0.95	0.95	0.95
CAPS	**1.00 (3%)**	**1.00**	**1.00**

Table 5: **Performance on Unseen Entity Classes (Easy Cases)** CAPS outperforms all methods by a small margin, performing 3% better than its nearest competitor (MAN).

3.4 Unseen Entity Classes

As argued in the introduction, in many Web applications, the target entity classes are not known in advance. This poses particular problems for standard NER approaches, which rely on textual cues that can vary widely among entity classes. In this experiment, we simulate the performance of each of our entity delimitation methods on unseen entity classes. Specifically, when evaluating performance on an entity class, we remove from the training set any sentences containing entities of that class. The results of these experiments on difficult cases are shown in Table 4. LEX outperforms the other approaches, with F1 performance more than 100% higher than the best competing method (SVMCMM). The precision and recall differences between LEX and SVMCMM on difficult cases are each statistically significant at $p < 0.001$ (Fisher Exact Test). As in the previous experiment, all methods perform relatively well on easy cases (Table 5).

3.5 Semi-supervised Comparison

LEX outperforms supervised NER methods by leveraging lexical statistics computed over a large, untagged corpus. Augmenting tagged examples with untagged data is known as *semi-supervised* learning, an increasingly popular approach for text processing tasks, in which untagged data is plentiful. In this section, we compare LEX with previous semi-supervised algorithms.

To create a semi-supervised baseline for our task, we augmented the supervised classifiers from our previous experiments to employ untagged data. As in other recent work on semi-supervised text segmentation [Ando and Zhang, 2005], we experimented with two semi-supervised approaches: co-training and self-training.

One of the first semi-supervised algorithms for named entity classification [Collins and Singer, 1999] was based on semi-supervised co-training [Blum and Mitchell, 1998]. Co-training relies on a partition of the feature set to produce independent "views" of each example to be classified, where it is assumed each individual view is sufficient to build a relatively effective classifier. In co-training, separate classifiers are trained for each view. Then each classifier is applied to the untagged examples, and the examples whose classifications are most certain are added as new training examples. By iteratively bootstrapping information from distinct views, co-training algorithms are able to create new training examples from untagged data.

The co-training approach developed in [Collins and Singer, 1999] does not actually perform entity delimitation. Instead, it assumes the entity delimitation problem is solved, and addresses the task of classifying a given set of entities into categories. In our attempts to utilize a similar co-training algorithm for our entity delimitation task, we found the algorithm's partitioning of the feature set to be ill suited to our task. The approach in [Collins and Singer, 1999] employed two views to classify named entities: the context surrounding a named entity, and the orthographic features of the entity itself. While the context surrounding a *known* entity is a good indicator of that entity's class, in our experiments we found that context was *not* effective for determining an unknown entity's boundaries. The co-training classifier trained on our training set using only the "context" view achieved F1 performance of 0.16 on difficult cases, and 0.37 on easy cases (versus 0.35 and 0.94 for the classifier trained on all features). This violates the central assumption in co-training, that each view be independently capable of producing a relatively effective classifier.

Because co-training was unsuited to our task, we experimented with "single-view" co-training, also known as *self-training*. This approach involved training a supervised classifier on the training set, and then iteratively applying the classifier to the untagged data, adding to the training set the r sentences on which the classifier is most confident.

Our experimental results are shown in Table 6. Because only the CRF classifier returned reliable probability estimates for sentences, it is the only classifier tested in a semi-supervised configuration. In these experiments, $r = 20$, and the algorithm was executed for a total of 40 self-training iterations. As shown in the table, the bootstrapped training examples make positive but insignificant impacts on the performance of the baseline supervised classifier. LEX's F1 performance of 0.63 is 73% higher than that of the best semi-supervised CRF in Table 6 (0.37).

3.6 Information Extraction Performance

One important application of Web named entity delimitation is found in unsupervised information extraction systems, e.g. [Etzioni *et al.*, 2005]. These systems use generic patterns (e.g. "films such as") to extract entities, but require some method to identify the boundaries of the entities to extract. Given a particular set of patterns, a more accurate name delimitation system should result in better extraction accuracy. In this experiment, we tested a pattern-based information extraction system on our evaluation corpus, using each of our methods to delimit entities. We ranked all extracted entities in order of decreasing frequency to obtain precision/recall curves for each class. Figure 1 shows the precision/recall curves for the 100 most frequent extractions of difficult cases in the `Film` class. As the graph shows, LEX enables significantly higher precision than the other methods (as before, all methods perform comparably well on easy extractions). Additional experiments are necessary to assess information extraction performance on other classes.

Self-tagged Sentences	F1	Recall	Precision
0	0.35	0.42	0.31
400	0.37	0.43	0.32
800	0.36	0.42	0.31

Table 6: **Performance of semi-supervised CRF (Difficult Cases)** The additional training sentences generated by the semi-supervised algorithm ("Self-tagged Sentences") do not significantly improve performance over the original 200 sentences from the training set.

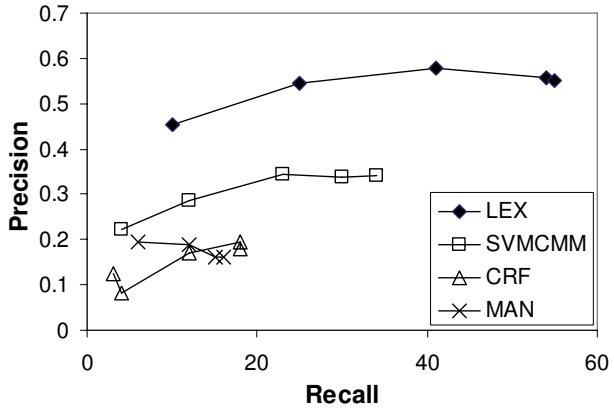

Figure 1: **Information Extraction performance for the `Film` class (Difficult Cases).** LEX's precision is 60-70% higher than its closest competitor (SVMCMM).

4 Error Analysis and Enhancements

LEX is a simple algorithm, and there are ways in which its simplistic assumptions about the structure of names hinder its performance. In this section, we detail three of LEX's primary error types, and we introduce LEX++, an enhanced algorithm which addresses LEX's limitations. LEX++ follows the LEX algorithm given in Section 2, with modifications for each of LEX's error types as detailed below.

Numbers and Punctuation at Name Boundaries

LEX's assumption that names begin and end with capitalized words is violated by any names that begin or end with numbers or punctuation—for example, the film "8 Mile" or a company name including trailing punctuation such as "Dell Inc." This limitation of LEX accounted for 38% of its false negatives on difficult cases in the experiments in Section 3.3.

LEX++ addresses this limitation of LEX by optionally extending names to the left or right (or both) to include numbers or punctuation. Formally, if a name n_i (output by LEX) is immediately preceded by a single number or punctuation mark t_{i-1}, LEX++ prepends t_{i-1} to n_i whenever $c_2(t_{i-1}, n_i)$ exceeds a threshold γ. Similarly, a number or punctuation mark t_i immediately following n_i is appended to n_i whenever $c_2(n_i, t_i) > \gamma$.

Common Prefixes and Suffixes

LEX's assumption that named entities correspond to lexical collocations does not always hold in practice. For example, LEX rarely attaches common prefixes and suffixes to names. In an entity like "Cisco Systems, Inc." the statistical association between the suffix "Inc" and the company name "Cisco Systems" is relatively small, because "Inc" occurs so frequently in the corpus as a part of other entity names. The situation is further complicated by the fact that most prefixes and suffixes should be considered part of the entity name (as in the "Inc." example above), but others should not (e.g., honorifics like "Sir" or "Mr.").[7] Mis-applying prefixes and suffixes is responsible for 27% of LEX's false negatives.

LEX++ attempts to attach common prefixes and suffixes to names. To determine whether a given phrase is a prefix or a suffix, we would like to measure how frequently the phrase appears as a constituent of other entity names. As an approximation to this quantity, we use the fraction of times the phrase is followed or preceded by a capitalized word. Formally, we consider a phrase $r = n_i w_i n_{i+1}$ (where n_i and n_{i+1} are distinct names output by LEX) to be a single name including a prefix $n_i w_i$ if the phrase $n_i w_i$ is followed by a capitalized word in the corpus more than a threshold ϕ of the time. Similarly, r is a single name including a suffix if $w_i n_{i+1}$ is preceded by a capitalized word more than ϕ of the time.[8]

Related Entities

LEX also makes errors due to the high statistical association between distinct but related entities. Phrases containing multiple related entities (e.g., conjunctions like "Intel and AMD") occur frequently relative to their constituent parts, and therefore appear to LEX to be MWUs. Thus, these phrases are often erroneously output as single names by LEX.

LEX++ attempts to identify phrases containing distinct names which happen to be mentioned together frequently. Specifically, LEX++ exploits the fact a named entity requires a unique *order* for its constituent terms, whereas a phrase containing distinct entities often does not. For example, consider that the phrase "Intel and AMD" appears with similar frequency in its reversed form ("AMD and Intel"), whereas the same is not true for the single entity "Pride and Prejudice."

Formally, consider a phrase $r = n_i w_i n_{i+1}$ which would be concatenated into a single name by LEX, that is, $g_3(n_i, w_i, n_{i+1}) > \tau$. LEX++ outputs r as a single name only if $p(n_{i+1} w_i n_i)$, the probability of the reverse of r, is sufficiently small. In our experiments, we use a heuristic of outputting r as a single name only when the reverse of r appears at most once in our corpus.

[7] These guidelines are taken from the MUC standard for named entity recognition, which we follow where applicable.

[8] We also limit names n_k within prefixes and suffixes to be fewer than three words in length.

	F1	Recall	Precision
LEX	0.63	0.66	0.59
LEX++	**0.74 (+17%)**	**0.76**	**0.72**

Table 7: **Performance of LEX++ versus LEX (Difficult Cases).** LEX++ outperforms LEX by 17%.

4.1 Experiments with LEX++

We compared the performance of LEX++ to LEX under the experimental configuration given in Section 3.3. The values of τ and δ were taken to be the same for both algorithms, and the thresholds ϕ and γ were set using linear search over the training set.

The results of our experiments are shown in Table 7. For difficult entity names, the enhancements in LEX++ offer a 17% improvement in F1 over the original LEX. The performance of both algorithms on easy cases is similar—LEX++ has a slightly lower F1 score (0.96) than that of LEX (0.97).

5 Related Work

The majority of named entity recognition (NER) research has focused on identifying and classifying entities for a small number of pre-defined entity classes, in which many entities can be identified using capitalization. Our focus, by contrast, is on delimiting names in cases where entity classes may be unknown, and when names are complex (involving a mixture of case).

Several supervised learning methods have been used previously for NER, including Hidden Markov Models, and more recently Maximum Entropy Markov Models [McCallum *et al.*, 2000] and Conditional Random Field (CRF) models [Lafferty *et al.*, 2001; McCallum and Li, 2003]. Our experiments show that LEX outperforms these methods on our task. Semi-supervised methods for NER, which leverage untagged data, were investigated specifically in the NER component of the CoNLL 2003 shared task [Tjong Kim Sang and De Meulder, 2003]. While this task was primarily aimed at typical NER categories (PERSON, LOCATION, and ORGANIZATION), it also encompassed other "miscellaneous" entities, including nationalities and events. The evaluation corpus consisted of high quality newswire text, and offered a relatively small untagged corpus (roughly six times the size of the tagged corpus). Our focus is on leveraging the Web, a vastly larger (and lower quality) untagged corpus. Further, our experiments show that LEX outperforms canonical semi-supervised methods on our task. These two distinctions are likely responsible for the fact that none of the top performing systems in the CoNLL task made profitable use of untagged data, in sharp contrast to our experiments in which LEX outperforms supervised methods by making crucial use of untagged data.

Our experiments compared LEX with previous semi-supervised methods for NER, including co-training [Collins and Singer, 1999; Blum and Mitchell, 1998] and single-view bootstrapping. In recent work, Li and McCallum used untagged data to devise context-based word clusters, which

were added as features to a supervised CRF algorithm. The untagged data was shown to improve performance in part of speech tagging and Chinese word segmentation [Li and McCallum, 2005]. Also, recently Ando and Zhang developed a semi-supervised method for text chunking, based on training thousands of classifiers on untagged data, and using the commonalities of the classifiers to devise relevant features for a particular task [Ando and Zhang, 2005]. This method was shown to improve performance in named entity delimitation on the CoNLL 2003 corpus mentioned above. In contrast to LEX, which is simple, the methods in [Li and McCallum, 2005] and [Ando and Zhang, 2005] are complex. Further, neither of these methods have been evaluated on tasks such as ours, where entity names are difficult and the target entity classes may not be known in advance. Comparing LEX with these semi-supervised techniques is an item of future work.

Lastly, the basic insight used in LEX – that named entities are a type of MWU – was also employed in [da Silva *et al.*, 2004] for an unsupervised entity extraction and clustering task. In contrast to that work, LEX is a technique for semi-supervised named entity delimitation, and we provide experiments demonstrating its efficacy over previous work.

6 Conclusions

This paper shows that LEX, a simple technique employing capitalization cues and lexical statistics, is capable of locating names of a variety of classes in Web text. Our experiments showed that this simple technique outperforms standard supervised and semi-supervised approaches for named entity recognition in cases where entity names are complex, and entity classes are not known in advance.

Google's recent release to the research community of a trillion-word n-gram model makes LEX practical at Web scale. Evaluating LEX in a large-scale experiment is an important item of future work. Further, although the statistical approach performed well in our experiments, it remains the case that textual features are valuable clues in identifying names—investigating ways to utilize both lexical statistics and textual features simultaneously is a natural next step. Finally, further investigation into leveraging improved Web entity location to enhance applications such as Web search, information extraction, and question answering is an important item of future work.

Acknowledgments

We thank Michele Banko, Charles Downey, Jonathan Pool, and Stephen Soderland for helpful comments. This research was supported in part by NSF grants IIS-0535284 and IIS-0312988, DARPA contract NBCHD030010, ONR grant N00014-02-1-0324 as well as gifts from Google, and carried out at the University of Washington's Turing Center. The first author was supported by a Microsoft Research Fellowship sponsored by Microsoft Live Labs.

References

[Ando and Zhang, 2005] R Ando and T Zhang. A high-performance semi-supervised learning method for text chunking. In *Proc. of ACL*, 2005.

[Banko, 2002] M. Banko. Askmsr: Question answering using the worldwide web. In *Proc. of EMNLP*, 2002.

[Blum and Mitchell, 1998] A. Blum and T. Mitchell. Combining Labeled and Unlabeled Data with Co-Training. In *Proc. of COLT*, 1998.

[Cohen, 2004] W. W. Cohen. Minorthird: Methods for identifying names and ontological relations in text using heuristics for inducing regularities from data, http://minorthird.sourceforge.net, 2004.

[Collins and Singer, 1999] M. Collins and Y. Singer. Unsupervised Models for Named Entity Classification. In *Proc. of EMNLP/VLC*, 1999.

[da Silva and Lopes, 1999] J. Ferreira da Silva and G. P. Lopes. A local maxima method and a fair dispersion normalization for extracting multi-word units from corpora. In *Sixth Meeting on Mathematics of Language*, 1999.

[da Silva *et al.*, 2004] J. Ferreira da Silva, Z. Kozareva, V. Noncheva, and G. P. Lopes. Extracting named entities. a statistical approach. In *TALN 2004*, 2004.

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

[Grishman and Sundheim, 1996] R. Grishman and B. Sundheim. Message understanding conference- 6: A brief history. In *Proc. of COLING*, 1996.

[Lafferty *et al.*, 2001] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*, 2001.

[Li and McCallum, 2005] W Li and A McCallum. Semi-supervised sequence modeling with syntactic topic models. In *Proc. of AAAI*, 2005.

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

[McCallum and Li, 2003] A. McCallum and W. Li. Early results for named entity recognition with conditional random fields. In *Proc. of CoNLL*, 2003.

[McCallum *et al.*, 2000] A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proc. of ICML*, 2000.

[Pasca, 2004] M. Pasca. Acquisition of categorized named entities for web search. In *Proc. of CIKM*, 2004.

[Schone and Jurafsky, 2001] P. Schone and D. Jurafsky. Is knowledge-free induction of multiword unit dictionary headwords a solved problem? In *Proc. of EMNLP*, 2001.

[Tjong Kim Sang and De Meulder, 2003] E. Tjong Kim Sang and F. De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proc. of CoNLL*, 2003.

Named Entities are named nouns which fall into the catagories following: People, Locations, and Organizations. The strategies explored are useing a text normalizer to shape the text into a format that NER programs can recognize and cross checking classiers to increase the precision of NER tools. I. INTRODUCTION Named Entity Recognition is widely used in many different kinds of natural language processing tasks.Â [4] D. Downey, M. Broadhead, O. Etzioni. 2007. Locating Complex Named Entities in Web Text. In Procs. of IJCAI 2007. [5] O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, A. Yates. 2005. Unsupervised Named-Enitiy Exs-traction from the Web: An Expirimental Study. Articial Intelligence, 165(1):91134, 2005. Named Entity Recognition (NER) is the task of locating and classifying names in text. In previous work, NER was limited to a small number of pre-dened entity classes (e.g., people, locations, and organizations). However, NER on the Web is a far more challenging problem. Complex names (e.g., lm or book titles) can be very difcult to pick out precisely from text. Further, the Web contains a wide variety of entity classes, which are not known in advance.Â This paper investigates a novel approach to the rst step in Web NER: locating complex named entities in Web text. Our key observation is that named entities can be viewed as a species of multi-word units, which can be detected by accumulat-ing n-gram statistics over the Web corpus. Well, mapped entities in EF basically represent database tables. If you project onto a mapped entity, what you basically do is partially load an entity, which is not a valid state. EF won't have any clue how to e.g. handle an update of such an entity in the future (the default behaviour would be probably overwriting the non-loaded fields with nulls or whatever you'll have in your object).Â If you are using Entity framework, then try removing property from DbContext which uses your complex model as Entity I had same problem when mapping multiple model into a viewmodel named Entity. public DbSet Entities { get; set; } Removing the entry from DbContext fixed my error. @inproceedings{Downey2007LocatingCN, title={Locating Complex Named Entities in Web Text}, author={Doug Downey and M. Broadhead and Oren Etzioni}, booktitle={IJCAI}, year={2007} }. Doug Downey, M. Broadhead, Oren Etzioni. Published in IJCAI 2007. Computer Science. Named Entity Recognition (NER) is the task of locating and classifying names in text. In previous work, NER was limited to a small number of pre-defined entity classes (e.g., people, locations, and organizations). However, NER on the Web is a far more challenging problem. Complex names (e.g., film or book titles) can be very difficult