WoSIT: A Word Sense Induction Toolkit for Search Result Clustering and Diversification

Daniele Vannella, Tiziano Flati and **Roberto Navigli**
Dipartimento di Informatica
Sapienza Università di Roma
{vannella,flati,navigli}@di.uniroma1.it

Abstract

In this demonstration we present WoSIT, an API for Word Sense Induction (WSI) algorithms. The toolkit provides implementations of existing graph-based WSI algorithms, but can also be extended with new algorithms. The main mission of WoSIT is to provide a framework for the extrinsic evaluation of WSI algorithms, also within end-user applications such as Web search result clustering and diversification.

1 Introduction

The Web is by far the world's largest information archive, whose content – made up of billions of Web pages – is growing exponentially. Unfortunately the retrieval of any given piece of information is an arduous task which challenges even prominent search engines such as those developed by Google, Yahoo! and Microsoft. Even today, such systems still find themselves up against the lexical ambiguity issue, that is, the linguistic property due to which a single word may convey different meanings.

It has been estimated that around 4% of Web queries and 16% of the most frequent queries are ambiguous (Sanderson, 2008). A major issue associated with the lexical ambiguity phenomenon on the Web is the low number of query words submitted by Web users to search engines. A possible solution to this issue is the diversification of search results obtained by maximizing the dissimilarity of the top-ranking Web pages returned to the user (Agrawal et al., 2009; Ashwin Swaminathan and Kirovski, 2009). Another solution consists of clustering Web search results by way of clustering engines such as Carrot[1] and Yippy[2] and presenting them to the user grouped by topic.

[1] http://search.carrot2.org
[2] http://yippy.com

Diversification and Web clustering algorithms, however, do not perform any semantic analysis of search results, clustering them solely on the basis of their lexical similarity. Recently, it has been shown that the automatic acquisition of the meanings of a word of interest, a task referred to as Word Sense Induction, can be successfully integrated into search result clustering and diversification (Navigli and Crisafulli, 2010; Di Marco and Navigli, 2013) so as to outperform non-semantic state-of-the-art Web clustering systems.

In this demonstration we describe a new toolkit for Word Sense Induction, called WoSIT, which i) provides ready implementations of existing WSI algorithms; ii) can be extended with additional WSI algorithms; iii) enables the integration of WSI algorithms into search result clustering and diversification, thereby providing an extrinsic evaluation tool. As a result the toolkit enables the objective comparison of WSI algorithms within an end-user application in terms of the degree of diversification of the search results of a given ambiguous query.

2 WoSIT

In Figure 1 we show the workflow of the WoSIT toolkit, composed of three main phases: WSI; semantically-enhanced search result clustering and diversification; evaluation. Given a target query q whose meanings we want to automatically acquire, the toolkit first builds a graph for q, obtained either from a co-occurrence database, or constructed programmatically by using any user-provided input. The co-occurrence graph is then input to a WSI algorithm, chosen from among those available in the toolkit or implemented by the user. As a result, a set of word clusters is produced. This concludes the first phase of the WoSIT workflow. Then, the word clusters produced are used for assigning meanings to the search results returned by a search engine for the query q, i.e. search result disambiguation. The

Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 67–72,
Baltimore, Maryland USA, June 23-24, 2014.

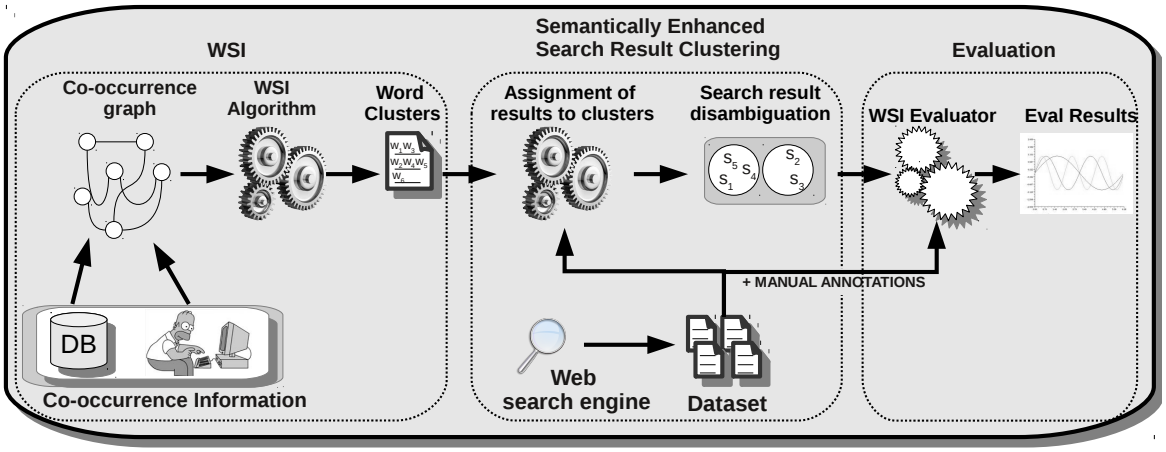

Figure 1: The WoSIT workflow.

outcome is that we obtain a clustering of search results. Finally, during the third phase, we apply the evaluation module which performs an evaluation of the search result clustering quality and the diversification performance.

We now describe in detail the three main phases of WoSIT.

2.1 Word Sense Induction

The first phase of WoSIT consists of the automatic identification of the senses of a query of interest, i.e. the task of Word Sense Induction. Although WoSIT enables the integration of custom implementations which can potentially work with any WSI paradigm, the toolkit provides ready-to-use implementations of several graph-based algorithms that work with word co-occurrences. All these algorithms carry out WSI in two steps: co-occurrence graph construction (Section 2.1.1) and discovery of word senses (Section 2.1.2).

2.1.1 Co-occurrence graph construction

Given a target query q, we build a co-occurrence graph $G_q = (V, E)$ such that V is the set of words co-occurring with q and E is the set of undirected edges, each denoting a co-occurrence between pairs of words in V. In Figure 2 we show an example of a co-occurrence graph for the target word *excalibur*.

WoSIT enables the creation of the co-occurrence graph either programmatically, by adding edges and vertices according to any user-specific algorithm, or starting from the statistics for co-occurring words obtained from a co-occurrence database (created, e.g., from a text corpus, as was done by Di Marco and Navigli (2013)). In either case, weights for edges have to be provided in terms of the correlation strength between pairs of words (e.g. using Dice, Jaccard or other co-occurrence measures).

The information about the co-occurrence database, e.g. a MySQL database, is provided programmatically or via parameters in the properties configuration file (`db.properties`). The co-occurrence database has to follow a given schema provided in the toolkit documentation. An additional configuration file (`wosit.properties`) also allows the user to specify additional constraints, e.g. the minimum weight value of co-occurrence (the `wordGraph.minWeight` parameter) to be added as edges to the graph.

The graphs produced can also be saved to binary (i.e. serialized) or text file:

```
g.saveToSer(fileName);
g = WordGraph.loadFromSer(fileName);
```

```
g.saveToTxt(fileName);
g = WordGraph.loadFromTxt(fileName);
```

We are now ready to provide our co-occurrence graph, created with just a few lines of code, as input to a WSI algorithm, as will be explained in the next section.

2.1.2 Discovery of Word Senses

Once the co-occurrence graph for the query q is built, it can be input to any WSI algorithm which extends the `GraphClusteringAlgorithm` class in the toolkit. WoSIT comes with a number of ready-to-use such algorithms, among which:

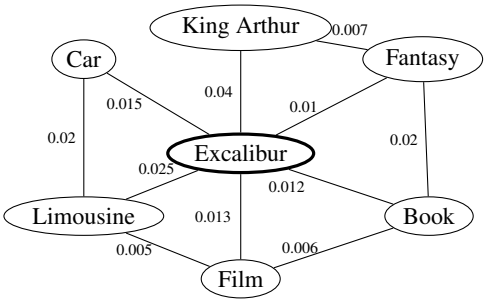

Figure 2: Example of a co-occurrence graph for the word *excalibur*.

- **Balanced Maximum Spanning Tree (B-MST)** (Di Marco and Navigli, 2013), an extension of a WSI algorithm based on the calculation of a Maximum Spanning Tree (Di Marco and Navigli, 2011) aimed at balancing the number of co-occurrences in each sense cluster.
- **HyperLex** (Véronis, 2004), an algorithm which identifies hubs in co-occurrence graphs, thereby identifying basic meanings for the input query.
- **Chinese Whispers** (Biemann, 2006), a randomized algorithm which partitions nodes by means of the iterative transfer of word sense information across the co-occurrence graph (Biemann, 2006).
- **Squares, Triangles and Diamonds (SquaT++)** (Di Marco and Navigli, 2013), an extension of the SquaT algorithm (Navigli and Crisafulli, 2010) which exploits three cyclic graph patterns to determine and discard those vertices (or edges) with weak degree of connectivity in the graph.

We also provide an implementation of a word clustering algorithm, i.e. **Lin98** (Lin, 1998), which does not rely on co-occurrence graphs, but just on the word co-occurrence information to iteratively refine word clusters on the basis of their "semantic" relationships.

A programmatic example of use of the B-MST WSI algorithm is as follows:

```
BMST mst = new BMST(g);
mst.makeClustering();
Clustering wordClusters =
                    mst.getClustering();
```

where `g` is a co-occurrence graph created as explained in Section 2.1.1, provided as input to the constructor of the algorithm's class. The `makeClustering` method implements the induction algorithm and creates the word clusters, which can then be retrieved calling the `getClustering` method. As a result an instance of the `Clustering` class is provided.

As mentioned above, WoSIT also enables the creation of custom WSI implementations. This can be done by extending the `GraphClusteringAlgorihm` abstract class. The new algorithm just has to implement two methods:

```
public void makeClustering();
public Clustering getClustering();
```

As a result, the new algorithm is readily integrated into the WoSIT toolkit.

2.2 Semantically-enhanced Search Result Clustering and Diversification

We now move to the use of the induced senses of our target query q within an application, i.e. search result clustering and diversification.

Search result clustering. The next step (cf. Figure 1) is the association of the search results returned by a search engine for query q with the most suitable word cluster (i.e. meaning of q). This can be done in two lines:

```
SnippetAssociator associator =
        SnippetAssociator.getInstance();
SnippetClustering clustering =
     associator.associateSnippet(
         targetWord,
         searchResults,
         wordClusters,
         AssociationMetric.DEGREE_OVERLAP);
```

The first line obtains an instance of the class which performs the association between search result snippets and the word clusters obtained from the WSI algorithm. The second line calls the association method `associateSnippet` which inputs the target word, the search results obtained from the search engine, the word clusters and, finally, the kind of metric to use for the association. Three different association metrics are implemented in the toolkit:

- `WORD_OVERLAP` performs the association by maximizing the size of the intersection between the word sets in each snippet and the word clusters;
- `DEGREE_OVERLAP` performs the association by calculating for each word cluster the sum

of the vertex degrees in the co-occurrence graph of the words occurring in each snippet;

- TOKEN_OVERLAP is similar in spirit to WORD_OVERLAP, but takes into account each token occurrence in the snippet bag of words.

Search result diversification. The above two lines of code return a set of snippet clusters and, as a result, semantically-enhanced search result clustering is performed. At the end, the resulting clustering can be used to provide a diversified reranking of the results:

```
List<Snippet> snippets =
        clustering.diversify(sorter);
```

The diversify method returns a flat list of snippet results obtained according to the Sorter object provided in input. The Sorter abstract class is designed to rerank the snippet clusters according to some predefined rule. For instance, the CardinalitySorter class, included in the toolkit, sorts the clusters according to the size of each cluster. Once a sorting order has been established, an element from each snippet cluster is added to an initially-empty list; next, a second element from each cluster is added, and so on, until all snippets are added to the list.

The sorting rules implemented in the toolkit are:

- CardinalitySorter: sorts the clusters according to their size, i.e. the number of vertices in the cluster;
- MeanSimilaritySorter: sorts the clusters according to the average association score between the snippets in the cluster and the backing word cluster (defined by the selected association metrics).

Notably, the end user can then implement his or her own custom sorting procedure by simply extending the Sorter class.

2.2.1 Search Result Datasets

The framework comes with two search result datasets of ambiguous queries: the AMBIENT+MORESQUE dataset made available by Bernardini et al. (2009) and Navigli and Crisafulli (2010), respectively, and the SemEval-2013-Task11 dataset.[3] New result datasets can be provided by users complying with the dataset format described below.

[3]For details visit http://lcl.uniroma1.it/wosit/.

A search result dataset in WoSIT is made up of at least two files:

- topics.txt, which contains the queries (topics) of interest together with their numeric ids. For instance:

```
id   description
1    polaroid
2    kangaroo
3    shakira
...  ...
```

- results.txt, which lists the search results for each given query, in terms of URL, page title and page snippet:

```
ID  url   title   snippet
1.1 http://www.polaroid.com/ Polaroid | Home ...
1.2 http://www.polaroid.com/products products...
1.3 http://en.wikipedia.org/wiki/Polaroid_Cor...
... ...
```

Therefore, the two files provide the queries and the corresponding search results returned by a search engine. In order to enable an automatic evaluation of the search result clustering and diversification output, two additional files have to be provided:

- subTopics.txt, which for each query provides the list of meanings for that query, e.g.:

```
ID  description
1.1 Polaroid Corporation, a multinational con...
1.2 Instant film photographs are sometimes kn...
1.3 Instant camera (or Land camera), sometime...
... ...
```

- STRel.txt, which provides the manual associations between each search result and the most suitable meaning as provided in the subTopics.txt file. For instance:

```
subTopicID   resultID
1.1          1.1
1.1          1.2
1.1          1.3
...          ...
```

2.3 WSI Evaluator

As shown in Figure 1 the final component of our workflow is the evaluation of WSI when integrated into search result clustering and diversification (already used by Navigli and Vannella (2013)). This component, called the WSI Evaluator, takes as input the snippet clusters obtained for a given query together with the fully annotated search result dataset, as described in the previous section. Two kinds of evaluations are carried out, described in what follows.

```
1  Dataset searchResults = Dataset.getInstance();
2  DBConfiguration db = DBConfiguration.getInstance();
3  for(String targetWord : dataset.getQueries())
4  {
5     WordGraph g = WordGraph.createWordGraph(targetWord, searchResults, db);
6     BMST mst = new BMST(g);
7     mst.makeClustering();
8     SnippetAssociator snippetAssociator = SnippetAssociator.getInstance();
9     SnippetClustering snippetClustering = snippetAssociator.associateSnippet(
10     targetWord, searchResults, mst.getClustering(), AssociationMetric.WORD_OVERLAP);
11    snippetClustering.export("output/outputMST.txt", true);
12 }
13 WSIEvaluator.evaluate(searchResults, "output/outputMST.txt");
```

Figure 3: An example of evaluation code for the B-MST clustering algorithm.

2.3.1 Evaluation of the clustering quality

The quality of the output produced by semantically-enhanced search result clustering is evaluated in terms of Rand Index (Rand, 1971, RI), Adjusted Rand Index (Hubert and Arabie, 1985, ARI), Jaccard Index (JI) and, finally, precision and recall as done by Crabtree et al. (2005), together with their F1 harmonic mean.

2.3.2 Evaluation of the clustering diversity

To evaluate the snippet clustering diversity the measures of S-recall@K and S-precision@r (Zhai et al., 2003) are calculated. These measures determine how many different meanings of a query are covered in the top-ranking results shown to the user. We calculate these measures on the output of the three different association metrics illustrated in Section 2.2.

3 A Full Example

We now show a full example of usage of the WoSIT API. The code shown in Figure 3 initially obtains a search result dataset (line 1), selects a database (line 2) and iterates over its queries (line 3). Next, a co-occurrence graph for the current query is created from a co-occurrence database (line 5) and an instance of the B-MST WSI algorithm is created with the graph as input (line 6). After executing the algorithm (line 7), the snippets for the given query are clustered (lines 8-10). The resulting snippet clustering is appended to an output file (line 11). Finally, the WSI evaluator is run on the resulting snippet clustering using the given dataset (line 13).

3.1 Experiments

We applied the WoSIT API to the AMBIENT+MORESQUE dataset using 4 induction algorithms among those available in the toolkit, where co-occurrences were obtained from the Google Web1T corpus (Brants and Franz, 2006). In Table 1 we show the clustering quality results output by the WoSIT evaluator, whereas in Figure 4 we show the diversification performance in terms of S-recall@K.

Algorithm	Assoc. metr.	Web1T ARI	JI	F1	# cl.
SquaT++	WO	69.65	75.69	59.19	2.1
	DO	69.21	75.45	59.19	2.1
	TO	69.67	75.69	59.19	2.1
B-MST	WO	60.76	71.51	64.56	5.0
	DO	66.48	69.37	64.84	5.0
	TO	63.17	71.21	64.04	5.0
HyperLex	WO	60.86	72.05	65.41	13.0
	DO	66.27	68.00	71.91	13.0
	TO	62.82	70.87	65.08	13.0
Chinese Whispers	WO	67.75	75.37	60.25	12.5
	DO	65.95	69.49	70.33	12.5
	TO	67.57	74.69	60.50	12.5

Table 1: Results of WSI algorithms with a Web1T co-occurrence database and the three association metrics (Word Overlap, Degree Overlap and Token Overlap). The reported measures are Adjusted Rand Index (ARI), Jaccard Index (JI) and F1. We also show the average number of clusters per query produced by each algorithm.

3.2 Conclusions

In this demonstration we presented WoSIT, a full-fledged toolkit for Word Sense Induction algorithms and their integration into search result clustering and diversification. The main contributions are as follows: first, we release a Java API for performing Word Sense Induction which includes several ready-to-use implementations of existing algorithms; second, the API enables the use of the acquired senses for a given query for enhancing

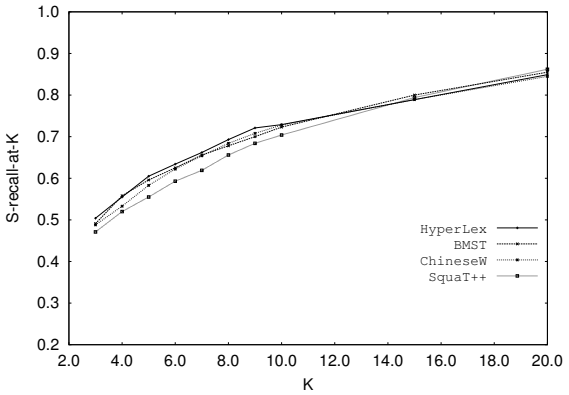

Figure 4: S-recall@K performance.

search result clustering and diversification; third, we provide an evaluation component which, given an annotated dataset of search results, carries out different kinds of evaluation of the snippet clustering quality and diversity.

WoSIT is the first available toolkit which provides an end-to-end approach to the integration of WSI into a real-world application. The toolkit enables an objective comparison of WSI algorithms as well as an evaluation of the impact of applying WSI to clustering and diversifying search results. As shown by Di Marco and Navigli (2013), this integration is beneficial and allows outperformance of non-semantic state-of-the-art Web clustering systems.

The toolkit, licensed under a Creative Commons Attribution-Non Commercial-Share Alike 3.0 License, is available at `http://lcl.uniroma1.it/wosit/`.

References

Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. 2009. Diversifying search results. In *Proc. of the Second International Conference on Web Search and Web Data Mining (WSDM 2009)*, pages 5–14, Barcelona, Spain.

Cherian V. Mathew Ashwin Swaminathan and Darko Kirovski. 2009. Essential Pages. In *Proc. of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 173–182.

Andrea Bernardini, Claudio Carpineto, and Massimiliano D'Amico. 2009. Full-Subtopic Retrieval with Keyphrase-Based Search Results Clustering. In *Proc. of Web Intelligence 2009*, volume 1, pages 206–213, Los Alamitos, CA, USA.

Chris Biemann. 2006. Chinese Whispers - an Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems. In *Proc. of TextGraphs: the First Workshop on Graph Based Methods for Natural Language Processing*, pages 73–80, New York City.

Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram, ver. 1, LDC2006T13. In *Linguistic Data Consortium*, Philadelphia, USA.

Daniel Crabtree, Xiaoying Gao, and Peter Andreae. 2005. Improving web clustering by cluster selection. In *Proc. of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 172–178, Washington, DC, USA.

Antonio Di Marco and Roberto Navigli. 2011. Clustering Web Search Results with Maximum Spanning Trees. In *Proc. of the XIIth International Conference of the Italian Association for Artificial Intelligence (AI*IA)*, pages 201–212, Palermo, Italy.

Antonio Di Marco and Roberto Navigli. 2013. Clustering and Diversifying Web Search Results with Graph-Based Word Sense Induction. *Computational Linguistics*, 39(3):709–754.

Lawrence Hubert and Phipps Arabie. 1985. Comparing Partitions. *Journal of Classification*, 2(1):193–218.

Dekang Lin. 1998. Automatic Retrieval and Clustering of Similar Words. In *Proc. of the* 17^{th} *International Conference on Computational linguistics (COLING)*, pages 768–774, Montreal, Canada.

Roberto Navigli and Giuseppe Crisafulli. 2010. Inducing Word Senses to Improve Web Search Result Clustering. In *Proc. of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 116–126, Boston, USA.

Roberto Navigli and Daniele Vannella. 2013. SemEval-2013 Task 11: Evaluating Word Sense Induction & Disambiguation within An End-User Application. In *Proc. of the* 7^{th} *International Workshop on Semantic Evaluation (SemEval 2013), in conjunction with the Second Joint Conference on Lexical and Computational Semantics (*SEM 2013)*, pages 193–201, Atlanta, USA.

William M. Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850.

Mark Sanderson. 2008. Ambiguous queries: test collections need more sense. In *Proc. of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 499–506, Singapore.

Jean Véronis. 2004. HyperLex: lexical cartography for information retrieval. *Computer, Speech and Language*, 18(3):223–252.

ChengXiang Zhai, William W. Cohen, and John Lafferty. 2003. Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval. In *Proc. of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 10–17, Toronto, Canada.