UNIBA: Super-sense Tagging at EVALITA 2011

Pierpaolo Basile

Dept. of Computer Science - University of Bari "Aldo Moro"
Via Orabona, 4 - 70125 - Bari (ITALY)
basilepp@di.uniba.it

Abstract. This paper describes our participation in EVALITA 2011 Super Sense Tagging (SST) task. The goal of the task is to annotate each word in a text within a general semantic taxonomy defined by the WordNet lexicographer classes called super-senses. In this task, we exploit structured learning based on Support Vector Machine. Moreover, we propose to solve the data sparseness problem by incorporating features provided by a semantic *WordSpace* built exploiting the distributional nature of words.

Keywords: Super-sense Tagging, Structured Learning, Distributional Approaches, Support Vector Machine

1 Motivation and Systems Description

Super-sense tagging is the task of annotating each word in a text with a concept coming from a general semantic taxonomy defined by lexicographer classes called super-senses. A super-sense defines a general concept such as animal, body, person, communication, motion. Super-sense tagging can be considered as an half-way task between Named Entity Recognition (NER) [3] and Word Sense Disambiguation (WSD) [6]. In the former a small set of categories is involved, for example: Person, Organization, Location, Time. In the latter a very large set of senses with very specific meanings is taken into account. Super-senses are not strictly related to proper nouns as named entity classes and provide a more abstract set of meanings which simplifies the problem of sense disambiguation. Super-sense tagging combines, in a such way, a small-sized set of categories typical of NER with meanings provided by super-senses. We can consider Super-sense tagging a simpler version of WSD, where a smaller number of meanings is involved.

The small set of senses allows to use robust supervised learning approaches, such as sequence labelling methods trained on a hand-tagged corpus. However, structured learning is subject to the data-sparseness problem. This side effect is more evident when lexical features are involved (like in super-sense task), because test data can contain words with low frequency (or absent) in training data.

In this paper, all proposed systems rely on supervised methods for super-sense tagging based on Support Vector Machines (SVM) a state-of-art machine

learning algorithm. Moreover, we propose to solve the data sparseness problem by incorporating features provided by a semantic *WordSpace* built exploiting the distributional nature of words. The core idea behind the *WordSpace* is that words and concepts are represented by points in a mathematical space, and this representation is learned from text in such a way that concepts with similar or related meanings are near to one another in that space (geometric metaphor of meaning). According to the *distributional hypothesis* [4], the meaning of a word is determined by the rules of its usage in the context of ordinary and concrete language behaviour. Hence, words are semantically similar if they share *contexts*.

The main idea of our work is to improve the robustness of our super-sense tagging approach by extending lexical information through distributional analysis. Using distributional analysis we expect that in semantic spaces, words with similar meaning are represented close in *WordSpace*. We can rely on this property to solve the problem of data-sparseness by adding distributional information about words as features into the structured learning strategy.

1.1 *WordSpace* Building

We use the SemanticVectors package[1] [7], an opensource tool, to build the *WordSpace*. SemanticVectors creates semantic *WordSpace* from free natural language text using the Random Indexing (RI) technique. RI is based on the concept of Random Projection [1]. Specifically, RI creates the *WordSpace* in two steps:

1. a random vector is assigned to each context. This vector is sparse, high-dimensional and ternary, which means that its elements can take values in {-1, 0, 1}. The random vector contains a small number of randomly distributed non-zero elements (seeds), and the structure of this vector follows the hypothesis behind the concept of Random Projection;
2. random vectors are accumulated incrementally by analyzing contexts in which terms occur. In particular, the semantic vector assigned to each word is the sum of the random vectors of the contexts in which the term occur. It should be pointed out that random vectors are added by multiplying them by the term frequency.

In particular, we exploit RI to build two different spaces using two different definitions of context:

1. Wikipedia pages: a random vector is assigned to each Wikipedia page;
2. Wikipedia categories: the idea is that categories can identify more general concepts in the same way of super-senses. In this case, for each category a random vector is created.

Before the building of *WordSpace*s, we need to index all Wikipedia pages using the last dump provided by Wikipedia foundation. During the indexing step we extract page categories using a regular expression[2] and add these as

[1] Available on-line: http://code.google.com/p/semanticvectors/
[2] Categories are defined in the page using Mediawiki syntax.

meta-data to each page. After this first indexing step, we build a second index containing a document for each category. That index is necessary to build the *WordSpace* which relies on Wikipedia category as context. We use Apache Lucene[3], an open-source API, for indexing.

Finally, we run SemanticVectors tool on each index, obtaining as result the two *WordSpace*s. Table 1 reports information about *WordSpace*s, in particular: the number of contexts (pages or categories) C, the space dimension D and the number of no-zero elements (seeds) in random vectors S.

Table 1. WordSpaces info

$WordSpace$	C	D	S
Wikipedia pages	1,617,449	4,000	10
Wikipedia categories	98,881	1,000	10

1.2 Learning Strategy and Features Description

We use a set of lexical/morphological and contextual features to represent each word w in the training data, in particular:

1. the word w plus contextual words: w_{-1} and w_{+1} (the first word to the left and the first word to the right);
2. the lemma of the word l_w plus contextual lemmas: l_{w-1} and l_{w+1};
3. the part-of-speech (PoS) tag of the word pos_w plus contextual PoS-tags: pos_{w-1} and pos_{w+1};
4. the super-sense assigned to the most frequent sense of the word w. The most frequent sense is computed according to sense frequency in MultiSemCor;
5. the first letter of the PoS-tag, generally it identifies the word-class: noun, verb, adjective and adverb;
6. a binary feature that indicates if the word starts with an upper-case character;
7. the grammatical conjugation of the word w (e.g. -are, -ere and -ire for Italian verbs);
8. distributional features: words in the *WordSpace* are represented by high-dimensional vectors. We use as features all the components in the word vector $\overrightarrow{w}$.

As learning method we adopt Support Vector Machines (SVM). In particular, we propose four systems:

1. **uniba_SST_Closed_yc** is based on SVM using only features provided by organizers (excluding the features 4, 5, 6, 7, 8). We use an open-source tool *YAMCHA* [5] that is a generic text chunker based on SVM adopted in several NLP tasks such as PoS tagging, named entity recognition and phrase chunking.

[3] Lucene is available on-line: lucene.apache.org.

2. **uniba_SST_Open_yo** works like the first system and uses all the features excluding the feature 8 (distributional information).
3. **uniba_SST_Open_SVMcat** relies on distributional information. Distributional features are numerics and cannot be represented in *YAMCHA* which manages only discrete values. Moreover, distributional features are represented by high-dimensional vectors. For these reasons we adopt *LIBLINEAR* [2], a library for large linear classification. *LIBLINEAR* provides good results where a large number of features is involved using a linear mapping instead of non-linear kernels such as polynomial kernels adopted, for example, by *YAMCHA*. LIBLINEAR implements linear support vector machines that are very efficient on large sparse datasets. This system exploits the *WordSpace* built using Wikipedia category.
4. **uniba_SST_Open_SVMterm** works like the previous system but it relies on the *WordSpace* built on Wikipedia pages.

2 Evaluation

The dataset used to perform the training step is provided by organizers. The training corpus consists in about 276,000 word forms divided into 11,342 sentences and 430 documents. Words are tagged with their super-sense in IOB2 format: B for the word at the begin of the annotation, I for inside and O for outside words. IOB2 schema allows to annotate multi-words expressions. The training contains also information about PoS-tag and lemma. The IOB2 format can be used by *YAMCHA* without any transformation, while to build the system based on *LIBLINEAR* we need to transform the IOB2 dataset into *LIBLINEAR* data-format which requires a line for each example (word). Each line contains the $class_{id}$, in our case the tag assigned to the word, and the list of features as follows: $feature_{id} : feature_{value}$. *LIBLINEAR* requires that each data instance (example) is represented as a vector of real numbers. For each value assumed by no-numeric feature, a $feature_{id}$ is generated. In this case, the $feature_{value}$ can assume only two values: 1 if the $feature_{id}$ occurs in the data instance otherwise 0. It is important to underline that also SVM implemented in *YAMCHA* requires only numeric features, but the transformation from IOB2 format is automatically made by *YAMCHA*.

The test data provided by organizers consists in about 44,900 words divided into 64 documents. The testing data are in the same format of training data without information about super-sense. For the evaluation, in order to compare the systems output against the super-sense labels in gold data, the organizers supply a script which provides information about accuracy, precision, recall and F-measure for each super-sense type.

2.1 Results

The task plans two different subtask. In the first one, called CLOSED, only the corpus provided for training can be used by participants. In the second one,

called OPEN, participants can exploit any external resources in addition to the training data. Table 2 reports the results obtained by all the participants in the CLOSED subtask, while Table 3 shows results about OPEN subtask. *System 1* is the system provided by the other participant.

Table 2. Results for CLOSE evaluation

System	*A*	*P*	*R*	*F*
System 1	0.8850	0.7682	0.7976	0.7827
System 1	0.8834	0.7669	0.7938	0.7801
System 1	0.8830	0.7664	0.7933	0.7796
System 1	0.8827	0.7648	0.7929	0.7786
uniba_SST_Closed_yc	0.8696	0.7485	0.7583	0.7534

Table 3. Results for OPEN evaluation

System	*A*	*P*	*R*	*F*
uniba_SST_Open_SVMcat	0.8877	0.7719	0.8020	0.7866
uniba_SST_Open_SVMterm	0.8864	0.7700	0.7998	0.7846
uniba_SST_Open_yo	0.8822	0.7728	0.7818	0.7773

Our system in the CLOSED subtask is not able to outperform the *System 1*, while in OPEN subtask the methods which exploit distributional features are able to outperform *System 1*. It is important to point out that we are the only participant in the OPEN subtask. Moreover, we do not know the strategy and the features exploited by the *System 1*. This makes impossible to provide some discussion about the difference between our system and *System 1*.

Taking into account only our systems, the focus of our discussion is twofold: (1) prove that distributional information are able to improve the performance of a structured learning strategy; (2) compare the two *WordSpaces*: Wikipedia pages and Wikipedia categories.

Regarding the first point, results show that methods based on distributional features are able to outperforms all the other systems. This is a very important outcome as it has been shown that distributional methods are able to deal with the data-sparseness problem as highlighted by improvements in recall values. Taking into account the second point, results in Table 3 show that the space built on Wikipedia categories (Open_SVMcat) provides better results, even though they are not significant.

Moreover, Table 4 reports results of our systems focusing on a subset of the evaluation. The table takes into account the five most frequent nouns and verbs super-sense. Table 4 reports the performance in terms of F-measure for each system; the best result is reported in bold face.

Table 4. Results considering the most five frequent super-senses

EVALITA 2011 SST					
NOUNS					
Super-sense	#n	Closed_yc	Open_SVMcat	Open_SVMterm	Open_yo
noun.act	2117	77.54	**83.99**	83.98	80.87
noun.possession	1525	72.54	77.44	**78.29**	73.48
noun.communication	1498	71.79	70.82	70.34	**74.17**
noun.group	1116	65.22	60.09	60.73	**66.11**
noun.artifact	1008	55.93	**63.79**	62.66	57.68
VERBS					
Super-sense	#n	Closed_yc	Open_SVMcat	Open_SVMterm	Open_yo
verb.stative	790	86.30	88.82	**88.96**	87.02
verb.communication	682	78.05	**85.39**	84.63	81.38
verb.change	583	68.85	81.70	**82.99**	75.04
verb.social	346	69.83	**78.43**	78.30	75.73
verb.cognition	331	74.76	**83.11**	82.96	79.76

Results show that methods based on distributional features achieve always the best performance with the exception of *noun.communication* and *noun.group*. In these two cases, the method based on *YAMCHA* obtains the best performance. Moreover, in these cases also the method based on only features provided by organizers (Closed_yc) outperforms the ones based on distributional features. Indeed, it seems that distributional features introduce some noise in training for these super-senses.

References

1. Dasgupta, S., Gupta, A.: An elementary proof of a theorem of Johnson and Lindenstrauss. Random Structures & Algorithms 22(1), 60–65 (2003)
2. Fan, R., Chang, K., Hsieh, C., Wang, X., Lin, C.: Liblinear: A library for large linear classification. The Journal of Machine Learning Research 9, 1871–1874 (2008)
3. Grishman, R., Sundheim, B.: Message understanding conference-6: a brief history. In: Proceedings of the 16th conference on Computational linguistics - Vol. 1. pp. 466–471. COLING '96, Association for Computational Linguistics, Stroudsburg, PA, USA (1996), http://dx.doi.org/10.3115/992628.992709
4. Harris, Z.: Mathematical Structures of Language. New York: Interscience (1968)
5. Kudo, T., Matsumoto, Y.: Fast methods for kernel-based text analysis. In: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics. pp. 24–31. Association for Computational Linguistics, Sapporo, Japan (July 2003), http://www.aclweb.org/anthology/P03-1004
6. Navigli, R.: Word sense disambiguation: A survey. ACM Comput. Surv. 41, 10:1–10:69 (February 2009), http://doi.acm.org/10.1145/1459352.1459355
7. Widdows, D., Ferraro, K.: Semantic Vectors: A Scalable Open Source Package and Online Technology Management Application. In: Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008). pp. 1183–1190 (2008)