Using Wikipedia for Hierarchical Finer Categorization of Named Entities [⋆]

Aasish Pappu

Language Technologies Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
aasish@cs.cmu.edu

Abstract. Wikipedia is one of the largest growing structured resources on the Web and can be used as a training corpus in natural language processing applications. In this work, we present a method to categorize named entities under the hierarchical fine-grained categories provided by the Wikipedia taxonomy. Such a categorization can be further used to extract semantic relations among these named entities. More specifically, we examine instances of different kinds of Named Entities picked from Wikipedia articles categorized under 55 categories. We employ a Maximum Entropy based method to perform supervised learning that learns from local context of a named entity as well as a higher-level context such as hypernyms/hyponyms from Wikipedia and WordNet.

Keywords: hierarchical categorization, Named entity categorization, Wikipedia as corpus

1 Introduction

Ever-growing demand for classification of Named Entities is driven by open domain Question Answering and other search problems. The ability to perform fine-grained categorization of named entities is important due to the fact that it provides useful context information in Question Answering, Web Search and many other major Natural Language Processing (NLP) tasks. Very few works, however, aimed at broader and highly diverse categorization of named entities in open-domain. A popular named entity could be classified under a different class as the local context deems it.

A named entity categorization model requires a large-scale open-domain tagged corpus for training. Of late, large-scale resources have been a choice of corpus. Therefore, we believe that a more structured and organized encyclopedic corpus like Wikipedia (www.wikipedia.org) is a suitable training corpus mainly for two reasons. Firstly, Wikipedia contains text on a wide range of topics that is continually created and edited by volunteers. Secondly, the structure of Wiki provides hyperlinks over important phrases linked to other articles from Wikipedia besides links to redirected and disambiguation pages. Therefore, it offers a natural way to harvest a large volume of "labelled" training data. In fact, some emerging works tried to exploit Wikipedia as a knowledge resource (Ponzetto and Strube, 2006; Cucerzan, 2007).

In this paper, as a first step towards our task, we discuss the usability of Wikipedia as training corpus and different facets of Wikipedia. One of these facets is exploiting structure of Wikipedia to create training corpus for training for use with different machine learning algorithms. Further,

[⋆] We would like to thank Chua Tat-Seng for his valuable guidance throughout this work and also thank anonymous reviewers for their useful feedback.

another facet being hierarchical taxonomy of Wikipedia and the way we employed this facet in performing hierarchical categorization of named entities. Then, we present how to induce WordNet and Wikipedia domain taxonomy into the feature space of our classification model. We performed experiments with Maximum Entropy and SVM classifier and observed statistically significant improvement with Maximum Entropy model compared to SVM classifier.

2 Related Work

As the focus is shifted from coarse categorization of named entities to finer categorization of named entities latest works emerged and tackled the current problem in different perspectives.

Kazama and Torisawa (2007) used Wikipedia as external resource to boost the performance of an CRF-based NE tagger. They extracted gloss text found in Wikipedia articles, induced them in feature space and obtained improved results for NER on CoNLL 2003 dataset.

Dakka and Cucerzan (2008) presented a work on tagging the Wikipedia data with coarse named entity tags. They employed page-based and link based features for their classification task.

Bunescu and Pasca (2006) built a disambiguation system based on Wikipedia. Their approach focussed on building a dictionary of proper names, then mapping them to their named entity types.

These works exploited various features of Wikipedia, substantiating our claim about Wikipedia as one of the best training corpus. We contend that besides above features, there are more nuggets in Wikipedia such as category hierarchy which could be directly used to perform hierarchical fine-grained categorization of named entities.

3 Corpus Creation

We have used snapshot of 10-18-2007 english version of Wikipedia as corpus. This particular dataset contains about 2 million articles and 292,384 categories. According to the snapshot we have used, there are 5882 Wikipedia Stub categories across 105 domains[1] (hereafter we refer sections as domains), are arranged in a taxonomy with a depth about ten.

There are about 105 basic and specific domains in Wikipedia taxonomy under which categories are created. Particularly, domain-wise placement of categories in Wikipedia stub taxonomy is clearly visible due to its smaller size compared to taxonomy with main-stream categories. There are about 17 higher level domains in Wikipedia and there is a sub-tree under each of the higher level domains. Wikipedia domain structure is quite flat and its depth is about five.

Now, we discuss the process of carving out training corpus from Wikipedia. Firstly, we introduce the process of identifying categories under which named entity instances are categorized.

3.1 Categories in Wikipedia

Articles in Wikipedia are placed under highly specific to highly generic concepts called categories. These categories constitute the Wikipedia taxonomy linked to other categories across depth and breadth of the taxonomy. Since, most of the categories are linked across the breadth also, Wikipedia contains cycles through its taxonomy. This was also observed by (Zesch and Gurevych, 2007) and they proposed a method to tackle with cycles in Wikipedia. Therefore, Wikipedia taxonomy is not a tree unlike commonly found knowledge structures.

3.2 Named entity categories

There are 17 basic domains and 88 sub-domains in the domain hierarchy. For an instance, some of the domains are specific like *FANTASY*, *SCIENCE FICTION*, *HORROR* and some are basic such as *ECONOMIC AND FINANCE*, *GOVERNMENT*, *COMPANIES*. Particularly, there are 51 second level domains and 4 out 17 basic domains do not have sub domains. We want to have unbiased choice of categories from all domains and sub-domains in the taxonomy. We follow certain rules to choose set of categories to qualify as named entity categories. These rules are discussed below.

[1] `http://en.wikipedia.org/wiki/Wikipedia:WikiProject_Stub_sorting/List_of_stubs`

1. To ensure diversity in the categorization task, we picked up atleast one category per every second level domain including its sub domains or atleast one category per every basic domain that do not have second level domains.
2. To ensure we select balanced categories: depth, number of sub-categories and number of pages are defined as parameters to determine characteristic of a category as discussed in previous section.
3. We consider category with value for each parameter closest to mean value for that parameter under that domain, including its sub-domains.

This procedure avoids the bias towards any particular domain, although there is a skewed distribution among the classes due to the fact that there are fewer instances in some of them.

3.3 Procedure

We extracted named entity phrases using Stanford POS tagger (Toutanova and Manning, 2000)that is reported to achieve 86.91% accuracy on unseen words. Then we extract typed dependency relationships between named entities and other tokens in a sentence. To extract the content words around a named entity, we used chunked output from the parser to collect the NPs (noun phrases) and VPs (verb phrases).

There is a problem in considering all of the named entities in a sentence because we do not know category of every instance in a sentence, hence cannot be considered for validation purposes. Although, we know the category of the main named entity. We also like to consider as many instances as possible. Therefore, we employed a heuristic to deterimine the category of instances other than main named entity.

1. Firstly, we look for redirected and disambiguated article titles matching with first name of the named entity.
2. If, there are more than one such titles, consider the target title using minimum edit distance metric.
3. Pick all articles that fall under the same category as the target article.
4. Look for those articles that fall under the special categories that are chosen for the classification task.
5. Find the article that shares maximum number of categories with the target article and label the target article with the its special category.

There are about 450 such instances whose categories were not predetermined and their categories were determined by the above-mentioned heuristics.

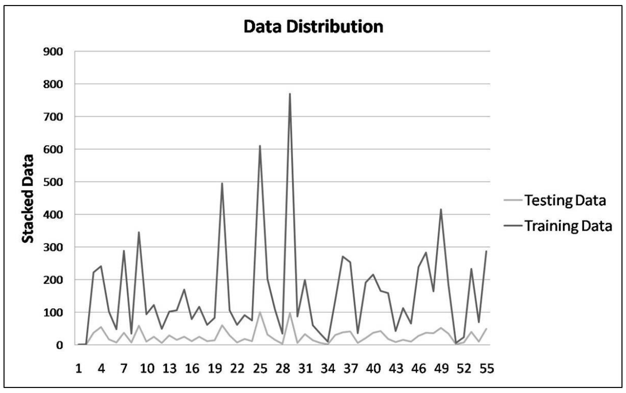

Figure 1: Data distribution

About 10,000 samples are divided into training and testing in 75% and 25% proportions respectively. Figure 1 shows the distribution of training data stacked over the distribution of the testing data. The major implication of this distribution is that number of training instances in a category is directly proportional to performance of system on a particular named entity category. Infact, we observed from the experimental results that 85% of the test results obey this implication.

4 Features

In this section, feature classes are discussed in detail. We employed four types of feature sets for our classification task. One of them is a syntactic feature set and the rest are semantic features. Since, we use a POS tagger to identify proper noun phrases in a sentence, lexical features are already handled by the POS tagger. It is necessary to supplement the system with syntactic and semantic features. We consider typed dependency relations to capture syntactic features in the context around a named entity instance. Regarding semantic features, we captured semantics in different perspectives of content words in the context around a named entity instance. Following are the semantic feature sets used in our model:

1. Hypernym of a content word.
2. WordNet domain of first synset of a content word.
3. Wikipedia domain of a content word.

We contend that hypernym of a word is more specific compared to a domain of a word. Therefore, we prefer to have a specific semantic feature and a generic semantic feature.

Features are discussed below :

4.1 Typed Dependency Feature

Typed dependencies and phrase structures are different ways of representing the structure of a sentence. While a phrase structure parse represents nesting of multi-word constituents, a dependency parse represents dependencies between individual words. A typed dependency parse additionally labels dependencies with grammatical relations, such as subject or indirect object.

Dependency relations give an additional clue about semantic relation between tokens in a sentence. As an example a sentence from an article in Wikipedia reads: *Chris Aldridge is a newsreader and continuity announcer on BBC Radio.* and one of the dependency relations of this sentence is *prep_on:BBC Radio*. The relation denotes the locativeness directed at the named entity *BBC Radio*. Clearly, it gives a clue about probable semantic relations that can be associated with the named entity.

4.2 Hypernyms

Limitations of a syntactic feature is that it is too generic, therefore we bank upon WordNet hypernyms as one of the semantic features pertaining to the content words in the context. We preferred to have a hypernym feature which is semantically specific. Therefore, hypernyms of all synsets are inversely ordered according to their depth in the hypernym tree. Then, deepest hypernym in the lot is choosen as the target feature for that content word.

We employed another criteria as a tie breaker if more than one hypernym have maximum depth. Then hypernym with least number of synsets as child nodes is chosen, once again to ensure specificity.

4.3 Domain based features

As discussed earlier, WordNet hypernym feature is specially incorporated as a specific natured feature in the feature space. To balance this act, we introduced two domain features each from different sources into feature space. One is based on WordNet based domain hierarchy and other is based on Wikipedia domain hierarchy discussed earlier.

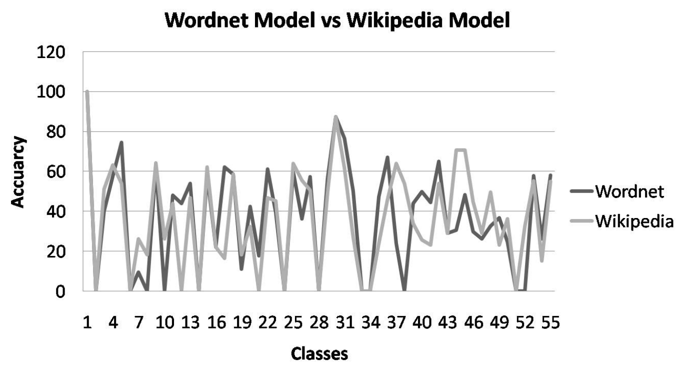

Figure 2: WordNet domain model vs wikipedia domain model

4.3.1 Wordnet domains Wordnet Domain Hierarchy (WDH) proposed by (Luisa *et al.*, 2004) is composed of 164 domain labels selected from dictionaries and codes in Dewey Decimal Classification (DDC), a widely used taxonomy for library organization purposes. WDH is interwined with Princeton WordNet application. Every synset in WordNet is associated a domain label in WDH. The domain labels are organized in five main trees of maximum depth of four.

There are five top-level domains and 46 basic domains in WDH. We incorporated this feature to evaluate the influence of domain-type semantic features on our system.

4.3.2 Wikipedia domains Similar to WordNet based domain system, we used the Wikipedia domain system, that has been discussed earlier sections. We indexed Wikipedia using Lucene and content words are searched in the index for the categories that contain more number of pages containing a content word. Especially, pages with links are weighed double the pages that contains the word without a hyperlink.

4.3.3 WDH vs Wikipedia Domain System We have evaluated each feature class' performance by building a model using each feature class individually. Figure 3 shows comparison among these features' (features discussed above) performance across all classes. We could see following observations from Figure 3:

- Only in four cases both of them failed to classify instances under the respective classes.
- When both of them reach their peak performance, they almost stand at same accuracy level.
- In 12 classes, both features perform equally, i.e. achieve equal accuracy. Whereas, in 23 classes, Wikipedia feature outperforms WDH feature. Overall, this observation shows that Wikipedia domain feature gets a slight edge over the WDH feature.

5 Experiments

We have chosen 75% training data and the rest is considered as testing data. L-BFGS(quasi-Newton optimization technique) parameter estimation has been chosen for training the Maximum Entropy Model with 20 iterations, the gaussian prior smoothing value set to 0.8. To avoid over-fitting while training, the tolerance value has been set to 2E-04. The log-likelihood convereged to -1.507336E+00 after 20 iterations and training accuracy was found to be 72.92%. There was no heldout data, but training has been performed for different set of parameters to check if there is any over-fitting while training. The difference between accuracies on training and testing data is minimum at the parameters mentioned above.

Additionally, we have tried out various combinations of features to generate the classification model to inspect the influence of each feature on the overall accuracy of the system. Please note that the accuracy reported in the results is 55-way classification.

We performed another set of experiments using multi-class LIBSVM classifier(Chang and Lin, 2001) with kernel set to radial basis function. Cost parameter and gamma value set to 2.0 and 0.125 respectively after performing cross-validation and grid-search. We observed SVM fails to do better than Maximum Entropy Model consistently on all kinds of models shown in Table 1. Also, the difference in the performance is also proportionately consistent as per the experimental results in Table 1.

Table 1: Results on combination of features

	Feature All		Feature 1,2,3		Feature 1,2,4		Feature 1,2		Feature 1,3		Feature 1,4	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
SVM	76.68	57.56	72.69	55.40	74.27	55.98	68.16	51.95	43.98	41.38	42.80	38.94
MaxEnt	72.92	60.79	69.17	58.78	71.25	58.57	68.22	58.13	49.01	47.28	48.38	45.41

From Table 1, we could interpret that system consistently improves on performance with both the machine learning methods, on adding additional features.

Please, note that for convenience we have used following labels to refer each feature in the analysis:

Table 2: Feature legend

Typed Dependency Feature	Feature 1
Hypernym Feature	Feature 2
WordNet Domain Feature	Feature 3
Wikipedia Domain Feature	Feature 4

Table 3: Results on individual features

	Feature 1	Feature 2	Feature 3	Feature 4
SVM	10.90	51.73	40.88	37.72
MaxEnt	10.61	58.64	44.90	43.75

5.1 Experiment 1: Feature wise model

We can infer from Table 3 that feature 1 individually doesnt look promising, attributes to its generic nature and proves that individually syntactic features cannot classify named entities. Yet, we have investigated its behavior with other features pair-wise to substantiate its presence improves overall performance.

Feature 2 individually looks impressive, almost close to the all-feature model with MaxEnt method but not close enough to the all-feature model with SVM classifier. Therefore, we have anticipated an improvement in the performance, coupled Feature 2 with Feature 1. We observed improvement on the individual feature 2 model with SVM classifier, but a slight decline in performance with MaxEnt method. Possibly, Feature 2 reaches its peak performance individually at 58.64 with MaxEnt.

Feature 3 and Feature 4 are of similar nature they are extracted from different sources. At times, both features behaved in a complementary manner and improving overall performance. Feature 3 and 4 individually behaves similarly by achieving accuracy about 44%.

5.2 Experiment 2: Feature combination model

From Table 1 we can observe that by adding Feature 1 to each model, they have achieved improvement with both of the machine learning methods, proving to be useful feature in spite of its poor individual performance.

Since, we realize that Feature 3 and Feature 4 behave similarly and best pair of features are 1 and 2 together. We tried out the following combinations of features as models: 1, 2, 3 and 1, 2, 4.

Interestingly, their performance on test set found to be similar. Once again proving that Feature 3 and 4 are similar in nature with respect to performance. Finally, we created a model using all features and obtained better performance than all earlier models. There is 1.68% improvement over 2nd best model with SVM classifier, whereas there is 2.01% improvement over 2nd best model with MaxEnt method. Over, class-level accuracy we have examined how significant is the improvement of Maximum Entropy Model over SVM. Therefore, we have performed t-test with a 95% confidence interval and found that p value equals 0.0147, which is considered statistically significant.

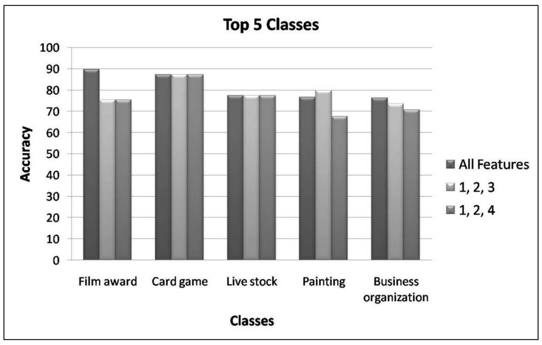

Figure 3: Top five classes

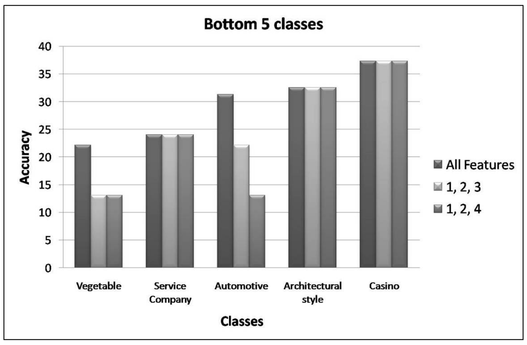

Figure 4: Bottom five classes

From Figure 3 we could see clear domination of All-Feature model over other models across the top 5 classes with respect to accuracy.

About the first ranked class, the 1,2,3 model and 1,2,4 model performs equally with 75.42%. Thus, giving a good discrimination after combining all features performing with 89.71% accuracy.

Whereas, there is no change with respect to 2nd and 3rd ranked class in the performance of the three models.

However, 4th ranked class shows interesting pattern, (1,2,4) model performs with 67.63% accuracy, on the other hand (1,2,3) model performs with 79.75% accuracy. There is major difference in the accuracy, this is one of the very few cases in our experiments when (1,2,3) model outperforms (1,2,4) by large margin. Each time, one of the (1,2,3), (1,2,4) models outperformed other by larger margin, it reflected as slight decline in the all-feature models performance compared to the best accuracy in that class. This observation leads to an interesting fact that not all content words in the context present in Wikipedia stub categories.

Whereas, according to Figure 4 most of the bottom classes have all-feature models with best performance. There is steady rise in the performance in other models also, although we have ranked the classes according to their all-feature model's accuracy.

5.3 Experiment 3: Error analysis

The arrow mark in the figure below (Figure 5) denotes mean error against the all-feature model. The minimum and maximum error are the lowest and highest points of the line corresponding to each model.

All-Feature Model under-performs in certain cases compared to other models. The figure depicts the error against each model. A soaring error against the 1, 3 model is the maximum error made by all-feature model. After inspecting that class, we understood that other features in the samples under that class failed to classify Named Entities. Overall, only Feature 3 successfully classified Named Entities under that class with 44%. It seems a rare instance of complete failure, since we could see that overall mean error is about 10%.

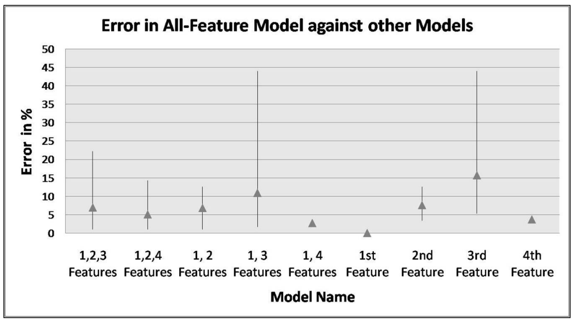

Figure 5: Error analysis

6 Conclusion

We have presented a named entity categorization system that employs Wikipedia categories as classes and exploits the vastness and wide variety of data present in Wikipedia. We adapted hierachial categorization of Wikipedia, thereby giving scope to mine relations among named entities. Furthermore, Wikipedia domain feature set leads to idea of topic modeling using various facets in Wikipedia. Our system explores the potency of Wikipedia as full-fledge open-domain corpus and substantiates its usefulness with the accuracies obtained over diverse classes.

References

Bunescu, R. and M. Pasca. 2006. Using Encyclopedic Knowledge for Named Entity Disambiguation. *Proceedings of Conference of the European Chapter of the Association for Computational Linguistics*, pp.9-16.

Chang, C. and C. Lin. 2001. LIBSVM: a library for support vector machines. Software available at `http://www.csie.ntu.edu.tw/˜cjlin/libsvm`.

Cucerzan, Silviu. 2007. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. *Proceedings of the 2007 Joint Conference on EMNLP and CoNLL*, pp.708-716.

Dakka, W. and S. Cucerzan. 2008. Augmenting Wikipedia with Named Entity Tags. *Proceedings of the IJCNLP*.

Kazama, J. and K. Torisawa. 2007. Exploiting Wikipedia as External Knowledge for Named Entity Recognition. *Proceedings of EMNLP and CoNLL 2007*, pp.698-707.

Luisa, B., F. Palmer, B. Magnini and E. Pianta. 2004. Revising WordNet Domains Hierarchy: Semantics, Coverage and Balancing. *COLING 2004 Workshop on Multilingual Linguistic Resources*, pp.101-108.

Ponzetto, S.P. and M. Strube. 2006. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. *Proceedings of the NAACL 2006.*.

Toutanova, K. and C. Manning. 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. *Proceedings of EMNLPNLC-2000*, pp.63-70.

Zesch, T. and I. Gurevych. 2007. Analysis of the Wikipedia Category Graph for NLP Applications. *Proceedings of TextGraphs-2: Graph based Algorithms for NLP*, pp.1-8.