Journal of Computational Information Systems 8: 3 (2012) 1245–1254
Available at http://www.Jofcis.com

Semantic Class Learning with Deep Coordinate Structures in Web Pages ⋆

Xingxing ZHANG, Zhifang SUI*

Institute of Computational Linguistics, Peking University, Beijing 100871, China

Abstract

In this paper, we introduce a kind of descriptive structure, the deep coordinate structure, which universally exists in web pages. And it's used for the task of semantic class learning. At first, we extract the deep coordinate structures in web pages, and use the seed instances to filter them and extract the candidates; during the stage of evaluating the candidates, we build a graph among the seeds, web pages, coordinate structures and candidates, and use PageRank to evaluate the candidates. We conduct experiments on 14 semantic classes (7 Chinese classes and 7 English classes) and consistently achieve high precisions. We compare our results with several state-of-art systems, and our system is better on some of the semantic classes.

Keywords: Deep Coordinate Structure; Semantic Class Learning; HTML Tags; PageRank

1 Introduction

One of semantic class learning's tasks is to find all the instances in a semantic class. It can be very valuable for some natural language processing tasks (such as Question Answering [1]). Search engines collect large sets of named entities [2] to improve query suggestions and to understand query intents. To construct large quantities of semantic classes is expensive and laborious, so some researchers are developing automatic semantic class learning technology.

A lot of research has been done in semantic class learning. Mainly are finding the coordinate terms of some seeds in corpus, this is also called set expansion [3]. Many of these approaches use text patterns to find the coordinate terms of the seeds, such as "NP such as NP,. . . , and NP"[4], "CLASS_NAME such as MENBER and *"[5], automatically extracted patterns([6], [2], [3], etc.). But the frequencies of these patterns in articles are relatively low and it's not often to find 10 or more NPs in "NP such as NP,. . . , and NP".

We can get plenty of web pages from the WWW, some html tags have the semantic of coordination (e.g. <ul><li>. . . </li>. . . </ul>), we call it coordinate structure in this paper. They

⋆This work is supported by NSFC Project 60873156, 61075067, National Social Science Fund 09BYY032 and HGJ Grant No. 2011ZX01042-001-001.

*Corresponding author.
Email address: szf@pku.edu.cn (Zhifang SUI).

are very common on the web and usually have many coordinate terms (in some movie sites, it's not unusual to see 20 movies in a HTML list or a HTML table). [7] uses coordinate structure to find coordinate terms, but the candidate terms must appears between two coordinate part tags (between <li> and </li>), if other tags embedded between the coordinate part tags, some hidden coordinate terms can't be found. Such as Table 1, it's an unordered HTML list with 4 coordinate parts (<li>. . . </li>), between<li> and </li> are many tags, it seems no coordinate terms, in fact the bolded movie *Bruke and Hare* and other bolded movies (*Insidious*, *Source Code* and *Your Highness*) between <li> and </li> are coordinate terms.

To find these hidden coordinate terms, we employ the deep coordinate structure (definition in section 3) and develop an algorithm to extract these deep coordinate structures in web pages. The corresponding concept to deep coordinate structure is basic coordinate structure. For example, the html tags in Table 2 form a basic coordinate structure as between the coordinate part <li>. . . </li> is a string (not other html tags) and the html tags in Table 1 form a deep coordinate structure. We will use DCS stands for Deep Coordinate Structure. Below, Section 2 presents the related work. Section 3 gives the definition of DCS and the architecture of the algorithm. Section 4 presents how we extract DCS and extract candidates. Section 5 presents the ranking method, and Section 6 is the experimental results. The paper concludes in Section 7.

```
< ul >< li >
< p >< imgsrc = "rzdf.jpg" >< /p >
< p > Name :< astyle = "front − size : 14px"href = "rzdf.htm" > BurkeandHare < /a >< /p >
< p > Score :< em > 9.6 < /em >< /p >< /li >
< li >
< p >< imgsrc = "gys.jpg" >< /p >
< p > Name :< astyle = "front − size : 14px"href = "gys.htm" > Insidious < /a >< /p >
< p > Score :< em > 6.0 < /em >< /p >< /li >
< li >
< p >< imgsrc = "zqxs.jpg" >< /p >
< p > Name :< astyle = "front − size : 14px"href = "zqxs.htm" > SourceCode < /a >< /p >
< p > Score :< em > 6.0 < /em >< /p >< /li >
< li >
< p >< imgsrc = "czj.jpg" >< /p >
< p > Name :< astyle = "front − size : 14px"href = "czj.htm" > YourHighness < /a >< /p >
< p > Score :< em > 8.1 < /em >< /p >< /li >
< /ul >
```

Table 1: A web page contains deep coordinate structure

```
< ul >< li > BurkeandHare < /li >
< li > Insidious < /li >
< li > SourceCode < /li >
< li > YourHighness < /li >< /ul >
```

Table 2: basic coordinate structure

2 Related Work

Existing efforts for semantic class learning has been done by different approaches. Some researchers use unsupervised clustering methods ([8, 9, 10]), but the granularities of the result are not easy to control; Some researchers use machine learning methods (e.g.[11]); But the most prevailing approaches are the weakly supervised approaches, using some seeds, and expand the seeds (set expansion).

Our research focuses on weakly supervised approaches. [12] uses coordinate text patterns to extract candidates from plain text, and then build co-occurrence vectors to evaluate the candidates. [5] uses "double anchor" pattern (which is a stronger edition of the patterns in [4]) as a query sending to a search engine and mining the snippets, then a graph is built to evaluation the candidates. [6] uses a class name and a seed to construct query "CLASS_NAME*seed*" to a search engine, and extract patterns automatically by mining the snippets, then evaluate the candidates by computing the pointwise mutual information between the candidates and the patterns. [2] uses the prefix and the postfix of seeds in a query as patterns to extract candidates, and evaluate the candidates by creating context vectors. All of these methods are based on text patterns, and their corpuses are all plain text (some use web pages, but they didn't use the tags in them). As has been mentioned in the introduction part, text patterns have some drawbacks.

Some researchers tried to use html tag patterns to mine the web pages (with tags). [7] only uses the basic coordinate structures (e.g. <ol><li>T</li>...</ol>) and the candidates must appear between exactly the coordinate part tags (e.g. between <li> and </li>), they can extract the movie names in Table 2, but not in Table 1. [3] tries to extract coordinate structures automatically by find the largest common prefix and postfix of all seeds on string level, but they didn't see the tags as a whole. [13] uses coordinate structures to extract hyponym relations. They define a path for each string between two tags, and the strings with the same path are in the same semantic class. Their work is very close to ours, but the candidates in different coordinate structures may have the same path, thus producing noisy candidates.

3 Deep Coordinate Structure

In this paper, we call the html tags that have the semantic of coordination *basic coordinate structure* (e.g. <ol><li>T</li>...</ol>, <table><tr><td>T</td></tr>...</table>, Table 2, etc.). In real pages, many tags can be nested in each coordinate part, and if all the tags nested in the coordinate parts are the same in the corresponding position (the tags' attributes and attribute values can be different and the text between the tags can be different), we call this coordinate structure *deep coordinate structure*. Such as in Table 1, the tags in each coordinate part (<li>...</li>) of the unordered html list are all "<p></p><p><a></a></p><p><em></em></p>", so it's a DCS.

The deep coordinate structure only requires the tags nested in coordinate parts to be the same; the tags with the same name but different attributes and attribute values are considered the same tag; the text between tags are also ignored. Why? Human can recognize Table 1's page is describing 4 entities (movies) because they have identical structures. They all made up of a picture, a line of text with hyperlink and another line of text, but we don't care what the picture is (we ignore the attributes and attribute values of <img> tag) and we don't care what the line of text is (we ignore the text between <a> and </a>).

The architecture of our semantic class learning system is as follows. The system is made up of 3 parts: *corpus fetcher*, *instance extractor* and *graph ranker*. The input of the system is the seeds and keywords related to the semantic class. The corpus fetcher sends the keywords related to the semantic class to a search engine and downloads the top 150 URLs returned by the search engine. For example, if we want to extract the latest movies, we can send the keywords "latest movies" to a search engine. The instance extractor extracts the candidates (section 4). The graph ranker gathers information when the instance extractor extracts the candidates, and builds a graph with the information. The candidates are ranked by their PageRank values in the graph (section 5).

4 Extracting the Candidates

4.1 Extracting the deep coordinate structures

In a DCS, the tags nested in the coordinate parts are the same in the corresponding position. Then a DCS can be represented as a list of html tags and a number indicating the number of repetitions. First, we convert the whole page to an html tag list, and in the list, 2 tags are equal if they have the same tag name. Then, find all the continuous repeated sub-sequence of the tag list, and they are the candidate DCSs. The length of repeated part and the number of repetitions can have some constraints, and in our experiment we only consider the sub-sequence whose length of repeated part is between 4 and 100 and the number of repetitions is larger than 3. Still there are some duplicate DCSs. E.g. if the sequence *<li><a></a></li>* appears 50 times continuously, then we will get a DCS whose length of repeated part is 4 and the number of repetitions is 50, a DCS whose length is 8 and the number of repetitions is 25, . . . , a DCS whose length is 48 and the number of repetitions is 4, and all these DCS start at the same position in the page, and in fact they are the same DCS with different spans. So we only retain the coordinate structure with the shortest length of repeated part.

4.2 Filtering the deep coordinate structures and extracting the candidates

Not all the DCSs extracted in 4.1 are useful, we only retain the DCSs which at least one seed appears exactly between two tags of the DCS. And the position of the seed in the repeated part should be recorded. We extract all the strings in the same position in the repeated part as the seed as candidates. For example, if "Burke and Hare" is a seed and we use the algorithm in 4.1 and 4.2 to extract the web page in Table 1. The extraction result is shown in Table 3.

Table 3: The extraction results of Table 1 with seed *Burke and Hare*

Coordinate Part	pos	N rep
<li><p><img></p><p><a></a></p><p><em></em><p></li>	6	4
Candidates: *Burke and Hare, Insidious, Source Code, Your Highness*		

5 Ranking the Candidates

It's inevitable that there will be some incorrect instances in the candidates. For example, maybe due to the page layout, some movie with a very long name, e.g. "tyler perry's madea's big happy family", can't be shown completely on the page, it may be "tyler perry's madea's". So we need to rank the candidates.

5.1 Building an extraction graph

[3] builds a graph when ranking the candidates. Our graph is similar to theirs except that we use DCS vertices instead of wrapper vertices in their graph. Intuitively, a good DCS should extract better candidates than a bad one; a DCS who can always extract good candidates should be better than a DCS who always extracts bad candidates. Also, a good page should extract better DCSs than a bad one; a page who can always extract good DCSs should be better than a page who always extracts bad DCSs.

We build a graph to model the relations above. We define the extraction graph as $G =< V, E >$, where each vertex $v \in V$ is seeds, a page, a DCS or a candidate. If a page w extracts a DCS d, then $< w, d >\in E$, and $< d, w >\in E$. If a DCS d can find a candidate c, then $< d, c >\in E$, and $< c, d >\in E$. During the stage of candidates extraction, the seeds s find the pages from which the candidates can be extracted, so $< s, w >\in E$, and $< w, s >\in E$ for each $w \in V$. We build a reverse edge so that the web pages and the DCSs (the DCSs and the candidates, the seeds and the pages) can evaluate each other in the following graph random walk.

Page proposed PageRank to evaluate the pages' importance[14]. It based on the idea that pages of different quality contribute differently to the pages they links to. We use PageRank to evaluate the importance of vertices in the graph. So that pages, DCSs, candidates of different quality contribute differently to their neighbors. PageRank exactly models the intuition at the beginning of 5.1. Fig. 1 is an example of the graph. "火箭湖人热火"(Rockets Lakers Heat) are seeds; "sports.sohu.com" and "china.nba.com" are 2 pages; DCS1, DCS2 and DCS3 are 3 DCSs; other vertices are candidates.

5.2 Ranking candidates by PageRank

The candidates are evaluated by PageRank. In the extraction graph G

$$\forall < u, v >\in E \qquad w(< u, v >) = 1; \qquad \forall < u, v >\notin E \qquad w(< u, v >) = 0 \tag{1}$$

$w(< u, v >)$ is the weight of $< u, v >$. The transition probability matrix M is

$$M_{x,y} = \frac{(1-\lambda) * w(< x, y >)}{\sum_{<x,u>\in E} w(< x, u >)} + \frac{\lambda}{|V|} \tag{2}$$

In the experiment, for λ we use the suggested value in [14], 0.15. We use v_t denotes the probability distribution of each vertex in the tth iteration, then

$${v_{t+1}}^T = {v_t}^T M \tag{3}$$

We use v_0 denotes the initial probability distribution of each vertex. In PageRank, the initial probability for each vertex in graph is $1/|V|$.When PageRank converges, we rank the candidate vertices by their probabilities in v_t. That is the final ranking of the candidates.

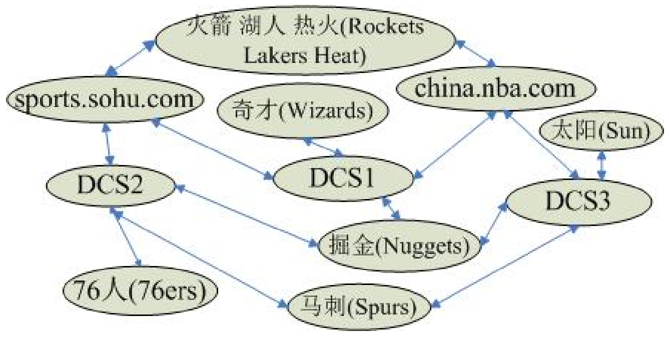

Fig. 1: An example of the extraction graph

Table 4: keywords used by *corpus fetcher*

Class	kwords & N Pages	Class	kwords & N Pages
电影(Movie)	最新电影@150(latest movies)	Movie	latest movies@150

歌曲(Song)
歌手(Singer)
	最新流行歌曲@150(latest popular songs)	Song
		Singer
			popular songs@150

NBA球队(NBA)	NBA@150	Video game	video games@150
汽车(car)	汽车报价@150(car price)	Car	car price@150
笔记本(laptop)	笔记本报价@150(laptop price)	Laptop	laptop prices@150
国家(country)	奥运金牌榜@50(Olympics medals table) 世界杯@50(World Cup) 欧洲杯@50(European Cup)	University	world university ranking@150

6 Experimental Setup

In the experiment, the corpus fetcher gets relevant pages from a search engine. For the Chinese class, we crawl the search results of Baidu; for the English class, we crawl the search results of Yahoo! The keywords used by corpus fetcher for different classes are shown in Table 4.

6.1 Instance extraction and evaluation criterion

During the stage of candidate extraction, the repeated part of DCS is not the longer the better. We observed that when the length of repeated part exceeds 100 tags, they always cross several lists (or tables, etc.). so we constrains the maximum length 100 html tags. During the stage of evaluation, we evaluate the candidates by PageRank, besides we evaluate the candidates by their extracted frequencies. We use Precision, Recall and MAP as our evaluation criterion. AP is average precision. MAP is the average of APs.

$$AP(L) = \frac{\sum_{r=1}^{|L|} Prec(r) * CorrectInstance(r)}{|CorrectInstances|} \tag{4}$$

Table 5: Experimental results

Class	N	Precision		Recall		MAP	
		Freq	PageRank	Freq	PageRank	Freq	PageRank
电影(Movie)	300	95.00%	97.00%			98.08%	99.20%
歌曲(Song)	100	84.00%	99.00%			90.96%	99.85%
歌手(Singer)	100	99.00%	99.00%			99.95%	99.87%
NBA球队(NBA)	30	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
汽车(car)	100	98.00%	98.00%	74.26%	76.24%	99.33%	99.92%
笔记本(laptop)	35	85.71%	100.00%	69.23%	73.08%	95.56%	100.00%
国家(country)	100	87.00%	94.00%	43.59%	47.69%	94.09%	97.67%
Movie	250	92.00%	91.60%			93.43%	98.91%
Song	100	87.00%	98.00%			85.77%	100.00%
Singer	180	97.77%	98.33%			97.75%	99.83%
Car	101	99.01%	99.01%	89.28%	89.28%	100.00%	99.96%
Laptop	30	83.33%	86.67%	73.33%	73.33%	97.78%	97.14%
University	250	97.20%	100.00%	76.67%	73.33%	99.02%	100.00%
Videogame	119	89.92%	89.92%			93.32%	94.46%
Average	128.21	92.50%	96.47%	75.19%	76.14%	96.07%	99.06%

L is a ranked list of candidates; $Prec(r)$ is the precision of the top r candidates in L; $CorrectIn-stance(r)$ returns 1 if the rth candidate is correct, otherwise returns 0. 3 groups of seeds are used to compute MAP.

6.2 Experimental results

The experimental results are shown in Table 5. N means the top N candidates will be used in the evaluation. "Freq" means the candidates are evaluated by their extracted frequencies. And "PageRank" means the candidates are evaluated by their PR values. For the classes "电影(Movie)", "歌曲(Song)", "歌手(Singer)", "Movie", "Song", "Singer", "Video game", their recalls are not computed, as these classes are changing too fast and it's very hard to find a gold standard. For "NBA球队(NBA)", we use the current 30 teams as the gold standard; For "汽车(Car)", we use the 101 car brands listed on www.autohome.com.cn; For "笔记本(laptop)", we use 26 major laptop brands on www.pconline.com.cn; For "Car", we use the gold standard "popular car maker" in [3] (56 English car brands); For "Laptop", we use the major 15 laptop brands in Wikipedia (List of laptop brands and manufacturers); For "University", we use the QS world university rankings 2010-2011 top 150. "笔记本(laptop)" and "Laptop" ("汽车(Car)" and "Car") we use different golden standards, as the popular laptops (cars) in China and other countries are different.

In Table 5, most of the time, the results of PageRank are better than the results of Freq (the Precision, Recall and MAP are all higher). We get the average precision of 14 classes 96.47% (evaluate by PageRank) and 92.50% (evaluate by frequencies). And the average recall is about

75% for the two evaluation methods. The recall is not very high, maybe it's because we only use 150 pages as our corpus, and most of the time only 50 to 70 pages will be extracted candidates. The result may have some improvement if a larger corpus is used. Also, we observed that if many of the candidates only extracted 1 time, the extracting result will be very bad, as our graph will become very sparse, and the PageRank value for each node will become similar.

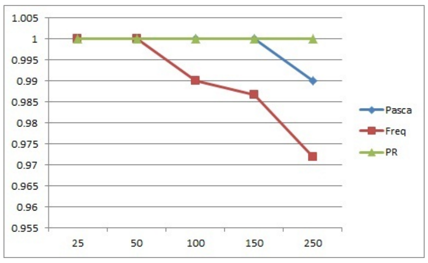

Fig. 2: Class: University

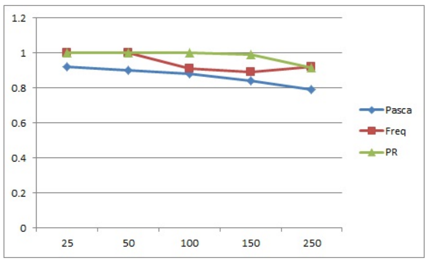

Fig. 3: Class: Movie

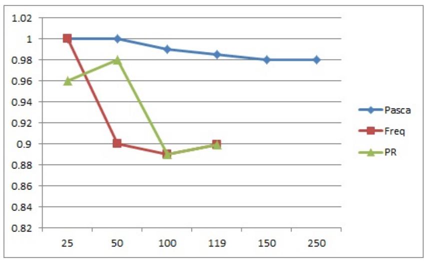

Fig. 4: Class: VideoGame

Fig. 5: Comparison of top 250 candidates precision with Pasca results

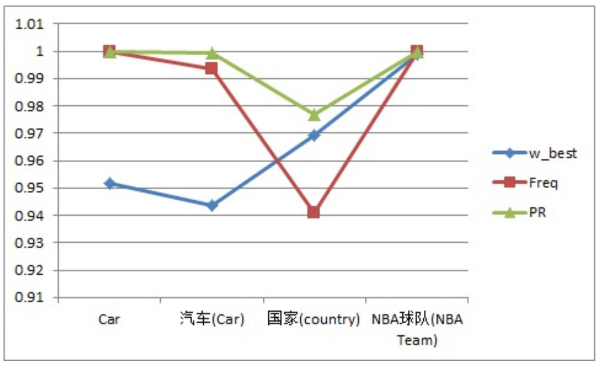

Fig. 6: Comparison of MAP at max.100 with Wang's results

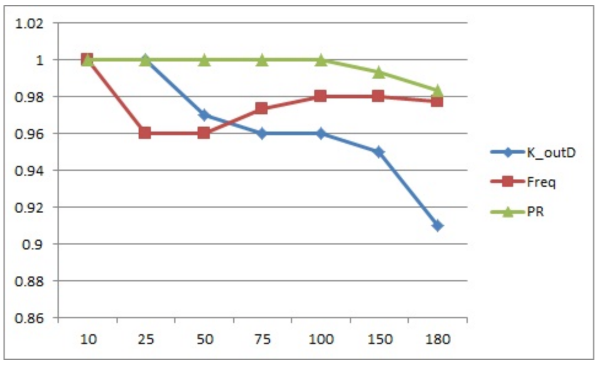

Fig. 7: Comparison of top 180 candidates precision of "singer" with Kozareva's best results

6.3 Comparison to previous results

Wang tested their algorithm [3] in 3 different languages, totally 36 classes. Wang only give the MAP for each class, and didn't provide the length of their ranked candidate list (only give the maximum length). We have 4 classes that are the same with theirs and the length of our list (N in table 5) is provided. We take MAP of theirs best results at max.100 to compare with ours. The comparison is shown in Fig. 6. 3 of the 4 classes are better than theirs, and only "NBA球队(NBA)" is the same with theirs (100%).

Pasca tested his algorithm on 10 classes[2], and 3 of them are the same with ours ("Movie", "University", "VideoGame"), and the comparison of top 250 candidates' precision is in Fig. 5. For "University" and "Movie", our results are better (even the frequency based method are better). For the class "VideoGame", our system only produces 119 candidates, and Pasca's result is much higher. We observe that only 19 (of 150) pages are extracted candidates. Increase the size of corpus and use more domain specific corpus may improve the results. Also, in their experiment, they use 50 million unique queries of Google in 2006; our corpus is only 150 web pages returned by a search engine. But we get better or comparable results.

Kozareva tested their algorithm on 4 classes[5]. They reported several results by different ranking scheme (out degree is the best). We compare their best results of "Singer" with ours. The comparison is shown in Fig. 7. "K_outD" means Kozareva's results by using out degree to rank the candidates. Our results of PageRank based method are better than theirs. Also, in Kozareva's method, for some classes, they need to crawl the search results of the search engine several thousand times. But our approach only needs to crawl the first 150 results, and we need about 10 seconds to get the extraction results.

7 Conclusion

In this paper, we find the deep coordinate structure (DCS), and developed a semantic class learning algorithm with DCS. When evaluating the candidates, we build an extraction graph, and use PageRank to evaluate the candidates. The experimental results show the effectiveness of our algorithm. In addition, a number of instances and their attributes may appear in a DCS simultaneously. So, in the next step, we will try to extract the instances and the attributes simultaneously.

References

[1] Richard C. Wang, Nico Schlaefer, William W. Cohen, and Eric Nyberg, 2008. Automatic set expansion for list question answering. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, Hawaii, October 2008: 947-954.

[2] Marius Pasca, 2007. Weakly-supervised discovery of named entities using web search queries. In *Proceedings of the 16 ACM conference on Conference on information and knowledge management*, New York, 2007: 683-690.

[3] Richard C. Wang and William W. Cohen. 2007. Language-independent set expansion of named entities using the web. In *Proceedings of ICDM*, 2007: 342-350.

[4] M. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. of the 14th conference on Computational linguistics*, pages 539-545.

[5] Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy, 2008.Semantic class learning from the web with hyponym pattern linkage graphs. In *Proceedings of ACL-08: HLT*, Columbus, Ohio, June 2008: 1048-1056.

[6] Patrick Pantel and Marco Pennacchiotti, 2006. Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations. In *Proceedings of Conference on Computational Linguistics/Association for Computational Linguistics*, Sydney, Australia, 2006: 113-120.

[7] Shuming Shi, Huibin Zhang, Xiaojie Yuan, and Ji-Rong Wen, 2010. Corpus-based Semantic Class Mining: Distributional vs. Pattern-Based Approaches. In *the 23rd International Conference on Computational Linguistics*, Beijing, August 2010.

[8] Dekang Lin. 1998. Automatic Retrieval and Clustering of Similar Words.*COLING-ACL 8.*

[9] D.Lin and P.Pantel. 2002. Concept discovery from text. In *Proc. of the 19th International Conference on Computational linguistics*, pages 1-7.

[10] D. Davidov and A. Rappoport. 2006. Efficient unsupervised discovery of word categories using symmetric patterns and high frequency words. In *Proc. of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL.*

[11] Huang, R. and Riloff, E., 2010. Inducing Domain-specific Semantic Class Taggers from (Almost) Nothing. In *Proceedings of The 48th Annual Meeting of the Association for Computational Linguistics*, 2010.

[12] Sarmento, L.; Jijkuon, V.; de Rijke, M.; and Oliveira, E., 2007. ore like these growing entity classes from seeds. In *Proceedings of CIKM-07*, Lisbon, Portugal, 2007: 959-962.

[13] Keiji Shinzato and Kentaro Torisawa. 2005. A Simple WWW-based Method for Semantic Word Class Acquisition. *Recent Advances in Natural Language Processing (RANLP 5)*, Borovets, Bulgaria.

[14] L. Page, S. Brin, R. Motwani, and T. Winograd, 1998. The PageRank citation ranking: Bringing order to the web[R]. Technical report, Stanford Digital Library Tech. Project, 1998.