



RICH ENTITY TYPE RECOGNITION IN TEXT

Senior Thesis

Rishav Bhowmick
rishavb@cmu.edu

Advisors

Kemal Oflazer
ko@cs.cmu.edu

Noah A. Smith
nasmith@cs.cmu.edu

Mentor

Michael Heilman
mheilman@cs.cmu.edu

May 1, 2010

ABSTRACT

Many natural language processing (NLP) applications make use of entity recognition as preprocessing step. Therefore there is a need to identify nouns (entities) and verbs in free text. The task boils down to using machine learning techniques to train a system that can perform entity recognition with performance comparable to a human annotator. Challenges like lack of large annotated training data corpus, impossible nature of listing all entity types and ambiguity in language make this problem hard. There are existing entity recognizers which perform this task but with poor performance. An obvious solution is to improve the performance of an existing entity recognizer. This Senior Thesis will analyze the existing features, through a series of experiments, that are important for the recognizer. This project will also suggest usage of additional features like Word Cluster features and Bigram features to improve the performance of the system. At the same time, experiments will show that lack of large annotated training data may not be as big of a problem it might seem at first.

ACKNOWLEDGEMENT

Tons of thanks to my advisors Dr. Kemal Oflazer and Dr. Noah A. Smith, and mentor Michael Heilman for their constant support throughout the year. Without their help and guidance, I would not be able to experience the field of natural language processing (NLP). At the same time, I would like to thank Dipanjan Das who provided the word clusters for this project. I also thank Dr. Smith for granting me access to Malbec cluster where most of the work was done. Not to forget, I thank Dr. Behrang Mohit to access his VM in Qatar. I want to take this opportunity to thank Dr. Brett Browning, Dr. Bernardine Dias and Dr. Majd Sakr who let me use the Student Robotics Lab to set up my workspace.

I would also like to thank Dr. Dudley Reynolds for his constant help in making me think about the writing aspect of the thesis by providing me with comments and suggestions.

Last but not the least, I would like to thank my parents for their everlasting support and understanding my need to stay at university late nights and early mornings. Finally, a big hug to my friends and classmates who were with me during these times.

TABLE OF CONTENTS

1	Introduction	5
2	Background and Related Work	6
2.1	Supersenses	6
2.2	Sequence Tagging	7
2.3	Perceptron-trained HMM	7
2.4	Incorporating Word Cluster Feature.....	8
2.5	The Baseline Tagger	8
3	Evaluation Metrics	9
3.1	Precision (P)	9
3.2	Recall (R)	9
3.3	F1.....	10
4	Approach.....	10
4.1	Varying Training Data Size	10
4.2	Feature Addition, Removal and Modification.....	10
5	Experiments and Results.....	11
5.1	Setup	11
5.2	Feature Extraction.....	11
5.3	Experiment 1- Training Data Size	11
5.4	Experiment 2- Feature Ablation.....	12
5.5	Experiment 3- Context Size.....	12
5.6	Experiment 4- Addition of Word Cluster Features	13
5.7	Experiment 5- Addition of Bigram Feature	14
6	Inference and Error Analysis	15
6.1	Looking At The Baseline	15
6.1.1	Input Data Analysis	15
6.1.2	First Sense Not The Winner Always	16
6.2	Context Size Analysis.....	16
6.3	Word Cluster Feature Benefits	16
7	Conclusion And Future Work	17
	References	18

1 INTRODUCTION

Common applications of natural language processing (NLP) include summarization of text, classifying documents or automatic answering of questions posed in natural language. Each of these applications require entity type recognition in the text as a pre-processing step. Here, “entity” refers to concrete and abstract objects identified by proper and common nouns. Entity recognition focuses on detecting instances of types like *person*, *location*, *organization*, *time*, *communication*, *event*, *food*, *plant*, *animal*, and so on. For example, an entity recognizer would take the following sentence as input:

George Washington was the first President of the United States of America.

and output:

<noun.person> *George Washington* **</noun.person>** *was the first* **<noun.person>** *President* **</noun.person>** *of the* **<noun.location>** *United States of America* **</noun.location>**.

Humans generally have no problems finding out what type a noun belongs to. In the example above, a human would look at “President” and know that it is of type *Person*. He/she would also know a location or organization can have a President. Additional knowledge about the country, makes him/her think it is a *location*. Finally, “George Washington” has to be a *person* as a president can only be a human.¹ The way a human figures out the entity types could be summarized in the following points:

- Recalling what entity type a word most likely belongs to
- Looking at the context the word appears in.
- Looking at features like word capitalization, punctuation marks. For example, the use of an upper-case letter after punctuation marks like periods or question marks does not indicate that the first word of the sentence is a proper noun. But in general, the use of capitalization does suggest a person, organization, or location.

Our task is to use machine learning techniques to train a system that can do entity type recognition with a performance comparable to human annotator. This problem is hard for a variety of reasons. In general, it is not possible to list all possible instances of a single entity type and feed it to the machine. The lack of a large annotated data corpus for training is another major impediment. Due to these reasons, existing entity recognizers are not very accurate (F1 ranging 70%-80%) (Shipra, Malvina, Jenny, Christopher, & Claire, 1900; Carreras, Màrquez, & Padró, 2002).² The obvious task then is to improve the performance of existing machine tagging systems. This would be achieved by looking for features (new as well as ones used with existing taggers) that affect the performance of the tagger the most. Additionally, finding out how much training data is needed, can help solve the problem of the lack of a large annotated training data corpus.

This Senior Thesis performs analysis on an existing entity recognizer by considering removal and addition of certain features that affect the performance of the recognizer. It will also address the issue of

¹ unless it is a line out of a fantasy novel, where an animal (other than a human) presides.

² Please refer to the section 3 for the definition of F1

whether large training data sets are necessary or not. The outcome of this project will be a step forward in making an enhanced entity recognizer which in turn will benefit other NLP applications.

2 BACKGROUND AND RELATED WORK

The entity recognizer that we are analyzing is the Supersense Tagger (SST) (Ciaramita & Altun, 2006). The tagger performs sequence tagging with a perceptron-trained Hidden Markov Model (HMM). The following section will describe the tag set used, the type of sequence tagging and the model used for training.

2.1 SUPERSENSES

The entity-type tag set we use in this research project contains types referred to as *supersenses* (Ciaramita & Altun, 2006; Ciaramita & Johnson, 2003; Curran, 2005). As opposed to the usual entity types *Person*, *Location* and *Organization*, and sometimes *Date* used in earlier Named Entity Recognition (NER),³ the supersense tag set includes 26 broad semantic classes. These semantic classes are labels used by lexicographers who developed Wordnet (Fellbaum & others, 1998), a broad-coverage machine readable lexical database which has proper and common nouns, verbs, adjectives and adverbs interlinked via synonym, antonym, hypernym, hyponym and variety of other semantic relations. **Table 1** (from (Ciaramita & Altun, 2006)) shows the supersense labels for nouns. Wordnet is used to lemmatize a word and provide the most frequent supersense for the word.⁴ Because this tag set suggests an extended notion of named entity, this particular process of recognition is called *supersense tagging*.

Furthermore, supersenses have been used to build useful latent semantic features in syntactic parse re-ranking (Koo & Collins, 2005). Supersense Tagging, along with other sources of information such as part of speech, domain-specific NER models, chunking and shallow parsing, can contribute a lot to question answering and information extraction and retrieval (Ciaramita & Altun, 2006).

NOUNS			
SUPERSENSE	NOUNS DENOTING	SUPERSENSE	NOUNS DENOTING
act	acts or actions	object	natural objects (not man-made)
animal	animals	quantity	quantities and units of measure
artifact	man-made objects	phenomenon	natural phenomena
attribute	attributes of people and objects	plant	plants
body	body parts	possession	possession and transfer of possession
cognition	cognitive processes and contents	process	natural processes
communication	communicative processes and contents	person	people
event	natural events	relation	relations between people or things or ideas
feeling	feelings and emotions	shape	two and three dimensional shapes
food	foods and drinks	state	stable states of affairs
group	groupings of people or objects	substance	substances
location	spatial position	time	time and temporal relations
motive	goals	Tops	abstract terms for unique beginners

TABLE 1 NOUNS SUPERSENSE LABELS, AND SHORT DESCRIPTION(CIARAMITA & ALTUN, 2006)

³ *Named Entity* here refers to proper nouns only. Some of the earlier works include (Borthwick, 1999; Carreras et al., 2002; Finkel, Grenager, & Manning, 2005; Florian, Ittycheriah, Jing, & Zhang, 2003; Mikheev, Moens, & Grover, 1999; Zhou & Su, 2002)

⁴ *Lemmatizing* refers to finding the root of a word. For example, the lemma of *ran* is *run*, while the lemma of *said*, is *say* and for *teachers*, it is *teacher*.

2.2 SEQUENCE TAGGING

In NLP, people often seek to assign labels to each element in a sequence. Here, sequence generally refers to a sentence where the words are the elements. Let $X = \{x^1, \dots, x^k\}$ denote the vocabulary of sequence elements, and $Y = \{y^1, \dots, y^m\}$ the vocabulary of tags. The task of sequence tagging is to assign lexical categories $y \in Y$ to words $x \in X$ in a given natural language sentence. NER and part-of-speech (POS) tagging are two such tasks which involve sequence labeling or tagging.

Label assignment may involve simply matching the element to a dictionary entry. For many NLP applications, however, the process can be improved upon by assigning the labels sequentially to elements from a string (usually a sentence). This allows the choice of a label to be optimized by considering previous labels.

The tagging scheme used in the Supersense Tagger is *begin-in-out (BIO) tagging scheme*. In this scheme, each token/word in a sentence is either marked as beginning of a chunk (B), continuing a chunk (I) or not part of any chunk (O) based on patterns identified on the basis of the training data. In the following example,

George Washington was the first President of the United States of America.

“George” would be labeled as *B-noun.person* and “Washington” as *I-noun.person*. This is because “George” is the beginning of the *noun.person* phrase and “Washington” continues that supersense. Similarly, “United” would be labeled as *B-noun.location*. Following this, “States”, “of” and “America” would be labeled as *I-noun.location*. The remaining tokens are labeled *O*.

2.3 PERCEPTRON-TRAINED HMM

As mentioned earlier, the SST uses a perceptron-trained Hidden Markov Model (P-HMM)(Ciaramita & Altun, 2006, 2005). This model uses Viterbi and perceptron algorithms to replace a traditional HMM’s conditional probabilities with discriminatively trained parameters. It has been successfully implemented in noun phrase chunking, POS tagging and Bio-medical NER (M. Collins, 2002; Jiampojarn, Kondrak, & Cherry, 2009) and many other NLP problems.

The advantages of using this kind of model are that it does not require uncertain assumptions, optimizes the conditional likelihood directly and employs richer feature representation (Ciaramita & Altun, 2006). These kinds of models represent the tagging task through a *feature-vector representation*.

A feature represents a morphological, contextual, or syntactic property and typically looks like this for example

$$\Phi_{100}(y, x) = \begin{cases} 1 & \text{if current word } w_i \text{ is the} \\ & \text{and } y = \textit{Determiner} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

A vector of these features is represented as:

$$\Phi(x, y) = \sum_{i=1}^d \sum_{j=1}^{|y|} \Phi_i(y_j, x) \quad (2)$$

Here d is the total number of features, x is the token that is being tagged and y is the label. The task of tagging can be represented as learning a discriminant function F which is the linear in a feature representation Φ defined over the space:

$$F(x, y; \mathbf{w}) = \langle \mathbf{w}, \Phi(x, y) \rangle \quad F: X \times Y \rightarrow \mathbb{R} \quad (3)$$

\mathbf{w} , is the parameter vector of d dimensions. For an observation sequence \mathbf{x} , the SST makes predictions by maximizing F over the response variables:

$$f_{\mathbf{w}}(\mathbf{x}) = \operatorname{argmax}_{y \in Y} F(x, y; \mathbf{w}) \quad (4)$$

The process involves Viterbi decoding with respect to $\mathbf{w} \in \mathbb{R}^d$. The complexity of the Viterbi algorithm scales linearly with the length of the sequence (Manning & Schütze, 1999).

In a nutshell, the perceptron is not a probabilistic model. It keeps scores for sequences and decides labels based on the scores.

The performance of perceptron-trained HMMs is competitive and comparable in performance to that of Conditional Random Field models (Ciaramita & Altun, 2006; Collins, 2002).

2.4 INCORPORATING WORD CLUSTER FEATURE

The addition of new features such as word clusters in a more restricted task of NER has shown considerable improvement in performance in the system (Lin & Wu, 2009). A Word cluster is a grouping of words which fall in similar context. An example of a word cluster could be ["pet", "cat", "dog", ...].

The use of word clusters alleviates the problem of lack of annotated data. Word clusters are generated from unlabeled data, which is available in plenty. Once word clusters are created, the feature that holds "word belonging to a particular cluster", can be used in a supervised training setting. Hence, even when a word is not found in the training data, it may still benefit from the cluster-based features as long as the word belongs to the same cluster with some word in the labeled data. For example, if the word "cat" is in the training data and the tagger encounters a new word "dog" in the test set, the tagger would not know what to do with the unseen word. But if a word cluster contains both of these words, the word cluster feature will be fired and the two words can share information. This improves the tagged output of the unseen words. In this project the word cluster used were created using Distributed K-means clustering (Lin & Wu, 2009).⁵

2.5 THE BASELINE TAGGER

The baseline tagger used for comparison with modified taggers is a reimplementation⁶ of the SST. It uses the same feature set as that of the SST to tag words which include both proper and common nouns, and verbs. The experiments conducted also involve tagging verbs along with nouns. The training data for the verbs is extracted the same way as for the nouns.

⁵ Word clusters are generated using Distributed K-means clustering, by Dipanjan Das (LTI).

⁶This re-implementation has been done by Michael Heilman (LTI, CMU)

3 EVALUATION METRICS

The following evaluations metrics are used to evaluate the performance of our tagger.

3.1 PRECISION (P)

Precision measures the percentage of the supersenses identified by the tagger that are correct. Large precision indicates almost everything the tagger tags are correctly tagged.

$$\text{overall precision} = \frac{\text{number of correctly tagged phrases by the tagger}}{\text{number of tagged phrases by the tagger}} \quad (5)$$

Note that a word is a phrase of size 1 token .

3.2 RECALL (R)

Recall measures the percentage of the supersenses in the test set that are actually correctly identified. In other words, it tells, how few errors of omission are made by the tagger.

$$\text{overall recall} = \frac{\text{number of correctly tagged phrases by the tagger}}{\text{number of correct hand labeled phrases from the test data}} \quad (6)$$

Incorrectly tagging a phrase as Y that should have been labeled X will lower recall for X and lower the precision for Y.

Here is an example to illustrate precision and recall for a sentence (O refers to no supersense for the particular token):

	Hand Labeling	Machine Tagging
John	B-noun.person	B-noun.location
Smith	I-noun.person	I-noun.location
is	B-verb.stative	B-verb.stative
in	O	B-noun.place
Doha	B-noun.location	B-noun.person

Here, number of correctly tagged phrases is 1. Number of tagged phrases is 4. Hence, overall precision is 1/4. Number of labeled phrases in this case is 3. Hence, overall recall is 1/3.

Another example:

	Hand Labeling	Machine Tagging
John	B-noun.person	B-noun.person
Smith	I-noun.person	B-noun.location
is	B-verb.stative	O
in	O	O
Doha	B-noun.location	B-noun.location

Here, number of correctly tagged phrases is 1, as “John Smith” is wrongly tagged. Number of tagged phrases is 3. Hence overall precision is 1/3. Number of labeled phrases is 3. Hence, overall Recall is 1/3.

3.3 F1

F1 (F-score) is simply the geometric mean of precision and recall, and combines the two scores.

$$F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (5)$$

EQUATION 1 F1

4 APPROACH

Our approach towards improving the performance of the SST involves two trials. For both trials, tagger performance is evaluated based on affects on F1.

4.1 VARYING TRAINING DATA SIZE

In the first trial, we experiment with different sizes of training data to determine if a threshold exists after which additional data does not improve the performance of the tagger. Therefore we train the system using different size training data and evaluate the trained model with test data.

4.2 FEATURE ADDITION, REMOVAL AND MODIFICATION

In the second trial, we investigate the effect of adding, removing and modifying features used in the tagging process and gauging which affect the performance of the tagger.

For this, we devise a series of experiments which involves removing one feature at a time and evaluating the tagger output. This task is termed as *feature ablation*. When a feature affects the F1 by +/-2 points, we mark it for future experimentation. As for the other features, we take conjunctions of these features and check if they collectively affect F1. Some of the baseline features from SST include most frequent sense (from Wordnet), POS tags, word shape (upper-case or lower-case, upper-case after period and so on) and label of preceding words.

Context is an essential feature while tagging words. As shown in the example in the section 1, while tagging *George Washington*, the knowledge about a President being of type *person* helps with tagging *George Washington* as *person*. The baseline tagger only looks at +/- 2 words around the current word being tagged. We perform additional experiments by reducing the context to not looking at any words (removing the existing context features) and then increasing the context to +/- 4 words (adding new context features) to see if the extra context helps.

Like any other entity recognizer, the SST can encounter words that it has never seen in the training corpus. A plausible solution to improve performance of the tagger to tag these unseen words correctly is to incorporate word cluster features (refer to section 2.4). As a result, we add new word cluster features for the current word and words (+/- 2 words) around it. We conjoin other *strong* features along with the word cluster features. These strong features are shortlisted from the feature ablation experiments and contextual analysis mentioned earlier.

5 EXPERIMENTS AND RESULTS

5.1 SETUP

We tested our tagger on the Semcor corpora (Miller, Leacock, Tengi, & Bunker, 1993) containing syntactically and semantically tagged text (articles) from the Brown Corpus. This includes the nouns and verbs labeled with their supersense. The Semcor data was split into 3 parts: Training, Development and Testing. The three parts were created by randomly selecting the articles. The size of the three parts were as follows:

	Training data	Development Data	Testing data
Number of sentences	11,973	4113	4052
Number of Tokens	248,581	92,924	93,269

5.2 FEATURE EXTRACTION

The Tagger extracts features with their names and values for a particular token in a sentence. All of these are aggregated to get the feature vector or score for the whole sentence. From the programming point of view, if we want to add a new feature, we create a mapping of the feature name to the value and add it to the feature vector. If we do not want to include a feature, we simply do not add the mapping to the feature vector.

5.3 EXPERIMENT 1- TRAINING DATA SIZE

In order to find out how the size of training data affects the performance of the tagger, multiple percentages of training data were made. The tagger was then trained on each of these parts and evaluated. The F1's were calculated at intervals of 5% of the training data.

Figure 1 shows the results for the varying F1 with respect to the amount of training data used.

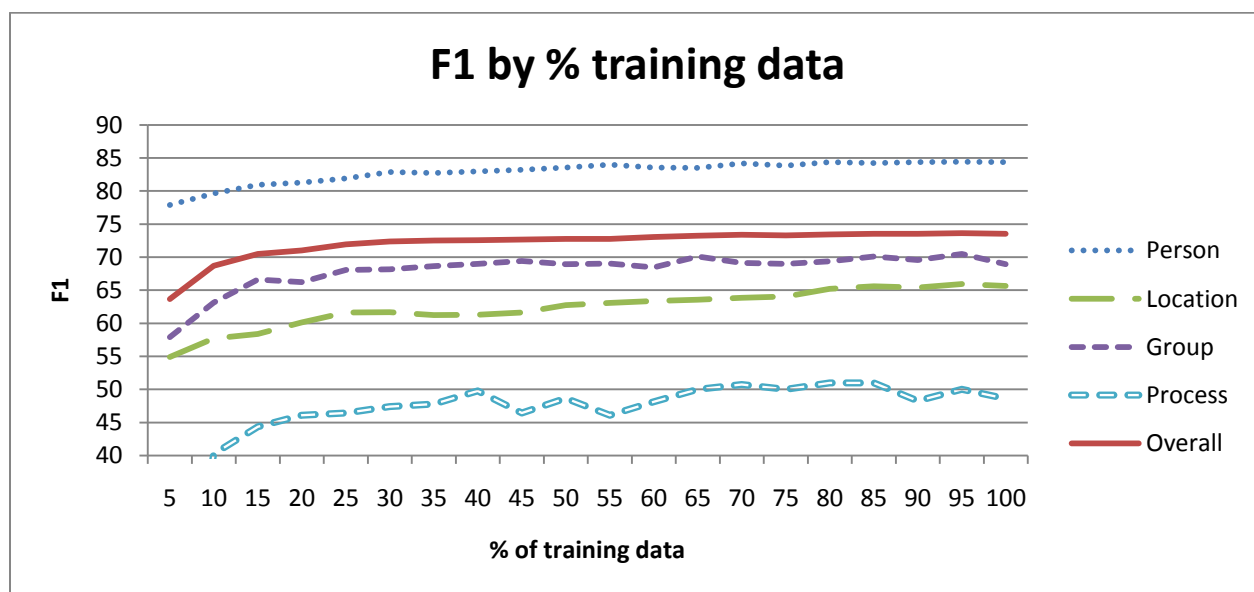


FIGURE 1 F1 BY % TRAINING DATA

The results of the experiment indicate that after about 33% of the training data is used, the overall F1 does not increase drastically. Results for the individual supersenses (a subset of supersenses shown in **figure 1**) are similar. However, there are fluctuations in the graph for *process*, while smoother curve for *Person*.

5.4 EXPERIMENT 2- FEATURE ABLATION

The experiment to find out which feature impacts the performance of the tagger the most is conducted by removing one baseline feature at a time. These features include:

- The first-sense feature which refers to the most frequent sense of the word or token being labeled
- The POS feature which includes the part-of-speech label of the current word and the two words in front and back of the current word or token
- The word shape feature which includes capitalization of the current word/token, presence of punctuation as the previous token along with the next two and previous two tokens
- The previous label feature which is simply the label of the previous token

Feature Removed	F1	Precision	Recall
Baseline	73.54%	73.12%	73.98%
First Sense (Most frequent sense)	57.11%	57.12%	57.09%
Part of speech	73.11%	73.11%	72.50%
Word Shape	73.51%	73.13%	73.90%
Previous Label	73.51%	73.15%	73.89%

TABLE 1 FEATURE REMOVED AND RESULTING F1

Table 2 shows that the First Sense feature has the greatest impact on the performance of the SST, as the performance of the tagger suffers severely when it is removed. But at the same time, the removal of previous label feature affects the performance minutely. This is striking considering the fact that the tagger performs sequence tagging.

5.5 EXPERIMENT 3- CONTEXT SIZE

In this experiment, we look at how context size affects the performance of the tagger. The Baseline Tagger looks at the current word and two words before and after it. The tagger extracts all other features like most frequent sense of current word, POS and word shape of the current word along with two words before and after it, and label for the previous word before the current word. If the context is limited to “no words”, then none of the other features are considered. Obviously the current token to be tagged is visible. When the context is “current word”, it includes the other features along with the current token. For larger context (+/-2, +/-3, +/-4) the POS and word shape of the surrounding words are the additional features incorporated into the feature vector.

Experiments on context size are tabulated in **table 3**:

Context	F1	Precision	Recall
Baseline - Current word +/- 2 words	73.54%	73.12%	73.98%
No words	70.19%	69.09%	71.32%
Current word	70.94%	69.23%	72.75%
Current word +/- 1 word	73.34%	72.67%	74.02%
Current word +/- 3 words	73.50%	73.11%	73.89%
Current word +/- 4 words	73.27%	72.84%	73.71%

TABLE 2 CONTEXT SIZE AND RESULTING F1, P AND R

As shown, the highest F1 results from the baseline current word +/- 2 words context size. Reducing the context lowers the F1 and increasing does not affect the F1 much.

5.6 EXPERIMENT 4- ADDITION OF WORD CLUSTER FEATURES

The first step to adding word cluster feature, is to fetch the word cluster ID of the word that is being tagged. The word cluster ID is simply a pointer to which cluster the word belongs to. This experiment considers a context of current word +/- 2. Hence, the cluster ID for these words are also needed. The next step involves adding the new features for each of these words as mentioned in section 5.2.

Word clusters of various sizes $K = 64, 128, 256, 512, 1024, 2048$ were used for this experiment. **Table 4** shows the results for the above experiment.

K (word cluster size)	F1	Precision	Recall
Baseline	73.54%	73.12%	73.98%
64	73.69%	73.26%	74.13%
128	73.77%	73.37%	74.18%
256	73.77%	73.39%	74.15%
512	73.73%	73.37%	74.09%
1024	73.91%	73.52%	74.30%
2048	73.80%	73.46%	74.15%

TABLE 3 WORD CLUSTER WITH VARYING K AND RESULTING F1, P AND R

Generally, the addition of word cluster feature has not led to poor results. As for $K = 1024$, the performance of the tagger shows promising results.

The obvious next step is to conjoin some strong features with the word cluster feature to expect good results. So following this experiment, we added the "First Sense" feature in conjunction with the existing word cluster feature. We chose the "First Sense" feature as it is the strongest of the features as shown in Experiment 2. We again evaluated the tagger with the same set of clusters and led to the following results in **table 5**:

K (word cluster size)	F1	Precision	Recall
Baseline	73.54%	73.12%	73.98%
64	73.69%	73.32%	74.02%
128	73.74%	73.32%	74.16%
256	73.80%	73.38%	74.22%
512	73.76%	73.39%	74.13%
1024	73.74%	73.42%	74.07%
2048	73.73%	73.34%	74.12%

TABLE 5 WORD CLUSTER AND FIRST SENSE FEATURE WITH VARYING K AND RESULTING F1, P AND R

These results also suggest the addition of word cluster features leads to slightly improved results but “First Sense” feature did not provide any extra help.

In both of the above evaluations, the SST had cluster features for all the words (current, previous and next, previous 2 and next 2). Next step, we took the case which had the best performance- the one with K=1024 and removed the cluster features for two words to the right and left of the current word. We evaluated the trained model and the result was as follows:

$$\mathbf{F1 = 73.98\%} \quad P = 73.58\% \quad R = 74.37\%$$

After this, we removed the cluster features for one word to the right and left of the current word. The result was as follows:

$$\mathbf{F1 = 73.83\%} \quad P = 73.45\% \quad R = 74.21\%$$

5.7 EXPERIMENT 5- ADDITION OF BIGRAM FEATURE

We also trained the tagger with Bigram feature (the feature considers groups of two words; for e.g.: current stem = “are” AND next stem =”going”) . The F1 for this turned out to be **74.10%**. The precision was 73.60% while the recall is 74.53%

The downside of training with the Bigram feature is that in the worst case, it would add $|V|^2$ features to the model, where V is the vocabulary. This eventually leads to more time for training. This experiment took around four hours as opposed to the previous ones which took only about two hours. Also note that the result achieved by including Bigram feature (with F1 = 74.10%) is almost equivalent to the result achieved by including word cluster feature for the current word and two words around it (with F1 = 73.98%).

6 INFERENCE AND ERROR ANALYSIS

6.1 LOOKING AT THE BASELINE

Figure 2 shows the F1 for some of the supersenses after using the baseline tagger.

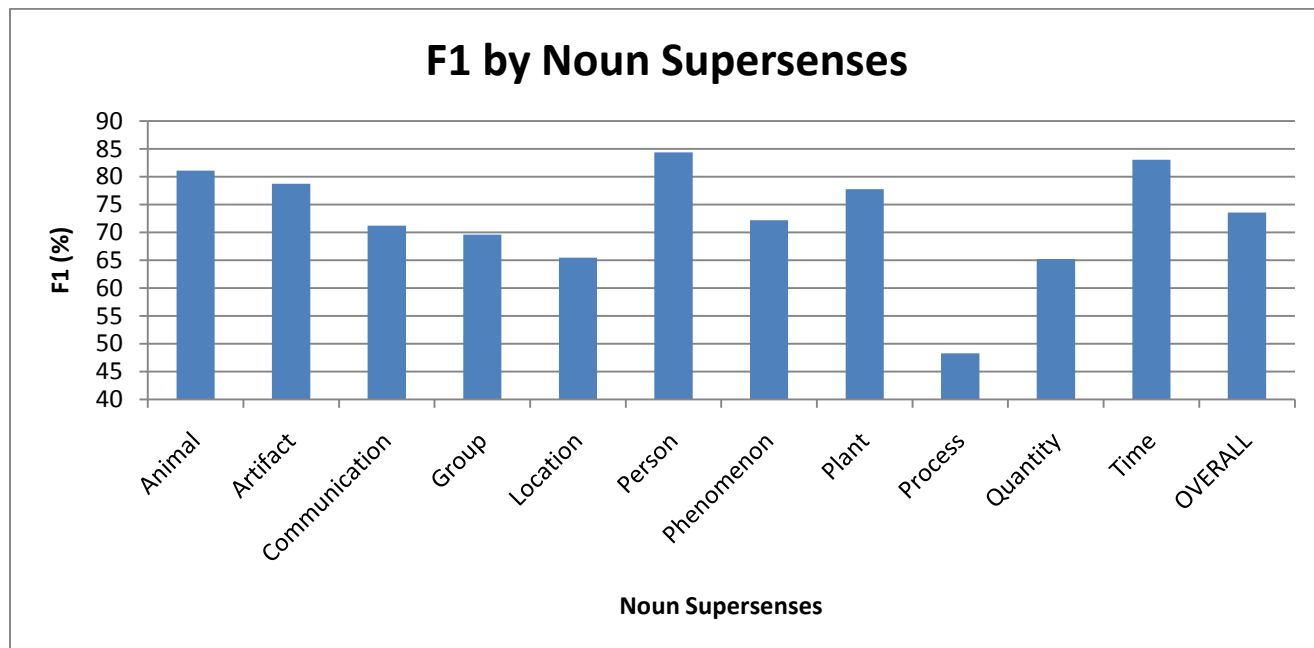


FIGURE 2 F1 BY SUPERSENCES

We can clearly see that supersense *noun.process*, *noun.location*, *noun.quantity* and some other supersense have low F1. This could be attributed to the fact that these supersenses have lesser number of training instances. Another possible reason could be that the most frequent sense for certain words from Wordnet is not the right sense for these words. The following sub-sections will address the two issues in hand.

6.1.1 INPUT DATA ANALYSIS

When we look at the number of instances of a supersense in the training set and compare with the F1 (which is also present in **figure 1**). **Table 6** contains the noun supersenses along with the number of instances in the training set, F1 and number of those instances in the test set. The table is sorted by the size of the instances in training set. The general trend would be – more the number of training instances, higher the F1. But this is not true in some cases like in that of *noun.shape*, *noun.motive* and *noun.process*. While *noun.process* has more instances in training set, its F1 is way lower than that of *noun.shape* or *noun.motive*. The small size of the test set affects the F1 vastly, even if a minor change in number of correctly tagged changes. This explains the fluctuations for *noun.process* curve in **figure 1**.

Noun Supersense	Number of instances in training set	F1 (%)	Number of Instances in test set
noun.person	8534	73.03	3080
noun.artifact	5498	80.99	1859
noun.act	4516	79.35	1870
noun.communication	4134	67.44	1448
noun.group	3840	83.39	1096
noun.cognition	3574	73.54	1530
noun.location	2773	70.82	887
noun.attribute	2600	62.83	990
noun.time	2367	75.54	988
noun.state	1912	80.24	727
noun.body	1560	70.27	662
noun.quantity	1133	65.62	409
noun.possession	1127	69.23	209
noun.substance	1081	61.41	594
noun.event	1051	84.35	398
noun.object	905	72.19	300
noun.phenomenon	647	77.71	304
noun.animal	638	71.24	339
noun.relation	573	48.62	157
noun.feeling	481	64.82	163
noun.food	410	58.21	157
noun.plant	350	50.00	87
noun.process	317	68.65	119
noun.shape	219	81.98	53
noun.motive	107	83.39	19

TABLE 6 NUMBER OF INSTANCES OF NOUN SUPERSENSE IN TRAINING AND TEST SET ALONG WITH F1

6.1.2 FIRST SENSE NOT THE WINNER ALWAYS

Digging deeper into the test data, words like “reaction” were tagged as *noun.act* or *noun.phenomenon* (which is the most frequent sense) while the right supersense was *noun.process*. Similarly, for “air”, the tagger marked it as *noun.substance* which is the most frequent sense while the original label for it is *noun.location*.

6.2 CONTEXT SIZE ANALYSIS

Experiment 3 led to the conclusion that further away the word is (for larger context) the less likely there will be any semantic relation. Therefore context of current word +/-2 seems to be optimal.

6.3 WORD CLUSTER FEATURE BENEFITS

Although the improvements with addition of word cluster features did not result in a very high F1, there are many instances where the word cluster feature has helped while the baseline tagger failed. Some of the examples are:

Sports Writer Ensign Ritche of Ogden Standard Examiner went to his compartment to talk with him.

The Baseline Tagger and Tagger with word cluster feature using cluster of size 1024 labeled “Sports Writer” as:

	Baseline	With Word Cluster Feature
Sports	B-noun.act	B-noun.person
Writer	B-noun.person	I-noun.person

In cluster 81, “Sports Writer” is with other occupations like “chemist”, “cartographer”, “scholar”, “meteorologist” and many more. Another example:

*Jim Landis’ 380 foot **home run** over left in first inning...*

The tagger with word cluster features recognizes “home run” as *B-noun.act* and *I-noun.act* respectively. On the other hand the baseline missed out on “run” after tagging “home” as *B-noun.act*.

7 CONCLUSION AND FUTURE WORK

In this work, we highlighted how syntactic, contextual and word cluster features affect the performance of a system for tagging words with high level sense information. This project will help further research by suggesting areas to explore or not:

- We have demonstrated that lack of large annotated data is not a major issue. Nevertheless, this does not mean more annotated training data is not needed. But this suggests that a big project to annotate more data would likely be fruitful.
- The fact that previous label did not greatly affect the performance of the tagger seems to suggest that a sequence labeling approach is not necessary for good performance (as long as the constraint of proper BIO output is satisfied).
- Feature ablation methods like the ones described in the experiments help find out which features are important and hereby suggest areas to work (e.g.: new features to extend or add). Addition of word cluster and bigram features is an option to be considered.
- More research can be encouraged in creating word clusters using different techniques and of different granularities.

More importantly, it boils down to finding out which features are significant and when they should be used so as to achieve high performance standards in an entity recognizer or supersense tagger.

REFERENCES

- Borthwick, A. E. (1999). *A Maximum Entropy Approach to Named Entity Recognition*. New York University.
- Carreras, X., Màrquez, L., & Padró, L. (2002). Named Entity Extraction using AdaBoost, proceeding of the 6th Conference on Natural language learning. *August, 31*.
- Ciaramita, M., & Altun, Y. (2005). Named Entity Recognition in Novel Domains with External Lexical Knowledge. In *Proceedings of the NIPS Workshop on Advances in Structured Learning for Text and Speech Processing*.
- Ciaramita, M., & Altun, Y. (2006). Broad-coverage Sense Disambiguation and Information Extraction with a Supersense Sequence Tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Ciaramita, M., & Johnson, M. (2003). Supersense Tagging of Unknown Nouns in Wordnet. In *Proceedings of EMNLP (Vol. 3)*.
- Collins, M. (2002). Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of conference on EMNLP-Volume 10*.
- Curran, J. R. (2005). Supersense Tagging of Unknown Nouns Using Semantic Similarity. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*.
- Fellbaum, C., & others. (1998). *WordNet: An Electronic Lexical Database*. MIT press Cambridge, MA.
- Finkel, J. R., Grenager, T., & Manning, C. (2005). Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting of the ACL*.
- Florian, R., Ittycheriah, A., Jing, H., & Zhang, T. (2003). Named Entity Recognition through Classifier Combination. In *Proceedings of CoNLL-2003*.
- Francis, W. N., & Kucera, H. (1967). *Computational Analysis of Present-day American English*. Brown University Press Providence.
- Jiampojarn, S., Kondrak, G., & Cherry, C. (2009). Biomedical Named Entity Recognition Using Discriminative Training. *Recent Advances in Natural Language Processing V: Selected Papers from Recent Advances in Natural Language Processing 2007*.
- Koo, T., & Collins, M. (2005). Hidden-variable Models for Discriminative Reranking. In *Proceedings of the conference on Human Language Technology and EMNLP*. Vancouver, British Columbia, Canada.
- Lin, D., & Wu, X. (2009). Phrase Clustering for Discriminative Learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.
- Manning, C., & Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.

- Mikheev, A., Moens, M., & Grover, C. (1999). Named Entity Recognition without Gazetteers. In *Proceedings of EACL*.
- Miller, G. A., Leacock, C., Teng, R., & Bunker, R. T. (1993). A Semantic Concordance. In *Proceedings of the 3rd DARPA workshop on Human Language Technology*.
- Shipra, D., Malvina, N., Jenny, F., Christopher, M., & Claire, G. (1990). A System for Identifying Named Entities in Biomedical Text: How Results from Two Evaluations Reflect on Both the System and the Evaluations. *Comparative and Functional Genomics*.
- Zhou, G. D., & Su, J. (2002). Named Entity Recognition Using an HMM-based Chunk Tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*.