Supervised TextRank

Fermín Cruz, José A. Troyano, and Fernando Enríquez

Department of Languages and Computer Systems
University of Seville
Av. Reina Mercedes s/n 41012, Sevilla Spain
troyano@lsi.us.es

Abstract. In this paper we investigate how to adapt the TextRank method to make it work in a supervised way. TextRank is a graph based method that applies the ideas of the ranking algorithm used in Google (PageRank) to Natural Language Processing (NLP) tasks. This approach has given very good results in many NLP tasks like text summarization, keyword extraction or word sense disambiguation. In all these tasks TextRank operates in an unsupervised way, without using any training corpus. Our main contribution is the definition of a method that allows to apply TextRank to a graph that includes information generated from a training tagged corpus. We have tested our method with the Part of Speech (POS) tagging task, comparing the results with those obtained with tools specialized in this task. The performance of our system is quite near to these tools, improving the results of two of them when the corpus tagset is big and therefore the tagging task more complicated.

1 Introduction

Graphs are a very natural representation for many NLP problems. In fact, we have a graph just splitting a text into words and linking them by means of some syntactic or semantic relationship. However, this obvious relationship between texts and graphs is not always present in the models employed to implement NLP applications. For example, generative approaches based on grammars tend to use trees as representation model as a natural consequence of derivation trees.

In the other hand, statistical methods (based on corpus) rely on a great variety of representations but only a few make use of the relationship between graphs and language. Techniques like Maximun Entropy Modelling, Decision Trees, Memory Based Learning or Transformation Based Learning are quite far of including graph representations in their models. Examples of graph based techniques are Markov Models and Neural Networks, though in this cases graphs are not used to represent texts and they just give a way of connecting various elements to build a model.

Recently, there have appeared research works that begin to make use of graphs as the central representation for their models. There are even workshops

⋆ Partially funded by Spanish Ministry of Science and Technology (TIN2004-07246-C03-03).

(like [9]) whose main subject is the use of general graph methods and algorithms for text processing tasks.

TextRank [6] is one of these approaches. This algorithm is based on the same idea used originally by Google [4] to calculate the relevance of each web page in Internet. It has been successfully applied to several NLP tasks. Despite of being an unsupervised method it reaches similar results in these tasks than systems that make use of additional information through annotated training corpora.

In this paper we investigate how to use TextRank in a supervised way. To do that, we have collected information from a tagged training corpus and we have included this information into a graph that is subsequently processed by the TextRank algorithm. Our intuition says that if TextRank has behaved so good working without training material, it would work better if we include in the graph information extracted from thousands of examples of a task. The key is to find the graph representation for a given problem that best exploits the power of TextRank.

We have defined a general method for constructing a graph from a tagged training corpus. This method is independent of the corpus tagset, so it can be applied to any task that attachs tags to words. We have chosen the POS tagging task for our experiments because it is easy to find resources to train the models and because there are many specialized tagger generators to compare with. We are aware that POS tagging is a well studied problem and that it is quite difficult to improve the results reached by well tested techniques like Markov Models [2] or Transformation Based Learning [3]. However, our aim is to learn from these initial experiments in order to apply these ideas to more complex tasks in the future.

The organization of the paper is as follows. In section two we present the original version of TextRank, in the third section we show how to build a graph from a training tagged corpus, fourth section includes the experimental design and the results. Finally, in section five we draw the final conclusions and point out some future work.

2 TextRank Algorithm

The main idea of TextRank is to apply a graph based ranking algorithm to NLP tasks. It uses the well known PageRank algorithm [4], one of the keys that converted Google in one of the most used browsers in Internet. PageRank provides a web page ranking that relies on the knowledge stored in web page links. It is used to calculate a relevance indicator for each page in Internet that allows Google to decide which pages would be more interesting given a user query. This idea has been successfully used in other domains, like social nets analysis or citation analysis.

Formalization of PageRank is quite easy, let $G = (V, E)$ be a graph where V is a set of vertices and E is a a set of directed edges between two vertices. Two functions are defined for a given vertex V_i:

– $In(V_i)$ calculates the set of vertices that point to V_i.
– $Out(V_i)$ calculates the set of vertices that V_i points to.

The score of a vertex V_i is computed by the following formula from In and Out operations:

$$PR(V_i) = (1 - d) + d \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} PR(V_j)$$

where d is a dumping factor that is used to include in the model the probability of a random jump from a vertex to any other, not necessarily linked to the first one. The formula models the behavior of an Internet user that chooses randomly a link with probability d and visit a completely new page with probability $1 - d$. In the original definition of PageRank a value of 0.85 is recommended for this factor d, we have also used this value in our experiments.

An iterative algorithm is used to compute the PageRank value of each vertex of the graph. This algorithm initially assigns arbitrary values to each node and then applies iteratively the formula until convergence. This convergence is achieved when the difference of the PageRank values in two consecutive iterations is less than a predefined threshold for all the vertices in the graph. Once the iteration has finished, the value calculated for each vertex represents the importance that the algorithm has associated it.

This formula can be easily extended to admit weighted graphs. In this case the score is computed using the following formula, where p_{ji} is the weight of the edge that goes from vertex V_i to V_j:

$$WPR(V_i) = (1 - d) + d \sum_{j \in In(V_i)} \frac{p_{ji}}{\sum_{k \in Out(V_j)} p_{jk}} WPR(V_j)$$

TextRank applies the ideas of PageRank to NLP tasks. To do that it is necessary to find a way of representing the task by means of a graph. Then, PageRank can be run and the resulting scores of each nodes can be used to make decisions about the textual entities that they represent. The authors of TextRank have successfully applied it to several NLP tasks, including Keyword Extraction and Text Summarization [6] or Word Sense Disambiguation [7]. In each task the method for building the graph is different. For example, in Keyword Extraction vertices denote words, and edges represent that two words appear close in a phrase.

3 Building the Graph from an Annotated Corpus

Until now, applications developed using TextRank have followed a non supervised approach. That is, the graph is built directly from the test corpus avoiding the use of any annotated training corpus. Despite of it, TextRank achieves results comparable to supervised learning systems that use annotated corpus in the three tasks mentioned earlier. Perhaps, the reason for such an unexpected fact (the same results are achieved using less information) may be found in the nature of the selected tasks, that fit very well to graph models.

So, how good a task fits to graph models seems to be a critical factor for using TextRank to solve it. In fact, we have not found any application to other classic tasks of NLP such as POS tagging, syntactic analysis or information extraction. The goal of this work is indeed to explore other application targets for this algorithm while trying to adapt its use to a supervised framework that takes advantage of the information available in a training annotated corpus.

The first thing we have to do is to decide a graph representation of our problem from all the possible ones, in order to apply a ranking algorithm as TextRank. We have chosen a representation as general as possible, so it could be used to any tagging task. Vertices of our graphs are composed by two information units, a word w and a tag t ($V = (w, t)$). For an ambiguous word (several tags can be associated to it), as many vertices as possible tags to the word are created. The main idea of our approach is building a graph for each sentence to be tagged, applying TextRank to each of them and assigning to each word the tag from its best ranked vertex. If a word appears more that once in a sentence, independent vertices are created for each of them, this way it is possible to assign different tags to each instance of the repeated word.

Edges in our graphs represent word cooccurrence, so between two vertexes $V_i = (w_i, t_i)$ and $V_j = (w_j, t_j)$ there is an edge if the word w_j appears in the sentence just after the word w_i.

Finally, information extracted from the training corpus appears in the graph as the weights of the edges. We have tested a few metrics to this purpose and the best results have been achieved using a combination between emission probabilities $P(w|t)$ and transition probabilities $P(t|t')$, the same ones used by the bigrams based Hidden Markov Models. These probabilities are estimated counting words and tags in the corpus:

$$P(w|t) = \frac{C(w,t)}{C(t)} \qquad\qquad P(t|t') = \frac{C(t',t)}{C(t')}$$

where $C(t)$ is the count of the tag t in the training corpus, $C(w|t)$ is the count of the word w tagged with tag t and $C(t', t)$ is the count of the tag t' appearing just before the tag t.

In the Hidden Markov Models, these probabilities are used to compute the best tagged sentence maximizing this probability:

$$P(t_{1,n}|w_{1,n}) = \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

We use the probability $P(w_i|t_i)P(t_i|t_{i-1})$ to weigh the edge going from vertex $V_{i-1} = (w_{i-1}, t_{i-1})$ to vertex $V_i = (w_i, t_i)$, and then we let TextRank to compute the importance of each node. Unlike Markov Models which consider all possible solution paths of the graph as competitors, searching for the one maximizing the earlier expression, our TextRank based approach is more collaborative, because probabilities from different edges are combined in order to compute a score for each vertex.

Many of the classic improvements of Markov Models, like trigrams and unigrams computing, unknown words estimations, or interpolation, may be easily added to our system, just redefining the expression that weights the edges.

4 Experimental Design and Results

In this section we present the results we have obtained by applying different variants of the supervised TextRank with two corpus annotated with POS tags. There are many resources for this task and there are also many approaches which we can compare the results to. This task, like many others in PLN, consists of deciding which tags must be associated to a word. The set of labels is usually medium size (between 50 and 100), there are words that can only be tagged with one tag and others for which there are several possibilities. The hardest problem in this task is raised by the unknown words, that are those that previously have not been observed in the training corpus.

We have compared our results with the ones obtained with the most used tools for the POS tagging. Both corpus used are written in English, one is the Susanne corpus and the other one is made up of the four first sections of the Penn TreeBank corpus, in table 1 there can be seen the sizes of the train and test partitions for both corpus.

Table 1. Sizes of the corpus

	Words (train)	Words (test)	Tags
Susanne Corpus	141140	15482	131
Penn Corpus	198550	46461	35

The most significant difference between both corpus is the number of tags. The Penn corpus has a quite small set of tags of only 35 tags, whereas the Susanne corpus triples that number with 131. In practice this can be translated in the fact that tagging using the Susanne corpus is a much more difficult task than with the Penn corpus, as it is verified in the results.

4.1 Other Systems

In order to compare the results obtained with our supervised version of TextRank we have trained both corpus with tools specialized in the task of POS tagging. Concretely we have used the following systems:

– **TnT** [2], is one of the most widely used, based in Markov Models, is very fast and usually obtains very good results.
– **TreeTagger** [10], is based in decision trees, it generates a database register for each word that is later used to obtain the decision tree.
– **MBT** [5], carries out the training by means of example based learning, an efficient implementation of the nearest neighbour technique.

- **fnTBL** [8], is an efficient implementation of the Brill method based in the generation of transformation rules guided by error discovery.
- **MaxEnt** [1], uses maximum entropy modelling to integrate, using restrictions, the problem knowledge provided by the training corpus.

Furthermore, we have a simple tagger that we have used as the baseline in our experiments. This simple tagger associates to each word the most repeated tag in the training corpus for it.

4.2 TextRank Variants

Besides the supervised method for TextRank presented in section 3 of this article, we have included in our group of experiments two variants of the original idea.

The first variant, namely inverse TextRank, consists in calculating the transition probabilities in reverse way. Therefore, $P(t|t')$ shows the probability of tag t' being found after the tag t, and it is estimated with the following formula from the examples of the training corpus:

$$P(t|t') = \frac{C(t,t')}{C(t')}$$

In this model, the edges of the graph represent the coocurrence relations of the viewed words from right to left.

The second variant consists of the combination of the results of TextRank and inverse TextRank. In order to do this we have applied the technique of stacking, that consists of using the results of a first stage of learning to train a second level classifier. In our case, for each word of the training corpus we have created a register that contains the three tags better located according to TextRank and inverse TextRank for this word, as well as the scores obtained by each proposal. The register is completed with the real tag that is extracted directly from the training corpus. With all the registers obtained from the training corpus, we trained a decision tree that determines the tag to assign to a word based on the proposals of TextRank and inverse TextRank.

4.3 Results

Before running these experiments we knew that it was going to be difficult to obtain better results than the tools with which we would compare ourselves. The reason: these tools are very specialized in the task of the POS tagging and include special heuristics to solve this problem in the best possible way.

The table 2 shows the results of all the systems described in this article. With respect to the baseline, TextRank fully surpasses the established ones for both corpus. Considering the comparative with the other tools, our method beats TreeTagger and MBT with the Susanne corpus, and remains quite close to the rest of taggers for this corpus. In the case of the Penn corpus we did not beat any of these tools, although our results are near to MBT and TreeTagger.

Table 2. Results of the experiments

	Susanne Corpus	Penn Corpus
Baseline	79.15%	80.01%
TnT	93.61%	95.48%
TreeTagger	91.27%	94.28%
fnTBL	93.01%	95.04%
MBT	91.16%	94.40%
MaxEnt	93.09%	95.47%
TextRank	90.32%	92.14%
Inverse TextRank	89.84%	91.70%
Combined TextRank	91.51%	93.28%

Although we have obtained worse results than most of the tools, it is necessary to emphasize that our method is still in a very preliminary phase. For example, no special heuristic for the unknown words is included. It is to expect that the results will improve when this type of information is included in the construction of the graph, as most of the tools which we have compared it with includes this information. It is also necessary to emphasize that the method has behaved better with the most difficult corpus (Susanne with 131 tags as opposed to the 35 tags of the Penn corpus), which makes us think that it can work better in another type of harder tasks.

As an interpretation of these results we can conclude that the supervised TextRank method can be an alternative to other learning methods used in NLP tasks, although it is still necessary to make a deeper study to know how far it can improve and in what tasks it can give better results.

5 Conclusions and Future Work

We have studied how to adapt the unsupervised algorithm TextRank to make it work in a supervised way. We have defined a method to build a graph from a phrase that integrates information extracted from a tagged training corpus. We apply TextRank to this graph and use the ranking of each node to assign the most plausible tag to each word of the phrase. The method is based on word coocurrence and emission and transition probabilities similar to those used in Markov Models.

We have developed an experimental study using two corpus (Susanne and Penn TreeBank) tagged with POS information, and we have compared our results with those obtained by specialized tools for this kind of task. Results show that our method produces a satisfactory tagging, improving the results of two of them when we train with the Susanne corpus (this corpus sets out a more difficult problem than Penn corpus because its tagset is bigger).

Our future work will concern three main lines: 1) to study different ways of building graphs that include more information extracted from the corpus, 2) to define heuristics to manage special cases like unknown words, and 3) to apply our method to other NLP tasks like Information Extraction or Shallow Parsing to check out how it behaves with more complex tasks than POS tagging.

References

[1] Baldridge, J., Morton, T., Bierner, G.: Maxent, Mature Java package for training and using maximum entropy models. An *OpenNLP* project. (2005)

[2] Brants, T.: TnT. A statistical part-of-speech tagger. In *Proceedings of the 6th Applied NLP Conference (ANLP00)*. USA (2000)

[3] Brill, E.: Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. In *Computational Linguistics* 21(4), (1995)

[4] Brin, S., Page, L.: The anatomy of a large-scale hypertextual Web search engine. In *Computer Networks and ISDN Systems*. (1998)

[5] Daelemans, W., Zavrel, J., van der Sloot, K., van den Bosch, A.: TiMBL Memory Based Learner, version 5.1, Reference Guide. ILK Research Group Technical Report Series no. 04-02. The Netherlands (2004)

[6] Mihalcea, R., Tarau, P.: TextRank. Bringing Order into Texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Barcelona, Spain (2004)

[7] Mihalcea, R., Tarau, P. Figa, E.: PageRank on Semantic Networks, with application to Word Sense Disambiguation. In *Proceedings of The 20st International Conference on Computational Linguistics* . Switzerland, Geneva (2004)

[8] Ngai, G., Florian, R.: Transformation-based learning in the fast lane. In *Proceedings of North Americal ACL 2001*, (2001)

[9] Radev, D., Mihalcea, R. (organizers): Graph-based Algorithms for Natural Language Processing. Workshop at *HLT/NAACL*. New York, USA (2006)

[10] Schmid, H.: Probabilistic Part-of-Speech Tagging Using Decision Trees. In *International Conference on New Methods in Language Processing*. Manchester, UK (1994)