# Re-implementation of An Algorithm that Learns What's in a Name

**Brandon Plaster**
Cornell Tech
New York, NY, USA
bp364@cornell.edu

## Abstract

This paper re-implements and expands on the Bikel'99 paper "An Algorithm that Learns What's in a Name." The original paper presents a hidden Markov model that learns to classify names, times, and numerical quantities for the Named Entity task using English data from the 6th and 7th Message Understanding Conferences [MUC-6, MUC-7], as well as Spanish data from the First Multilingual Entity Task [MET-1]. To expand on Bikel's results, this paper looks at Spanish and Dutch datasets from the 2002 Conference on Computational Natural Language Learning, and adapts the previous bigram model into an n-gram model. This paper finds a similar effectiveness on the Spanish and Dutch datasets as with the original model in English, and also finds a slight (~2%) improvement using 3-gram and 4-gram models.

## 1 Introduction

The named entity task, as defined by the 6th and 7th Message Understanding Conferences, is the annotation of entity names (organizations, persons, locations), temporal expressions (dates, times), and numerical expressions (monetary values, percentages). The motivation for this task is that named entities help create an understanding of who the actors are in a text, what a text is about and what about it is important, and what information could be considered sensitive.

This paper re-implements Bikel's '99 paper "An Algorithm that Learns What's in a Name" and expands on it with both further dataset validation and an expansion of the model. The original paper looks at the task as a classification problem where the possible categories are either one of the aforementioned MUC definitions or part of the NOT-A-NAME (or "NAN") classification. Viewing the problem through this lens allows a comparison to other problems, such as the task of part-of-speech tagging, which has seen a great deal of success using hidden Markov models (HMMs) for classification. HMMs focus on local information in text, specifically preceding words and preceding tags. This reasoning motivated Bikel et al. to pursue a variant of the bigram HMM, as entities often rely on local preceding words, such as a person's name relying on the preceding "Mrs." or a monetary value relying on the preceding symbol "$". They were also motivated to use a learning model as most methods at the time of original publication used manually generated state models to tackle the problem. A further benefit to the HMM is that it can be generalized across languages and the challenge at the time was multi-lingual named entity analysis.

The original paper addressed the MUC-6 English and Spanish datasets, and used a MUC-supplied evaluation program. The MUC datasets were formatted using the Standard Generalized Markup Language, or SGML, and tagged entities appear in the datasets like the following:

Mr. <ENAMEX=PERSON>West</ENAMEX> won.

In order to expand on the learned datasets, this paper used the Spanish and Dutch datasets from the 2002 Conference on Computational Natural Language Learning (ConLL02) and a Twitter dataset from Ritter et al. These datasets were structured differently in that each word was on a different line and was followed either by the tag of the named entity or with an "O" if the word was NOT-A-NAME. As a result, the evaluation method from ConLL02 was used because it is an updated evaluation methodology and there existed functional evaluation software.

The MUC evaluation defines a label as correct if both the beginning and end tag are correctly located and gives partial credit for classifications that have at least a beginning or end tag correct. ConLL02 evaluation differs in that it only gives credit for classifications where the beginning and

end tag are correct, thus the ConLL02 evaluation is stricter and results in lower F-measures.

In order to expand on the original model, this paper adapted the original bigram model to an n-gram model. The n-gram HMM used the previous n – 1 words and tags for generating the classification of the current word.

## 2 Data and Evaluation

While other more recent definitions of the named entity task exist (such as the inclusion of different categories), the original paper adheres to the MUC definition. In order to expand to other datasets, such as those from the Conference on Computational Natural Language Learning, the code was generalized to allow for any set of classifications, or name-classes.

Three different languages, across four datasets were used for this paper. The English news dataset, originating from the MUC-6, is a collection of 30 Wall Street Journal documents. These texts were first reformatted to match the ConLL02 documents, and the training, development, and test data contained 169148, 6707, and 7115 lines respectively. The Spanish dataset, originating from the ConLL02, is a collection of Spanish News Agency texts where the training, development, and test data contained 273037, 54837, and 53049 lines respectively. From the same conference, the Dutch dataset consists of four editions of the Belgian newspaper "De Morgan" and the training, development, and test data files contained 218737, 40656, and 74189 lines respectively. A fourth dataset, in English, was used from Twitter. The training, development, and test data files contained 38736, 5274, and 4850 lines respectively.

As the MUC datasets were not tokenized, considerable effort was needed to properly parse the data. A complex set of regular expressions was used to both extract the tags and to break apart the tokens according to the MUC specific guidelines. The expression needed to address the following possible cases:

(1) Extract entity tags
(2) Extract ellipses
(3) Extract special characters
(4) Extract numbers that contain special characters
(5) Extract words excluding *'s*
(6) Extract contractions as single words
(7) Extract separately the *'s*
(8) Extract words that end with a period

The final pattern to address these possibilities was:

```
(1)  <.+?>(.+?)</.+?>|
(2)  ([.]+)|
(3)  ([{}()\"%,:?!$-])|
(4)  (\\d+([:/.,-]\\d+)*)|
(5)  (\\w+(?='[sS]))|
(6)  (\\w+'([a-rA-R]|[t-zT-Z])\\w*)|
(7)  ('s)|
(8)  (\\w+([']|[.](?!\\s*$))?)
```

While the expression worked well, the tokenization had a large impact on the results of the paper. While testing, variations of ~10% in F-measure were seen based on changes in the tokenization. These types of issues arose due to the ambiguity of language coupled with the strict definition of named entities. Take for example the simple case of:

<ENAMEX=PERSON>Bob</ENAMEX>'s hat.

The named entity here is Bob, so if the model finds that "Bob's" is the entity, then it is marked incorrect. Similar issues arise with other special characters such as dollar and percent signs.

As mentioned, the performance of the model was evaluated using F-measure, which is a combination of the precision and recall metrics, which are defined as:

$$P = \frac{number\ of\ correct\ responses}{number\ of\ responses}$$

$$R = \frac{number\ of\ correct\ responses}{number\ correct\ in\ key}$$

$$F = \frac{RP}{\frac{1}{2}(R+P)}$$

Responses are deemed correct if the label and both of the boundaries for the label are correct.

## 3 Approach

This section covers the approach used in both the original bigram model and the updated n-gram model. It should be noted that while an overview exists of the original model, this paper covers primarily the deviations and changes made to the original model, and does not include some of the unchanged specifics that are detailed in the original Bikel'99 paper, such as a list of all the original unchanged back-off models.

### 3.1 The Model Overview

The Markov model follows a 3-step generative process. First, the name-class is generated based on the previous name-classes and the previous words. Then, the first word of the name-class is

generated based on the current and previous name-classes. Finally, all the subsequent words of the name-class are generated based on the name-class and the preceding word until the final word of the name-class is generated.

This process is repeated until the entire sentence is generated and then using the Viterbi algorithm, the space of possible tag sequences is generated and maximized for the highest probable sequence.

Along with taking into account words, the model also looks at a small set of language-specific features and incorporates these features into a word-feature vector. An example of a word feature is if the word is capitalized, then the feature for that word is "allCaps". The word-feature vector is denoted $\langle w|f \rangle$, where $w$ is the word and $f$ is the single feature assigned to the word. A list of all the word features and their preference hierarchy can be found in Table 3.1 of Bikel'99.

### 3.2 The Formal Model

The top-level model consists of three sub-models.

(3.2.1) A model to generate a name-class based on the previous name-classes and previous words:

$$Pr(NC|NC_{-1}, w_{-1}, NC_{-2}, w_{-2}, NC_{-3}, w_{-3}, \ldots)$$

(3.2.2) A model to generate the first word in the name-class based on the current and previous name-classes:

$$Pr(\langle w|f \rangle_{first}|NC, NC_{-1}, NC_{-2}, \ldots)$$

(3.2.3) A model to generate all subsequent words in a name-class based on the current name-class and the previous word:

$$Pr(\langle w|f \rangle|\langle w|f \rangle_{-1}, NC)$$

To signify the end of a name-class, there is a virtual "_end_" word that must also be incorporated when both training the model and predicting the end of the name-class. As any current word could signify the end of a class, the probability must also be incorporated into the generation of words:

(3.2.4) A model to generate the final word of the name-class:

$$Pr(\langle end|other \rangle|\langle w|f \rangle_{final}, NC)$$

As is required by the Markov models, the probabilities are assumed independent. Thus, the first word of every name-class was generated by multiplying equations 3.2.1, 3.2.2, and 3.2.4 (where 3.2.4 conditioned on the previous word, since by definition, if the current word is the first word of a name-class, then the previous word is the last word of the last name-class.) All other non-first-words were generated using 3.2.3.

To help understand, consider the example:

Mr. <ENAMEX=PERSON>West</ENAMEX> won.

The likelihood would be:

$$\begin{aligned} Pr(NAN|\langle S \rangle, \text{"\_end\_"}) * Pr(\text{"Mr."}|NAN, \langle S \rangle) \\ * Pr(\text{"\_end\_"}|\text{"}\langle S \rangle\text{"}, \langle S \rangle) \\ * Pr(NAME|NAN, \text{"Mr."}) \\ * Pr(\text{"West"}|NAME, NAN) \\ * Pr(\text{"\_end\_"}|\text{"Mr."}, NAN) \\ * Pr(NAN|NAME, \text{"West"}) \\ * Pr(\text{"won"}|NAN, NAME) \\ * Pr(\text{"\_end\_"}|\text{"West"}, NAME) \\ * Pr(\text{"."}|\text{"won"}, NAN) \\ * Pr(\langle /S \rangle|NAN, \text{"."}) \\ * Pr(\text{"}\langle /S \rangle\text{"}|\langle /S \rangle, NAN) \\ * Pr(\text{"\_end\_"}|\text{"."}, NAN) \end{aligned}$$

As seen here, the virtual start word at the beginning of a sentence is considered a special "_end_" word as well. This is factored into training as well.

### 3.3 Training

Training the model was done through the simple counting of occurrences within the training set. For each of the aforementioned probabilities, they following the counting scheme:

$$\Pr(X|Y) = \frac{c(X,Y)}{c(Y)}$$

where c(Y) is the number of occurrences, or the count, of Y in the training data.

There are also several special "virtual" words and tags that need to be incorporated while training.

"<S>" is added at the beginning of every sentence and is the starting word. "<S>" is the associated tag for the starting word.

Similarly, "</S>" is added at the end of every sentence and is the stopping word. "</S>" is its associated tag.

As mentioned before, a special "_end_" word is used to signify the end of a name class. And

similarly, a special "_begin_" word is used to denote the first word of a name class. The "_begin_" word is used specifically for one of the original back-off models.

### 3.4 Smoothing and Back-off

Although ideally the training sets would contain adequate and diverse data to account for all possibilities, in reality, the data suffers from two problems: data sparsity (insufficient occurrences of all possible n-gram-to-word pairs) and unknown words.

In order to better handle the issue of data sparsity within the training sets, each of the three top-level models have sequences of weighted back-off models that get more generalized as the models are further backed off. The weights of the models are computed during testing, as they rely on the specific n-grams and words. The weight λ of a given model is defined for a probability Pr(X|Y) as:

$$\lambda = \left(1 - \frac{old\ c(Y)}{c(Y)}\right) * \frac{1}{1+\frac{unique\ outcomes\ of\ Y}{c(Y)}}$$

The direct weight λ is multiplied by the model from which the model is backing off (i.e. the model for which *old c(Y)* is derived) and the weight (1 – λ) is multiplied by the model to which the model is backing off to.

For models that were not defined as conditional probabilities, the assumption was made that with each back-off model, the model slowly approaches a more generalized and uniform distribution, and thus the weights should reflect that.

The expansion from a bigram to an n-gram model was straightforward in computing the additional back-off models, as the back-off model for an n-gram model was simply the (n-1)-gram model. Back-off models from the bigram model and after can be seen in Table 3.2 of Bikel'99.

### 3.5 Unknown Words

In order to build a model for unknown words, the training set was split into two equal parts. The first half was treated normally and was used to build the known words tables. For the second half, if a word was found in the known words tables, it was treated normally, however, if it was not found in the known word tables, it was treated as an unknown word and its statistics were added to the unknown word model which was represented by a special "_UNK_" word.

### 3.6 Decoding

To determine the tagging sequence of maximum likelihood for a given sentence, rather than comparing all the possible sequence likelihoods, the Viterbi decoding algorithm was used. This allows for the decoding to occur linearly with the number of possible tags. This is done through dynamic programming, by progressing forward through the sequence of possible states, and storing the current maximum probability along with the back pointer to the previous state.

## 4 Results

Throughout development, a specific development dataset was used for each different corpus. For MUC6, half of the "NE.02may95" set was held off until the results for formal testing, while the other half was used for development. For ConLL Spanish and Dutch datasets, the corresponding dataset *testa* was used for development and *testb* was used for formal results. For the Twitter data, half was used as a holdout set for formal results and half was used for development.

Table 1 showcases the results for the four datasets using the original re-implemented bigram model. As to be expected, the MUC6 results do not match the original paper's results due to the new evaluation methodology from ConLL02. When comparing the results of the Spanish and Dutch datasets to the 2002 conference submission results, the results are within the range of the submissions. For Spanish, the range of F-measures was from 61.0% - 81.4%, and for Dutch, the range of F-measures was from 53.1% - 77.1%. The Twitter results, while much worse than the other datasets, have an order of magnitude smaller training set, and so worse results were expected. Also, Twitter data, by nature, is much less well formed than news texts and contains a higher variation in the type of language seen, implying that the data is inherently sparser.

| | MUC6 | ConLL02 Spanish | ConLL02 Dutch | Twitter |
|---|---|---|---|---|
| **F-measure** | 79.2% | 69.4% | 64.5% | 31.3% |
| **Precision** | 83.4% | 66.3% | 66.0% | 35.6% |
| **Recall** | 75.4% | 72.8% | 63.0% | 28.0% |

**Table 1: Bigram Model Language Comparison – Formal**

Table 2 shows the comparison of F-measures across the four datasets for different *n* numbers.

Both the 3-gram and 4-gram models show improvement over the original model. For *n* numbers above 4, the results show either a plateau or, in some instances, a slight reduction in F-measure. This is thought to be a result of over fitting and sparsity, as the model begins to take into account unrelated words that are too far away while not having enough occurrences of the 5-gram sequences.

| | n = 2 | n = 3 | n = 4 | n = 5 |
|---|---|---|---|---|
| **MUC6** | 79.2% | 80.3% | 81.1% | 80.9% |
| **ConLL02 Spanish** | 69.4% | 72.1% | 73.5% | 73.6% |
| **ConLL02 Dutch** | 64.5% | 64.8% | 65.6% | 65.5% |
| **Twitter** | 31.3% | 36.9% | 37.8% | 35.9% |

**Table 2: n-Gram Model F-measure – Formal**

Table 3 shows the in-depth tag analysis compared across different *n* values for the MUC6 data. While most of the tags followed the overall trend's increase with *n* value, it is interesting to note the PERSON tag drop-off after *n=3*. The practical explanation behind this is that, given that PERSON names tend to be of length three (i.e. first, middle, and last name), then after *n=3*, the model begins to over fit the data.

| | n=2 | n=3 | n=4 |
|---|---|---|---|
| **OVERALL** | 79.2% | 80.3% | 81.1% |
| DATE | 80.0% | 84.5% | 84.5% |
| LOCATION | 68.8% | 71.3% | 71.7% |
| MONEY | 82.7% | 85.1% | 89.0% |
| ORGANIZATION | 79.7% | 78.9% | 79.8% |
| PERCENT | 60.3% | 63.1% | 63.2% |
| PERSON | 85.3% | 88.1% | 86.6% |

**Table 3: MUC6 n-Gram Tags F-measure – Formal**

## 5 Conclusion

While better results have been seen using other methods for identifying named entities, the results on the expanded Dutch and Spanish datasets show the benefits of a primarily language-independent approach to the task. There is also support of the benefits of expanding the bigram model to a 4-gram model for most tags; however, care should be taken for the possibility of over fitting with a higher-ordered model as specific tags react differently to different gram counts.

Furthermore, as the results were highly dependent on tokenization, care should be taken in being too strict in the definition of named entities.

## Reference


Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (EMNLP '11). Association for Computational Linguistics, Stroudsburg, PA, USA, 1524-1534.

Daniel M. Bikel, Richard Schwartz, Ralph M. Weischedel, An Algorithm that Learns What's in a Name, Machine Learning, v.34 n.1-3, p.211-231, Feb. 1999

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: language-independent named entity recognition. In *proceedings of the 6th conference on Natural language learning - Volume 20* (COLING-02), Vol. 20. Association for Computational Linguistics, Stroudsburg, PA, USA, 1-4.

GuoDong Zhou and Jian Su. 2002. Named entity recognition using an HMM-based chunk tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (ACL '02). Association for Computational Linguistics, Stroudsburg, PA, USA, 473-480.

Nancy Chinchor. 1998. MUC-7 Named Entity Task Definition Dry Run Version, Version 3.5 17 September 1997. In *Proceedings of the Seventh Message Understanding Conference* (MUC-7). Fairfax, Virginia: Morgan Kaufmann Publishers, Inc.