Combining n-gram based statistics with traditional methods for named entity recognition

Ian Fette ICF@CS.CMU.EDU
School of Computer Science, Carnegie Mellon University
5000 Forbes Ave, Pittsburgh PA 15217

Abstract

In this paper, we show three main results. First, we show that an n-gram dataset built from a large web crawl, as opposed to data from the specific target domain, can be used to perform the task of named entity recognition with reasonable accuracy. Second, we show that for complex domains, such as the MUC-7 NER task, the Lex method may not perform as well as other methods, due largely in part to subjective definitions of what constitutes a named entity. Finally, we show that for both clear domains and more subjective domains, combining Lex with traditional lexical approaches can achieve higher accuracy than either alone.

1. Introduction

Named Entity Recognition (NER) is an important step in many Information Extraction pipelines. Over the past years, many papers have been written on the subject, ranging from approaches using purely lexical features (Bikel et al., 1999), to approaches including human knowledge in the form of dictionaries (Borthwick et al., 1998), to boosted wrapper induction (Freitag & Kushmerick, 2000), to purely statistical approaches based on word co-occurrence methods (Downey et al., 2007). The latter approach represents some of the most recent research on NER, and is made even more interesting by the recent release of a very large n-gram dataset from Google.

In this paper, we have three primary objectives. First, we wish to see whether this new dataset from Google can be used to increase accuracy in NER tasks. Second, we wish to examine the accuracy of the Lex method presented in (Downey et al., 2007) on a standard (and more difficult) dataset. Third, we wish to see if it is possible to combine the lexical features favored in previous approaches with these more recent n-gram based approaches.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

2. Background

2.1. NER Methods

The problem of *named entity recognition* requires identification of the names of companies, places, people, etc. in text. There are many approaches to this problem involving supervised learning. However, as shown in Minkov et al. (Minkov et al., 2005), adapting supervised approaches to named entity recognition to domains which are not as clean as newswire text can take considerable effort and knowledge of the domain. (In (Minkov et al., 2005), the authors are looking at emails, the language of which is less formal and the text less grammatical than the "clean" newswire text).

Recently, more purely statistical approaches have been explored which are more robust to dealing with noisy text. Silva et al. first noted that named entities tend to appear as Multiword Lexical Units (MWUs)– strings of words that occur together much more than they would by chance (Silva et al., 2004). In this paper, Silva et al. automatically extract these MWUs, and then apply a method to select possible named entities therefrom. To select candidate MWUs, neighboring words are considered as a possible n-gram, and the final result is the n-gram with highest "cohesion", that is, the probability of seeing that n-gram divided by the probability of seeing its constitutent parts alone. The size of the n-gram being considered is increased, stopping when the cohesion of the n-gram is larger than the cohesion of both the (n-1) grams that are part of the n-gram, as well as the (n+1)-grams containing the n-gram. The authors discuss methods for filtering out results, including throwing out MWUs that do not both start and end with a capitalized word.

Downey et al. used this insight to identify named entities in web text (Downey et al., 2007). Their work noted that web text is very noisy compared to other text sources, making sophisticated Natural Language Processing approaches that depend on deep linguistic analysis difficult. Instead, they used the MWU insight combined with a corpus of nine-million web pages to identify named entities. This results in impressive performance at the task of finding complex named entities– entities such as *Proctor and Gamble* that are difficult to distinguish from cases like *Microsoft and Intel* where two separate entities occur with a conjunction between them. The authors do note, however, that the Google n-gram data offers an opportunity to achieve even better performance due to the much larger data set. That insight was one of the motivating factors for this paper.

A similar approach is used by Pantel and Lin (Pantel & Lin, 2001) for finding terms in a corpus. Terms are indivisible units of language, where their meaning cannot be recovered if broken up. Examples include "machine gun", "Proctor and Gamble", "machine learning", and "hard drive". This work uses a statistical approach to find words that co-occur much more often than chance. In this paper, the authors start with pairs of words, and then expand these pairs to larger multi-word terms. The authors define a "mutual information" metric similar to the "cohesion" metric of (Silva et al., 2004) and co-location probability of (Downey et al., 2007) for 2-grams, but also define a second log-likelihood metric for use in the case of low frequency counts. Given the huge dataset we're working with, it is not clear whether we will have the problem of low frequency counts.

2.2. Datasets

In this work, we re-implement the Lex method described in (Downey et al., 2007), but rather than using a random selection of a few hundred thousand sentences, we use a much larger dataset computed by doing a crawl of (a significant portion of) the web. We also use a standard dataset for testing the accuracy of our approaches to NER. This has the advantage of making it easier to directly compare our results to other works, as well as removing some of the bias potentially introduced by creating one's own train and test set for NER, which in many cases involves rather subjective decisions.

2.2.1. Statistical Dataset

The Lex method requires a large statistical dataset containing counts of the number of times word phrases (n-grams) occur. Downey et al. counted word frequency in a few hundred thousand sentences downloaded from the web. We take advantage of a newer, much larger dataset. We refer to this data as "The Google n-gram data" (Brants & Franz, 2006), which was released by Google Research in 2006. The data collection gives the number of times an ordered sequence of tokens occurs in a collection of public web pages containing over one trillion words. Sequences of lengths one, two, three, four, and five are reported. Words that occurred fewer than 200 times were replaced by the token '<UNK>', and n-grams that occurred fewer than 40 times were not reported. Punctuation and capitalization are preserved in the data set. This dataset is 87GiB on disk, without any indexes. To illustrate what this data looks like, we have included a small section of a view that maps the 3gram table to its character representation below.

Table 1. Content of table 3grams

word1	word2	word3	frequency
Hello	world	by	86
Hello	world	does	65
Hello	world	example	445
Hello	world	fails	62
Hello	world	for	90

One of the motivations stated by Google Research in releasing this dataset was to allow researchers access to information that would be impractical to assemble without large-scale computing resources. They sought to level the playing field in some sense to allow researchers with modest computing resources to perform research with large volumes of web data. We hope to find that this dataset does in fact make such research viable.

2.2.2. Tagged data

We need some dataset on which to train our NER system, as well as a dataset on which to test its accuracy. The MUC-7 (MUC-7, 1998) data provides an ideal solution to both of these problems. Prepared for the 7th Message Understanding Conference, the MUC-7 data contains hundreds of articles from the New York Times that have been annotated with markers indicating Persons, Locations, Organizations, and Places, among other things. An added advantage to using this data is that many other systems have been tested on this data, and as such it is easier to directly compare accuracy of our endeavors with other systems.

Alternately, we also have access to the data that Downey et al. used to train and test their system, as reported in (Downey et al., 2007). This dataset has its own benefits and drawbacks. On the plus side, Downey et al. used classes that were far less ambiguous than the phrases appearing in MUC-7 (e.g. names of books, actors, films, and companies). This removes much of the subjectiveness from the task of labeling named entities. On the other hand, their test method (and data) is somewhat non-standard, and as such we still cannot directly compare results. In the test data, only one named entity is tagged per sentence, irrespective of the number of acutal named entities (i.e. it might be the case that three actors are mentioned in a sentence, but only one would be tagged). In personal correspondence, Downey indicated that this was because they were "evaluating the system only when it overlapped with a test entity". This has the effect of producing test sentences like the following:

> Filmed in Scotland , Shoebox Zoo boasts big name actors such as Peter Mullan , Jason Connery and <E>Alan Cumming< /E> , and introduces talented young Canadian actress Vivien Endicott-Douglas in the lead role of Marnie McBride .

In the above sentence, there are clearly multiple named entities (Scotland, Shoebox Zoo, etc.) yet only one is tagged. Thus, it is difficult to use this test data to evaluate accuracy. If the system tags Scotland as a named entitiy, according to Downey that would be ignored because it does not overlap with a test entity. This clearly affects the precision that will be reported, as well as the recall (i.e. if Filmed is tagged as a named entitiy, should it be ignored as it does not overlap?)

Given these oddities with the test data used by Downey et al. we use only Downey's training data, which does label all named entities in a sentence. This still leaves us with approximately 200 sentneces, which is enough to perform cross validation on.

3. Methods

We used three distinct methods for named entity recognition on the data mentioned in Section 2.2.2. The first method was a re-implementation of Lex (with slight variations), which is described in detail in Section 3.1. The second was a SVM-CMM, described in Section 3.2. The third approach was a hybrid approach combining the two, described in Section 3.3.

3.1. Lex re-implementation

We implemented a variant of Lex as a feature extractor within MinorThird (Cohen, 2004), a Java package for, among other things, classifying and performing extraction tasks on text. This allowed us to to implement the method rather quickly, without having to worry about much of the support code required in text extraction and results evaluation. Our implementation follows that of (Downey et al., 2007) rather closely, but there are a few key differences. The overall method is described in Section 3.1.1, with the key differences pointed out in Section 3.1.2.

3.1.1. Details of Lex

The overall Lex method boils down to first extracting all sequences of capitalized words to create an initial list of candidate named entities. Candidates occurring within three words of each other are then evaluated to see if they are a single named entity (e.g. "Proctor and Gamble") or two separate named entities (e.g. "Intel and Microsoft"). This is done by querying the database to see if the single phrase formed by combining the two candidate named entities occurs more frequently than would be predicted if the two entities were assumed to be independent. More precisely, if you have two candidates c_1 and c_2 separated by some short "glue" phrase g, then the goal would be to calculate

$$\frac{p(c_1 g c_2)^3}{p(c_1)p(g)p(c_2)}$$

These probabilities correspond to counts from the n-gram tables described in Section 2.2.1. Continuing with the "Proctor and Gamble" example, we would perform a query to determine the number of times the 3-gram "Proctor and Gamble" occurs. We would then perform three queries to see how many times the individual words "Proctor", "and", and "Gamble" occur, and by normalizing these counts we can come up with an estimate of the probability of seeing these words on the web, and use this to determine whether these words are independent (separate named entities) or dependent (a single named entitiy).

This can become more complicated when the three sets to be merged (the two upper-case MWUs and intervening text) are longer than five tokens. Take, for instance, the phrase "United Nations Educational, Scientific and Cultural Organization". This is actually eight tokens (the comma is considered a separate token). After two steps, we will need to evaluate whether to merge $<$ { United Nations Educational , Scientific }, { and }, { Cultural Organization } $>$ into one named entity. Given that we only have data up to 5-grams, we need to somehow stitch results together to allow us to evaluate longer named entities. We use the following algorithm to evaluate whether to merge the phrases a, b and c.

```
1: if $size(b) > 3$ then
2:     return;
3: end if
4: $abc \leftarrow a + b + c$
5: $LeftIndex \leftarrow a.size()$
6: $RightIndex \leftarrow a.size() + b.size() - 1$
7: while $(RightIndex - LeftIndex + 1) < 5$ do
8:     if $LeftIndex > 0$ then
9:         $LeftIndex - -$
10:    end if
11:    if $((RightIndex - LeftIndex + 1) < 5) \&\& RightIndex < (abc.size() - 1)$ then
12:        RightIndex++
13:    end if
14: end while
15: $numABC \leftarrow queryNgram(abc[LeftIndex : RightIndex])$
16: $sizeA \leftarrow a.size() - LeftIndex$
17: $sizeB \leftarrow b.size()$
18: $sizeC \leftarrow RightIndex - (a.size() + b.size() - 1)$
19: $numA \leftarrow queryNgram(abc[LeftIndex : LeftIndex + sizeA - 1])$
20: $numB \leftarrow queryNgram(abc[LeftIndex + sizeA : RightIndex - sizeC])$
21: $numC \leftarrow queryNgram(abc[RightIndex - sizeC + 1 : RightIndex])$
22: $probA \leftarrow numA/totalNgrams(sizeA)$
23: $probB \leftarrow numB/totalNgrams(sizeB)$
24: $probC \leftarrow numC/totalNgrams(sizeC)$
25: $probABC \leftarrow numABC/totalNgrams(sizeA + sizeB + sizeC)$
26: $probTogether \leftarrow \frac{ProbABC^3}{probA * probB * probC}$
27: if $probTogether > \alpha$ then
28:     merge(a,b,c)
29: end if
```

Thus, as a concrete example, let us again consider the case of $<$ { United Nations Educational , Scientific }, { and }, { Cultural Organization } $>$. The algorithm will set "abc" equal to "United Nations Educational , Scientific and Cultural Organization", LeftIndex will be 3, and RightIndex will be 7. Essentially, we will wind up calculating

$$\frac{Prob(\text{, Scientific and Cultural Organization})^3}{Prob(\text{, Scientific})Prob(\text{and})Prob(\text{Cultural Organization})}$$

If this is greater than a threshold learned from a training set, we will merge the parts into one named entity (and hopefully correctly recognize that UNESCO is, in fact, a single organization). This process of considering merging MWUs is repeated for each neighboring pair of MWUs within a sentence, until no further mergers exceed the threshold.

3.1.2. Key differences in Lex implementation

The key difference between our version of Lex, and the original version used by Downey et al., arises from our handling of phrases of length greater than the size of n-grams available (in our implementation, the maximum size is 5-grams, whereas the original version used 4-grams). Downey et al. attempted to estimate the probability of a long phrase by chaining the conditional probabilities to the maximum extent possible, e.g. given a 5-gram model, the probability of ABCDEFG would be estimated as $P(ABCDE) * P(F|BCDE) * P(G|CDEF)$.

In our implementation, we would not try to estimate the probability of the entire phrase ABCDEFG, but would rather simply estimate the probability of BCDEF (i.e. the uncapitalized "glue" in the middle, plus as much of the candidate NERs to either side as will fit in our 5-gram model). It is unclear exactly how much of an effect this difference has, although given more time, it would be a relatively simple exercise to precisely measure the effect.

3.2. SVM-CMM

Our implementation of SVM-CMM is simply the default implementation from MinorThird (Cohen, 2004). In this model, a set of lexical features are extracted from each word. These features include the word itself, as well as lexical properties of the word such as character patterns. A SVM is then trained based on this feature representation, this SVM then outputs labels indicating whether or not the word is a candidate named entity fragment. A CMM is then trained on the output of the SVM classifier, which then actually marks spans of text as opposed to simply the individual tokens.

3.3. Hybrid approach

Since we are using a package like MinorThird, and have essentially implemented Lex as a "feature", it seems very natural to consider combining Lex with other approaches. In the case of Lex alone, we essentially have a classifier trained on a single feature - whether Lex says a token is part of a named entity or not. We have much more context available to us, however. As we show later in Section 5, a model trained on purely lexical features, such as the SVM-CMM, is very competitive. In theory, features such as the word itself as well as the context of surrounding words, ought to help create a more accurate classifier. As such, we also look at a hybrid approach, in which we again use a SVM-CMM, but add the output of Lex as an addi-

Table 2. Lex results on MUC-7 data, dry-run test data

Span/Token	Precision	Recall	F-1
Token	.6979	.9621	.8089
Span	.5596	.8220	.6659

Table 3. SVM-CMM results on MUC-7 data, dry-run test data

Span/Token	Precision	Recall	F-1
Token	.9266	.9334	.9300
Span	.8749	.8689	.8719

tional feature, complimenting the existing features of the SVM-CMM described in the previous section.

4. Experiments

We ran experiments to test the accuracy of all three methods mentioned in Section 3 on both the MUC-7 data, and Downey's hand tagged training data, as described in Section 2.2.2. For the MUC-7 data, we show the results on the "dry-run" dataset (which was on the same topic as the training data), and on the "formal test" dataset, which was on a slightly different topic than the training data. For Downey's hand tagged data, we run 10-fold cross validation experiments for Lex, SVM-CMM, and the hybrid approach (again using the training data only, for reasons described in Section 2.2.2).

5. Results

We first examine the accuracy of Lex, SVM-CMM, and the hybrid approach on the dry-run test set that was supplied as part of the MUC-7 contest, as discussed in Section 2.2.2. As noted earlier, this data comes from the same domain as the training dataset. We then compare accuracy on the formal test set, which came from a related but not exactly the same domain.

Table 2 shows the accuracy of the Lex method using the Google n-gram dataset on the MUC-7 dry-run test data, evaluated at both a span-level and a token-level. Span accuracy is a measure of whether the entire named entity was classified correctly, whereas token accuracy is a measure of whether the tokens comprising the named entity were correctly classified. That is, given a named entity like "Proctor and Gamble", it is possible to identify the tokens "Proctor" and "Gamble" as being two separate named entities, while not identifying the token "and" as being part of a named entity. This assignment of named entity boundaries would get the span wrong (the span either exactly matches or it is wrong), while two of the three constituent tokens would be marked as correctly classified.

The meaning of the columns in Table 2 is as follows. Precision is a measure of the precentage of labels that are correct. As such, the interpretation of the data shown is that of all the tokens being marked as part of a named entity, only 69.79% actually belong inside of a real named entity. Recall is a measure of the percentage of labels in the document that are identified. From Table 5, it can be seen that 96.21% of the tokens in the document that are actually part of a named entity are marked as such by our NER code. F-1 measure is a combined measure of precision and recall, and is equal to $\frac{2*Precision*Recall}{Precision+Recall}$ Table 3 shows the accuracy of the SVM-CMM classifer on the same data.

The results indicate that Lex is getting reasonable coverage of the named entities (token recall of 95%), but that it's also labeling extra words as being part of a named entity when in fact they are not (token precision of 53.28% indicates that approximately half of the tokens we identify as being part of named entities are in fact not.) At the same time, the SVM-CMM has significantly lower recall, but much higher precision.

We examined the test data to try and determine what sorts of mistakes were causing the lower than desirable precision when using Lex. In short, the boundaries of the test cases are much more complex than this method was designed to handle. In (Downey et al., 2007), the method was evaluated using relatively simple cases, specifically titles of books and movies, and names of authors and companies. The MUC-7 data contains New York Times articles on aviation and plane and sattelite crashes, where the boundaries are much more complex. For instance, in one of the articles, former president Bill Clinton is referred to as "President Clinton". Our code identifies the entire phrase "President Clinton" as a named entity, whereas in the test dataset the word "President" is not part of the named entity. Similarly, the data includes a reference to "Senate Majority Leader Robert J. Dole", which our code again identifies as a single entity, but in the data is identified as two separate entities ("Senate" and "Robert J. Dole"). It appears that the major hit to accuracy occurs because the method we employ automatically assumes that consecutive capitalized words are part of a single named entity, when in the test data that assumption turns out not to hold. This assumption is not made by the SVM-CMM method, and so it is able to acheive a higher precision rate on this difficult dataset.

Table 4. Hybrid results on MUC-7 data, dry-run test data

Span/Token	Precision	Recall	F-1
Token	.9266	.9435	.9348
Span	.8817	.8831	.8824

Table 5. Lex results on MUC-7 data, formal test

Span/Token	Precision	Recall	F-1
Token	.6758	.9571	.7922
Span	.5328	.8222	.6466

Table 4 shows the accuracy of the hybrid method on the dry-run test data. The accuracy is slightly higher than that of SVM-CMM alone, and significantly higher than that of Lex alone. What's interesting is that both the precision and recall figures of the SVM-CMM are improved. It's not the case that one method is being used to simply refine the other, e.g. using the SVM-CMM features to increase the precision of the Lex method, rather both the precision and the recall of the SVM-CMM are improved by the addition of the Lex input.

While the SVM-CMM appears to perform extremely well on the dry-run data, the formal test set from MUC-7 used articles from a slightly different domain. We are interested in seeing how much this affects SVM-CMM, relative to the effect on Lex, which is supposed to be more domain independent.

Table 5 shows the result of Lex on the formal test data, while Table 6 shows the results of SVM-CMM on the formal test data. We see a decrease in Span F-1 score of approximately .02, compared to the decrease in accuracy shown by the SVM-CMM of 0.11. Here, we see that Lex is indeed less susceptible to differneces between training and test domains. The hybrid approach on this data has accuracy as shown in Table 7. Although the hybrid approach outperforms both Lex and SVM-CMM, it also suffers approximately a 0.10 loss in Span F-1 score on the formal test data compared with the dry run data, i.e. it is also susceptible to differences between training and test domains.

We also ran 10-fold cross validated tests on the training data from Downey et al., to see whether a hybrid

Table 6. SVM-CMM results on MUC-7 data, formal test

Span/Token	Precision	Recall	F-1
Token	.8519	.8743	.8629
Span	.7682	.7643	.7662

Table 7. Hybrid results on MUC-7 data, formal test

Span/Token	Precision	Recall	F-1
Token	.8568	.8940	.8750
Span	.7797	.7850	.7823

Table 8. Lex results on Downey's training data, 10x cross validation

Span/Token	Precision	Recall	F-1
Token	.9362	.9277	.9319
Span	.7343	.8137	.7720

approach has any benefit in cases where the decision boundaries are perhaps a bit clearer. For these tests, we considered the entire training set as a whole (actors, books, films and companies). Results for Lex are shown in Table 8, while results for SVM-CMM and the hybrid approach are shown in Tables 9 and 10, respectively.

Here, we see that our Lex implementation barely outperforms the SVM-CMM method on this dataset. Both have a token F-1 measure of around 0.93. However, the Hybrid approach once again outperforms both Lex and SVM-CMM, reaching a token F-1 measure of .945, and a span F-1 measure of .8337, approximately 0.06 better than either SVM-CMM or Lex alone.

6. Conclusion

In this paper, we have shown three main results. The first result is that an n-gram dataset built from a large web crawl, such as the one released by Google, can be used to perform the task of named entity recognition. Rather than using statistics based on sentences pulled from the target domain, as in (Downey et al., 2007), we have successfully used statistics based on a crawl of the entire web and have still achieved reasonable accuracy. Secondly, we have shown that for complex domains, such as the MUC-7 NER task, the Lex method may not perform as well as other methods. This is due largely in part to subjective definitions of what constitutes a named entity, however. On the MUC-7 task,

Table 9. SVM-CMM results on Downey's training data, 10x cross validation

Span/Token	Precision	Recall	F-1
Token	.9432	.9169	.9299
Span	.7186	.8248	.7681

Table 10. Hybrid results on Downey's training data, 10x cross validation

Span/Token	Precision	Recall	F-1
Token	.9410	.9491	.9450
Span	.8149	.8534	.8337

Lex had excellent recall, but somewhat poorer precision due to its including words like "President" in the phrase "President Clinton" as being part of the named entity. Finally, we have shown that for both clear domains like those tested by Downey, and more subjective domains like those present in the MUC-7 data, combining Lex with traditional lexical approaches can achieve higher accuracy than either alone.

References

Bikel, D. M., Schwartz, R., & Weischedel, R. M. (1999). An algorithm that learns what's in a name. *Mach. Learn.*, *34*, 211–231.

Borthwick, A., Sterling, J., Agichtein, E., & Grishman, R. (1998). Exploiting diverse knowledge sources via maximum entropy in named entity recognition.

Brants, T., & Franz, A. (2006). Web 1t 5-gram version 1.

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

Downey, D., Broadhead, M., & Etzioni, O. (2007). Locating complex named entities in web text. *IJCAI*.

Freitag, D., & Kushmerick, N. (2000). Boosted wrapper induction. *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence* (pp. 577–583). AAAI Press / The MIT Press.

Minkov, E., Wang, R. C., & Cohen, W. W. (2005). Extracting personal names from email: Applying named entity recognition to informal text. *HLT/EMNLP*. The Association for Computational Linguistics.

MUC-7 (1998). The seventh message understanding conference (MUC-7), 1998.

Pantel, P., & Lin, D. (2001). A statistical corpus-based term extractor. *AI '01: Proceedings of the 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence* (pp. 36–46). London, UK: Springer-Verlag.

Silva, J., Kozareva, Z., Noncheva, V., & Lopes, G. (2004). Extracting named entities. A statistical approach. *TALN*. Poster Paper.