Clustering and Diversifying Web Search Results with Graph-Based Word Sense Induction

Antonio Di Marco[*]
Sapienza University of Rome

Roberto Navigli[*]
Sapienza University of Rome

Web search result clustering aims to facilitate information search on the Web. Rather than the results of a query being presented as a flat list, they are grouped on the basis of their similarity and subsequently shown to the user as a list of clusters. Each cluster is intended to represent a different meaning of the input query, thus taking into account the lexical ambiguity (i.e., polysemy) issue. Existing Web clustering methods typically rely on some shallow notion of textual similarity between search result snippets, however. As a result, text snippets with no word in common tend to be clustered separately even if they share the same meaning, whereas snippets with words in common may be grouped together even if they refer to different meanings of the input query.

In this article we present a novel approach to Web search result clustering based on the automatic discovery of word senses from raw text, a task referred to as Word Sense Induction. Key to our approach is to first acquire the various senses (i.e., meanings) of an ambiguous query and then cluster the search results based on their semantic similarity to the word senses induced. Our experiments, conducted on data sets of ambiguous queries, show that our approach outperforms both Web clustering and search engines.

1. Introduction

The Web is by far the largest information archive available worldwide. This vast pool of text contains information of the most wildly disparate kinds, and is potentially capable of satisfying virtually any conceivable user need. Unfortunately, however, in this setting retrieving the precise item of information that is relevant to a given user search can be like looking for a needle in a haystack. State-of-the-art search engines such as Google and Yahoo! generally do a good job at retrieving a small number of relevant results from such an enormous collection of data (i.e., retrieving with high precision, low recall). Such systems today, however, still find themselves up against the lexical ambiguity issue

[*] Dipartimento di Informatica, Sapienza Università di Roma, Via Salaria, 113, 00198 Roma Italy. E-mail: {dimarco,navigli}@di.uniroma1.it.

Submission received: 25 April 2012; revised submission received: 26 July 2012; accepted for publication: 12 September 2012.

doi:10.1162/COLI_a_00148

(Furnas et al. 1987; Navigli 2009), that is, the linguistic property due to which a single word may convey different meanings.

Recently, the degree of ambiguity of Web queries has been studied using WordNet (Miller et al. 1990; Fellbaum 1998) and Wikipedia[1] as sources of ambiguous words.[2] It has been estimated that around 4% of Web queries and 16% of the most frequent queries are ambiguous (Sanderson 2008), as also confirmed in later studies (Clough et al. 2009; Song et al. 2009). An example of an ambiguous query is *Butterfly effect*, which could refer to either chaos theory, a film, a band, an album, a novel, or a collection of poetry. Similarly, *black spider* could refer to either an arachnid, a car, or a frying pan, and so forth.

Lexical ambiguity is often the consequence of the low number of query words that Web users, on average, tend to type (Kamvar and Baluja 2006). This issue could be solved by expanding the initial query with unequivocal cue words. Interestingly, the average query length is continually growing. The average number of words per query is now estimated around three words per query,[3] a number that is still too low to eradicate polysemy.

The fact that there may be different informative needs for the same user query has been tackled by diversifying search results, an approach whereby a list of heterogenenous results is presented, and Web pages that are similar to ones already near the top are prevented from ranking too highly in the list (Agrawal et al. 2009; Swaminathan, Mathew, and Kirovski 2009). Today even commercial search engines are starting to rerank and diversify their results. Unfortunately, recent work suggests that diversity does not yet play a primary role in ranking algorithms (Santamaría, Gonzalo, and Artiles 2010), but it undoubtedly has the potential to do so (Chapelle, Chang, and Liu 2011).

Another mainstream approach to the lexical ambiguity issue is that of Web clustering engines (Carpineto et al. 2009), such as Carrot[4] and Yippy.[5] These systems group search results by providing a cluster for each specific meaning of the input query. Users can then select the cluster(s) and the pages therein that best answer their information needs. These approaches, however, do not perform any semantic analysis of search results, clustering them solely on the basis of their lexical similarity.

For instance, given the query *snow leopard*, Google search returns, among others, the snippets reported in Table 1.[6] In the third column of the table we provide the correct meanings associated with each snippet (i.e., either the operating system or the animal sense). Although snippets 2, 4, and 5 all refer to the same meaning, they have no content word in common apart from our query words. As a result, a traditional Web clustering engine would most likely assign these snippets to different clusters. Moreover, snippet 6 shares words with snippets referring to both query meanings (i.e., snippets 1, 2, and 3 in Table 1), thus making it even harder for Web clustering engines to group search

[1] `http://www.wikipedia.org`.

[2] Note that we focus here on the ambiguity of queries in terms of their polysemy, rather than on the identification of aspects or subsenses of a given meaning of a query, as was done in recent work on topic identification (Wang, Chakrabarti, and Punera 2009; Xue and Yin 2011; Wu, Madhavan, and Halevy 2011). We discuss this point further in Section 2.6.

[3] See Hitwise on 2008–2009 Google data: `http://www.hitwise.com/us/press-center/press-releases/google-searches-apr-09`.

[4] `http://search.carrot2.org/stable/search`.

[5] `http://search.yippy.com`.

[6] Results retrieved in May 2011.

Table 1
Some of the top-ranking snippets returned for *snow leopard* by Google search.

#	Snippet	Meaning
1	To advance Mac OS X *Leopard*, Apple engineers...	SOFTWARE
2	The *snow leopard* (Uncia uncia or Panthera uncia) is a moderately large cat native to the mountain ranges	ANIMAL
3	Mac OS X *Snow Leopard* (version 10.6) is the seventh and current major...	SOFTWARE
4	Get the facts on *snow leopards*. Endangered Species Act (ESA): the *snow leopard* is listed as endangered...	ANIMAL
5	*Snow leopards* are exceptional athletes capable of making huge leaps over ravines.	ANIMAL
6	*Snow Leopard*. Even the name seems to underpromise – it's the first 'big cat' OS X codename to reference	SOFTWARE

results effectively. Finally, none of the top-ranking snippets refers to *The Snow Leopard*, a popular 1978 book by Peter Matthiessen.

In this article, we present a novel approach to Web search result clustering that explicitly addresses the language ambiguity issue. Key to our approach is the use of Word Sense Induction (WSI), that is, techniques aimed at automatically discovering the different meanings of a given term (i.e., query). Each sense of the query is represented as a cluster of words co-occurring in raw text with the query. Each search result snippet returned by a Web search engine is then mapped to the most appropriate meaning (i.e., cluster) and the resulting clustering of snippets is returned.

This article provides four main contributions:

- We present a general evaluation framework for Web search result clustering, which we also exploit to perform a large-scale *end-to-end* experimental comparison of several graph-based WSI algorithms. In fact, the output of WSI (i.e., the automatically discovered senses) is evaluated in terms of both the quality of the corresponding search result clusters and the resulting ability to diversify search results. This is in contrast with most literature in the field of Word Sense Induction, where experiments are mainly performed in vitro (i.e., not in the context of an everyday application; Matsuo et al. 2006; Manandhar et al. 2010).
- In order to test whether our results were strongly dependent on the evaluation measures we implemented in the framework, we complemented our extrinsic experimental evaluation with a qualitative analysis of the automatically induced senses. This study was performed via a manual evaluation carried out by several human annotators.
- We present novel versions of previously proposed WSI graph-based algorithms, namely, SquaT++ and Balanced Maximum Spanning Tree (B-MST) (the former is an enhancement of the original SquaT algorithm [Navigli and Crisafulli 2010], and the latter is a variant of MST [Di Marco and Navigli 2011] that produces more balanced clusters).
- We show how, thanks to our framework, WSI can be successfully integrated into real-world applications, such as Web search result

Table 2
The top five categories returned by the Open Directory Project for the query *snow leopard*.

ODP Category	# pages
Science: Biology: Flora and Fauna: . . . Felidae: Uncia	6
Kids and Teens: School Time: Science: . . . Leopards: Snow Leopards	4
Science: Environment: Biodiversity: Conservation: Mammals: Felines	3
Kids and Teens: School Time: Science: . . . Animals: Endangered Species	1
Computers: Emulators: Apple: Macintosh: SheepShaver	1

> clustering, so as to outperform non-semantic state-of-the-art Web clustering systems. To the best of our knowledge, with the exception of some very preliminary results (Véronis 2004; Basile, Caputo, and Semeraro 2009), this is the first time that unsupervised text understanding techniques have been shown to considerably boost an Information Retrieval task in a solid evaluation framework.

This article extends previous conference work (Navigli and Crisafulli 2010; Di Marco and Navigli 2011) by performing a novel, in-depth study of the interactions between different corpora and several different WSI algorithms, including novel ones, within the same framework, and, additionally, by providing a comparison with a state-of-the-art search result clustering engine.

The article is structured as follows: in Section 2 we present related work, in Section 3 we illustrate our approach, end-to-end experiments are reported in Section 4, and in vitro experiments are discussed in Section 5. We present a time performance analysis in Section 6, and conclude the paper in Section 7.

2. Related Work

Our work is aimed at addressing the difficulties arising within the different approaches to the issue of lexical ambiguity in Web Information Retrieval. Given the large body of work in this field, in this section we summarize the main research directions on the topic.

2.1 Web Directories

In Web 1.0—mainly based on static Web pages—the solution to clustering search results was that of manually organizing and categorizing Web sites. The resulting repositories are called **Web directories** and list Web sites by category and possible subcategories. These categories are sometimes organized as taxonomies (like in the Open Directory Project, ODP[7]).

Although Web directories are not search engines, information can be searched therein. So, given a query, the returned search results are organized by category. For instance, given the query *snow leopard* the ODP returns the categories shown in Table 2

[7] http://www.dmoz.org.

(the number of matching Web pages is reported in the second column). As can be seen from this example, the Web directory approach has evident limits:

1. It is static, thus it needs manual updates to cover new pages and new meanings (e.g., the book sense of *snow leopard* is not considered in ODP).
2. It covers only a small portion of the Web (e.g., we only have one Web page categorized with the computing sense of *snow leopard*, cf. the last row of Table 2).
3. It classifies Web pages using coarse categories. This latter feature of Web directories makes it difficult to distinguish between instances of the same kind (e.g., pages about artists with the same surname classified as `Arts:Music:Bands and Artists`).

Although methods for the automatic classification of Web documents have been proposed (Liu et al. 2005; Xue et al. 2008, inter alia) and some problems have been tackled effectively (Bennett and Nguyen 2009), these approaches are usually supervised and still suffer from reliance on a predefined taxonomy of categories. Finally, it has been reported that directory-based systems are among the most ineffective solutions to Web information retrieval (Bruza, McArthur, and Dennis 2000).

2.2 Semantic Information Retrieval

A second approach to query ambiguity consists of associating explicit semantics (i.e., word senses or concepts) with queries and documents, that is, performing **Semantic Information Retrieval** (SIR). SIR is performed by indexing and searching concepts rather than terms, that is, by means of Word Sense Disambiguation (WSD; Navigli 2009), thus potentially coping with two linguistic phenomena: expressing a single meaning with different words (**synonymy**) and using the same word to express various different meanings (**polysemy**). The main idea is that assigning concepts to words can potentially overcome these two issues, enabling a shift from the lexical to the semantic level to be achieved.

Over the years, various methods for SIR have been proposed (Krovetz and Croft 1992; Voorhees 1993; Mandala, Tokunaga, and Tanaka 1998; Gonzalo, Penas, and Verdejo 1999; Kim, Seo, and Rim 2004; Liu, Yu, and Meng 2005, inter alia). Contrasting results have been reported on the benefits of these techniques, however: It has been shown that WSD has to be very accurate to benefit Information Retrieval (Sanderson 1994)—a result that was later debated (Gonzalo, Penas, and Verdejo 1999; Stokoe, Oakes, and Tait 2003). Also, it has been reported that WSD has to be very precise on minority senses and uncommon terms, rather than on frequent words (Krovetz and Croft 1992; Sanderson 2000).

The main drawback of SIR is that it relies on the existence of a reference dictionary to perform WSD (typically, WordNet) and thus suffers this dictionary's static nature and its inherent paucity of most proper nouns. This latter problem is particularly important for Web searches, as users tend to retrieve more information about named entities (e.g., singers, artists, cities) than concepts (such as abstract information about singers or artists). Although lexical knowledge resources that integrate lexicographic senses with named entites on a large scale have recently been created (Navigli and Ponzetto 2012), it is still to be shown that their use for SIR is beneficial. Moreover, these resources do not yet tackle the dynamic evolution of language.

In contrast, our WSI approach to search result clustering automatically discovers both lexicographic and encyclopedic senses of a query (including new ones), thus taking into account all of the mentioned issues.

2.3 Search Result Clustering

A more popular approach to query ambiguity is that of **search result clustering**. Typically, given a query, the system starts from a flat list of text snippets returned from one or more commonly available search engines and clusters them on the basis of some notion of textual similarity. At the root of the clustering approach lies van Rijsbergen's cluster hypothesis (van Rijsbergen 1979, page 45): "closely associated documents tend to be relevant to the same requests," whereas results concerning different meanings of the input query are expected to belong to different clusters.

Approaches to search result clustering can be classified as data-centric or description-centric (Carpineto et al. 2009). The former focus more on the problem of data clustering than on presenting the results to the user. A pioneering example is Scatter/Gather (Cutting et al. 1992), which divides the data set into a small number of clusters and, after the selection of a group, performs clustering again and proceeds iteratively. Developments of this approach have been proposed that improve on cluster quality and retrieval performance (Ke, Sugimoto, and Mostafa 2009). Other data-centric approaches use agglomerative hierarchical clustering (e.g., LASSI [Maarek et al. 2000]), rough sets (Ngo and Nguyen 2005), or exploit link information (Zhang, Hu, and Zhou 2008).

Description-centric approaches are, instead, more focused on the description to produce for each cluster of search results. Among the most popular and successful approaches are those based on suffix trees. Suffix trees are rooted directed trees that contain all the suffixes of a string s. The label of each edge is a non-empty substring of s and each vertex v is labeled with the concatenation of the edge labels on the path from the root to v. If we view the search result snippets to be clustered as a set of strings (i.e., their bag of words), each vertex of the corresponding suffix tree can be considered as a set of documents that share a phrase (i.e., the label of the vertex itself) and therefore the vertices represent a set of base clusters $B = (b_1, b_2, \ldots, b_n)$. The original Suffix Tree Clustering (STC; Zamir et al. 1997; Zamir and Etzioni 1998) algorithm obtains the final clustering by merging the clusters in B with a high overlap in the documents they contain. A scoring function is defined, based on both the number of documents in the base cluster and the length of the common phrase, with the aim of returning only the top k clusters.

Later developments improved the performance of the STC algorithm using document–document similarity scores in order to overcome the low scalability of the original approach (Branson and Greenberg 2002). Crabtree, Gao, and Andreae (2005) identified an issue in the original scoring function whereby unreasonably high scores tend to be assigned to clusters obtained as a result of the merging of very similar base clusters. To solve this problem, they proposed the Extended Suffix Tree Clustering algorithm (ESTC) with a novel scoring function and a new procedure for selecting the top k clusters to be returned.

More recent approaches based on suffix trees extract relevant keyphrases from generalized suffix trees (i.e., trees which contain suffixes of a set of strings

$S = \{s_1, s_2, \ldots, s_{|S|}\}$) in order to choose meaningful labels for the output clusters (Bernardini, Carpineto, and D'Amico 2009; Carpineto, D'Amico, and Bernardini 2011).

Other approaches to description-centric search result clustering in the literature are based on formal concept analysis (Carpineto and Romano 2004), singular value decomposition (Osinski and Weiss 2005), spectral clustering (Cheng et al. 2005), spectral geometry (Liu et al. 2008), link analysis (Gelgi, Davulcu, and Vadrevu 2007), and graph connectivity measures (Di Giacomo et al. 2007). Search result clustering has also been viewed as a supervised salient phrase ranking task (Zeng et al. 2004).

Whereas search result clustering has heretofore been performed without the explicit use of lexical semantics, in our work we show how to exploit search result clustering as the common evaluation framework of both semantic and non-semantic clustering engines.

2.4 Diversification

Rather than clustering the top search results by their similarity, one can aim at reranking them on the basis of criteria that maximize their diversity, so as to present top results which are as different from each other as possible. This technique, called **diversification** of search results, is a recent research topic that, yet again, deals with the query ambiguity issue. To some extent, today's search engines, such as Google and Yahoo!, apply some diversification technique to their top-ranking results.

One of the first examples of diversification algorithms was based on the use of similarity functions to measure the diversity between documents and between document and query (Carbonell and Goldstein 1998). Other diversification techniques use conditional probabilities to determine which document is most different from higher-ranking ones (Chen and Karger 2006), or use affinity ranking (Zhang et al. 2005), based on topic variance and coverage.

An algorithm called Essential Pages (Swaminathan, Mathew, and Kirovski 2009) has been proposed that aims to reduce information redundancy and returns Web pages that maximize coverage with respect to the input query. In this approach the Web search results for a query q are transformed into bags of words containing the terms occurring in the corresponding Web page. Frequency information from raw corpora is then used to find relevant words for q, that is, words which are generally infrequent, but occur often in the results retrieved for q. The coverage score of a search result r is then calculated as a function of the number of terms relevant for q and contained in r. Another interesting approach reformulates the problem explicitly in terms of how to minimize the risk of dissatisfaction for the average user (Agrawal et al. 2009). A greedy algorithm is proposed that balances between relevance and diversity of the search results. The algorithm is evaluated using generalizations of classical Information Retrieval metrics that are based on statistical considerations and take into account the intentions of the user.

More recently, vector space model representations have been explored to improve diversity in search results (Santamaría, Gonzalo, and Artiles 2010). Web page results are represented as vectors and compared against vector representations of encyclopedic entries available from Wikipedia using cosine similarity. Search results are diversified accordingly.

Finally, in the last few years the specific structure of the Web has been exploited to perform diversification, as proposed by Ma, Lyu, and King (2010), who make use of Markov random walks on query-URL bipartite graphs, and Chandar and Carterette

(2010), who cluster search results by exploiting the links in Web pages in order to identify the subtopics of the returned documents.

2.5 Word Sense Induction

A fifth solution to the query ambiguity issue is Word Sense Induction (WSI), namely, the automatic discovery of word (i.e., query) senses from raw text (see Navigli [2009, 2012] for a survey). WSI allows us to go beyond the surface similarity of Web snippets (which hampers the performance of Web search result clustering) by dynamically acquiring an inventory of senses of the input query. The core idea is to then use these query senses to cluster the Web snippets returned by a traditional search engine.

Very little work on this topic exists: Vector-based WSI was successfully shown to improve bag-of-words ad hoc Information Retrieval (Schütze and Pedersen 1995) and preliminary studies (Udani et al. 2005; Chen, Zaïane, and Goebel 2008) have provided interesting insights into the use of WSI for Web search result clustering. A more recent attempt at automatically identifying query meanings is based on the use of hidden topics (Nguyen et al. 2009). In this approach, however, topics (estimated from a universal data set) are query-independent and thus their number needs to be established beforehand. In contrast, we aim to cluster snippets on the basis of a dynamic and finer-grained notion of sense.

An exploratory study on ten query words has shown that the majority of relevant uses of query words can be identified using graph-based WSI (Véronis 2004). In the present work we take this preliminary finding to the next level, by studying the impact of several graph-based WSI algorithms on a large scale and by integrating them into a Web search result clustering framework. As a result, we are able not only to perform an end-to-end evaluation of WSI approaches, but also to compare them with traditional search result clustering techniques, which instead lack explicit semantics for the query meanings.

2.6 Aspect Identification

Over recent years a line of research has been developed in the field of Information Retrieval that makes use of query logs and clickthrough information to identify and model the aspects of a given query in terms of the user intents for that query. Aspects can be identified by exploiting those queries in the past that enabled the user to retrieve documents that are close to the current input query (Wang and Zhai 2007). A different approach aims, instead, at extracting related queries from query logs as candidate aspects and discarding duplicate and redundant aspects using search results. Wikipedia InfoBoxes are used to cluster candidate aspects into classes (Wu, Madhavan, and Halevy 2011). Latent aspects of queries can also be extracted from query reformulations within historical search session logs (Wang, Chakrabarti, and Punera 2009). More recently, a topic modeling approach based on query logs and click data has been proposed that aims at discovering generic aspects pervading manually fixed categories of named entities (Xue and Yin 2011). The implicit user-specific aspect of a query can be obtained from short query log sessions of other users using a Markov logic learning model. This results in the documents that best model the user's intentions when entering a query (Mihalkova and Mooney 2009). Finally, a semi-supervised approach has recently been applied to create class labels that are later assigned to latent clusters of queries using a Hierarchical Dirichlet Process (Reisinger and Pasca 2011).

This line of research has some points of contact with WSI, but also important differences:

- Most important, aspect identification aims at discriminating between very fine-grained facets of a given query, such as those of rental, pricing, and accidents of a car, in contrast to WSI whose goal is that of inducing different meanings of the given query, such as car as a motor vehicle, railroad car, song, novel, or even primitive in the LISP programming language. In this respect, the two tasks are complementary, because once WSI has discovered the different senses of a query, then one can apply aspect identification to detect subsenses of each meaning.
- Much work based on query logs and click data requires reliable statistics, which are not always available in all languages. WSI relies instead on raw text corpora, which can easily be obtained for any language. This difference also holds for custom search engines not working on the Web, which might not have enough statistics from their users, but could instead resort to raw (domain) corpora.
- Privacy and availability issues are often mentioned in connection with query logs and clickthrough data, therefore making research on this topic hard to replicate and evaluate objectively, especially in comparison with other systems.

The framework presented in this article focuses on the ambiguity of queries at the meaning level, leaving the further application of aspect identification techniques to future work, in the hope that the previously mentioned issues of privacy and availability will somehow be mitigated.

3. Semantically Enhanced Search Result Clustering

Web search result clustering is usually performed in three main steps:

1. Given a query q, a search engine is used to retrieve a list of results $R = (r_1, \ldots, r_n)$.
2. A clustering $\mathcal{C} = (C_1, \ldots, C_m)$ of the results in R is obtained by means of a clustering algorithm.
3. The clusters in $\mathcal{C}$ are optionally labeled with an appropriate algorithm (e.g., Zamir and Etzioni 1998; Carmel, Roitman, and Zwerdling 2009) for visualization purposes.

First, we preprocess the set R of search results returned by the search engine (Section 3.1). Next, to inject semantics into search result clustering, we propose improving Step 2 by means of a WSI algorithm: Given a query q, we first dynamically induce, from a text corpus, the set of word senses of q (Section 3.2); next, we cluster the Web results on the basis of the word senses previously induced (Section 3.3). We show our framework in Figure 1.

3.1 Preprocessing of Web Search Results

As a result of submitting our query q to a search engine, we obtain a list of relevant search results $R = (r_1, \ldots, r_n)$. In order to make this list usable by a clustering algorithm,

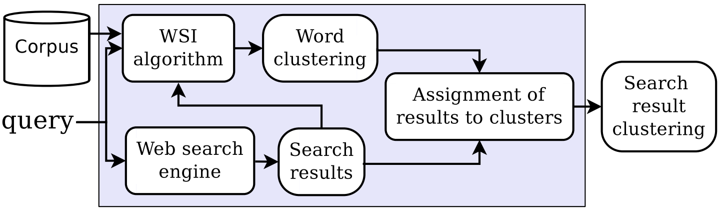

Figure 1
The workflow of semantically enhanced Web search result clustering.

each result r_i is processed by means of four steps aimed at transforming it into a bag of words b_i:

1. We obtain the snippet s_i corresponding to the result r_i.
2. We apply tokenization to s_i, thus splitting the string into tokens and setting them to lowercase.
3. We augment the current token set with multi-word expressions obtained by compounding subsequent word tokens up to ϕ words (a parameter whose tuning is described later in Section 4.1.4). The terms in the resulting token set are lemmatized using WordNet as reference lexicon. We remove tokens that are not in the WordNet lexicon (e.g., *the*, *esa*).
4. We remove the stopwords (e.g., *get*, *on*, *be*, *as*) and the target query words (e.g., *snow*, *leopard*, *snow leopard*) from the token set.

An example of the application of the four steps to a snippet returned for the query *snow leopard* is shown in Table 3. As a result of this process, we obtain a list of bags of words $B = (b_1, \ldots, b_n)$, where b_i is the bag of words of the search result r_i.

3.2 Graph-Based Word Sense Induction

The next step is to dynamically discover the senses of the input query q and provide a representation for them that will later be used for semantically clustering the snippets preprocessed in the previous step. WSI algorithms are unsupervised techniques aimed at automatically identifying the set of senses denoted by a word. These methods induce word senses from text by clustering word occurrences on the basis of the idea that a

Table 3
Processing steps for one of the search results of the query *snow leopard*.

step	output
initial snippet	“Get the facts on snow leopards. Endangered Species Act (ESA): the snow leopard is listed as endangered”
tokenization	{ get, the, facts, on, snow, leopards, endangered, species, act, esa, leopard, is, listed, as }
compounding and lemmatization	{ get, fact, on, snow, leopard, snow_leopard, endangered, species, endangered_species, act, be, listed, as }
stopword and query words removal	{ fact, endangered, species, endangered_species, act, listed }

given word—used in a specific sense—tends to co-occur with the same neighboring words (Harris 1954). Several approaches to WSI have been proposed in the literature (see Navigli [2009, 2012] for a survey), ranging from clustering based on context vectors (e.g., Schütze 1998) and word similarity (e.g., Lin 1998) to probabilistic frameworks (Brody and Lapata 2009), latent semantic models (Van de Cruys and Apidianaki 2011), and co-occurrence graphs (e.g., Widdows and Dorow 2002).

In our work, we chose to focus on approaches based on co-occurrence graphs for two reasons:

i) They have been shown to achieve state-of-the-art performance in standard evaluation tasks (Agirre et al. 2006b; Agirre and Soroa 2007; Korkontzelos and Manandhar 2010).

ii) Other approaches are either based on syntactic dependency statistics (Lin 1998; Van de Cruys and Apidianaki 2011), which are hard to obtain on a large scale for many domains and languages, or based on large matrix computation methods such as context-group discrimination (Schütze 1998), non-negative matrix factorization (Van de Cruys and Apidianaki 2011) and Clustering by Committee (Lin and Pantel 2002). Instead, in our approach we aim to exploit the relational structure of word co-occurrences with lower requirements (i.e., using just a stopword list, a lemmatizer, and a compounder, cf. Section 3.1), assuming that the semantics of a word are represented by means of a co-occurrence graph whose vertices are co-occurrences and whose edges are co-occurrence relations.

We therefore integrated the following algorithms into our framework:

- **Curvature clustering** (Dorow et al. 2005), an algorithm based on the participation ratio of words in graph triangles, that is, complete graphs with three vertices.
- **Squares, Triangles, and Diamonds (SquaT++)**, an algorithm that integrates two graph patterns previously exploited in the literature (Navigli and Crisafulli 2010), namely, squares and triangles, with a novel pattern called diamond.
- **Balanced Maximum Spanning Tree Clustering (B-MST)**, an extension of a WSI algorithm based on the calculation of a Maximum Spanning Tree (Di Marco and Navigli 2011) that aims at balancing the number of co-occurrences in each sense cluster.
- **HyperLex** (Véronis 2004), an algorithm based on the identification of hubs (representing basic meanings) in co-occurrence graphs.
- **Chinese Whispers** (Biemann 2006), a randomized algorithm that partitions the graph vertices by iteratively transferring the mainstream message (i.e., word sense) to neighboring vertices.

All of these graph algorithms for WSI consist of a common step, namely, co-occurrence graph construction (described in Section 3.2.1) and a second step, namely, the discovery of word senses, whose implementation depends on the specific algorithm adopted. We discuss the second phase of each algorithm separately (Section 3.2.2).

Table 4
Example co-occurrences of word w = *lion*.

word w'	$c(w')$	$c(w, w')$	$Dice(w, w')$
animal	213,414	5,109	0.2534
videogame	201,342	4,945	0.2042
mac	194,056	4,940	0.1568
africa	189,011	4,521	0.1961
feline	167,487	4,548	0.1472
cat	161,980	4,493	0.1214
savannah	159,693	3,535	0.1091
predator	145,239	3,643	0.1065
apple	140,670	3,261	0.1043
tiger	134,702	2,147	0.1024
technology	129,483	2,017	0.0097
software	113,045	1,846	0.0084
iPod	112,100	1,803	0.0070
simulation	93,899	1,367	0.0031

3.2.1 Step 1: 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[8] co-occurring with q, and E is the set of undirected edges, each denoting a co-occurrence between pairs of words in V. We harvest the statistics for co-occurring words V from a text corpus (we used two different corpora, see Section 4.1.2), which was previously tokenized and lemmatized.

First, for each word w we calculate the total number $c(w)$ of its occurrences and the number of times $c(w, w')$ that w occurs together with some word w' in the same context (to this end, we use the lemmas corresponding to inflected forms in the text). For instance, in Table 4, assuming w = *lion*, we show the absolute count $c(w')$ of some words (second column) together with the joint co-occurrence count $c(w, w')$ of words w' occurring with w = *lion* in the same context (third column). Note that the co-occurrences w' may refer to different senses of word w—for example, *africa* and *savannah* refer to the animal sense of *lion*, whereas *technology* and *software* to the operating system sense. Moreover, w' may be ambiguous itself in the context of w (e.g., *tiger* as either an animal or an operating system).

Second, we calculate the Dice coefficient to determine the strength of co-occurrence between any two words w and w':[9]

$$Dice(w, w') = \frac{2c(w, w')}{c(w) + c(w')}. \tag{1}$$

Table 4 reports the Dice coefficients in the fourth column for the example words. The rationale behind the use of the Dice coefficient, as opposed to, for example, a simple co-occurrence count such as $c(w, w')$, is that dividing by the average of the total

[8] Because our application (i.e., Web search result clustering) typically deals with nominal senses, and to avoid overly large graphs, we restrict our vocabulary to nouns only.

[9] We note that the Dice coefficient can have a probabilistic interpretation in terms of the conditional probabilities $P(w|w')$ and $P(w'|w)$ or, alternatively, the joint probability $P(w, w')$ and the marginal probabilities $P(w)$ and $P(w')$ (Smadja, McKeown, and Hatzivassiloglou 1996).

counts of the two words drastically decreases the ranking of words that tend to co-occur frequently with many other words (*home*, *page*, etc.).

Finally, we use the occurrence and co-occurrence counts just collected to construct the co-occurrence graph $G_q = (V, E)$ for the input query q. The pseudocode of our graph construction procedure is shown in Algorithm 1 and consists of the following steps:

a. **Initialization with snippet words** (*lines 1–2*)**:** Initially we set V to contain all the content words from the bags of words obtained from the snippet results of query q, that is, $V := \bigcup_{j=1}^{n} b_j$, where b_j is the bag of words corresponding to the search result $r_j \in R$ as obtained after the preprocessing step (see Section 3.1). We also set $E := \emptyset$, that is, the edge set is initially empty.

b. **Adding first-order co-occurrences** (*lines 3–5*)**:** We augment V with the highest-ranking words co-occurring with query q in the selected text corpus, that is, those words w for which the following equations are satisfied:

$$\begin{cases} \dfrac{c(q,w)}{c(q)} & \geq \delta \\ Dice(q,w) & \geq \delta' \end{cases} \tag{2}$$

where δ and δ' are experimentally tuned thresholds (cf. Section 4.1.4).

c. **Adding second-order co-occurrences** (*lines 6–11*)**:** Optionally, we create an auxiliary copy $V^{(0)}$ of V. For each word $w \in V^{(0)}$ we augment V with those words w' which are strongly related to w in the text corpus. In other words we add w' to V if both Equations (2) are satisfied for the pair of words w and w'.

Algorithm 1 The graph construction algorithm.

Input: query q, the bag of words $(b_1, \cdots, b_n)$ for q
Output: a graph $G_q = (V, E)$

```
 1: $V := \bigcup_{j=1}^{n} b_j$
 2: $E := \emptyset$
 3: for each word $w$ in the corpus
 4:     if $c(q,w)/c(q) \geq \delta$ and $Dice(q,w) \geq \delta'$ then
 5:         $V := V \cup \{w\}$
 6: if $second\_order$ = true then
 7:     $V^{(0)} := V$
 8:     for each word $w \in V^{(0)}$
 9:         for each word $w'$ in the corpus
10:             if $c(w,w')/c(w) \geq \delta$ and $Dice(w,w') \geq \delta'$ then
11:                 $V := V \cup \{w'\}$
12: for each $(w,w') \in V \times V$ s. t. $w \neq w'$
13:     if $Dice(w,w') \geq \theta$ then
14:         $E := E \cup \{\{w,w'\}\}$
15: remove all disconnected vertices from $V$
16: return $G_q = (V,E)$
```

d. **Creating the co-occurrence graph** (*lines 12–15*)**:** For each pair of words $(w, w') \in V \times V$, we add the corresponding edge $\{w, w'\}$ to E with weight $Dice(w, w')$ if the following condition is satisfied:

$$Dice(w, w') \geq \theta \tag{3}$$

where θ is a confidence threshold for the co-occurrence relation. Note that we use a threshold δ' to select which vertices to add to the graph G_q (Step [b]) and we use a potentially different threshold θ for the selection of which edges to add to G_q. Finally, we remove from V all the disconnected vertices (i.e., those with degree 0).

As a result of this algorithm a co-occurrence graph G_q for the query q is produced. Consider again the target word *lion* and let us assume that the words in Table 4 are the only co-occurrences of *lion*. In Figure 2 we show the execution of the four steps of our graph construction algorithm for the input query *lion*, assuming

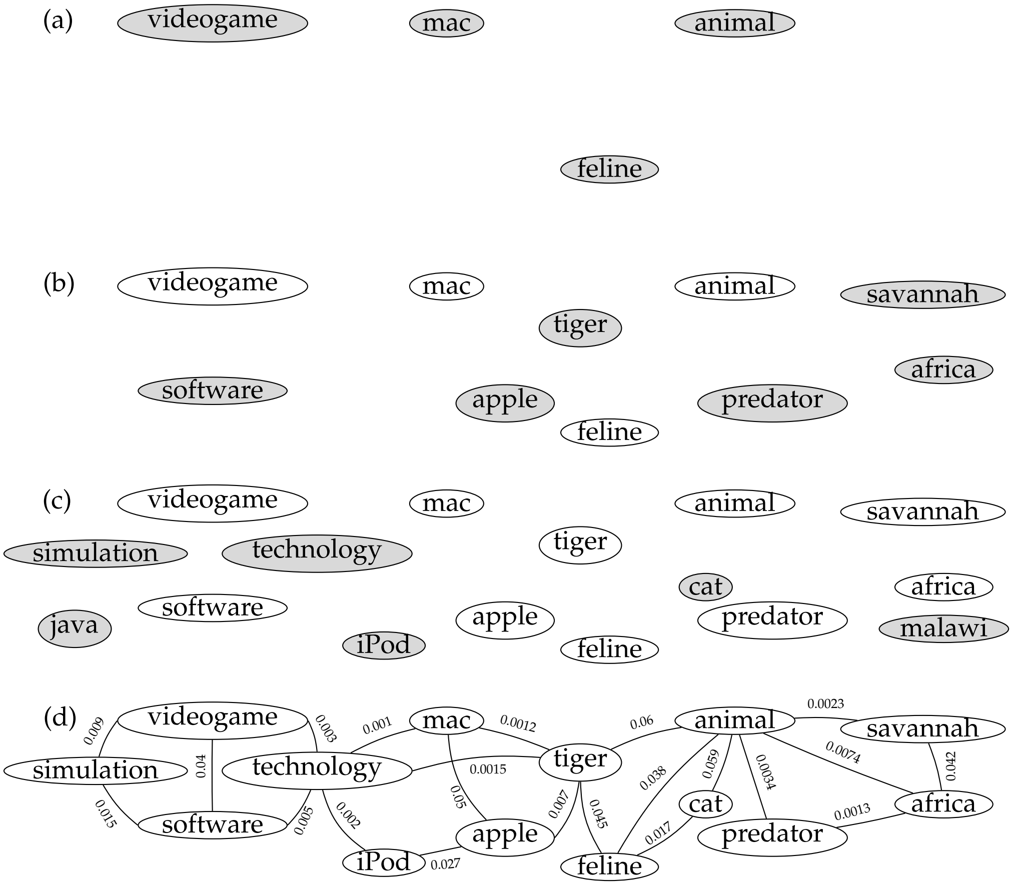

Figure 2
Graph construction for the *lion* example: (a) initializing V with the snippets words (lines 1–2); (b) adding first-order co-occurrences to V (lines 3–5); (c) adding second-order co-occurrences to V (lines 6–11); (d) adding the edges corresponding to strong relations and removing disconnected vertices (lines 12–15).

$\delta = 0.38$, $\delta' = \theta = 0.003$, and $c(lion) = 350,727$. First, we initialize the graph with the words in the snippets returned for *lion* (Figure 2a), next we add the words co-occurring with the query (Figure 2b), then second-order co-occurrences, that is, words co-occurring with those just added to the graph (Figure 2c), and finally we add those edges between word pairs whose Dice value is above a threshold (Figure 2d).

3.2.2 Step 2: Sense Discovery. All the graph-based WSI algorithms that we implemented in our framework are designed to discover the senses of an input term, which in our specific application is the query q. This process of meaning discovery is carried out through the use of the relational and structural information contained in the co-occurrence graph we have just created. In fact, a co-occurrence graph $G_q = (V, E)$ for a query q contains: (i) vertices $w \in V$ corresponding to words highly related to q, and (ii) edges $e \in E$ representing co-occurrence relations between vertices (i.e., words) in V. The key idea behind graph-based WSI is to obtain a partition $S = (S_1, \ldots, S_m)$ of G_q such that each component $S_i = (V_i, E_i)$ contains structurally (i.e., semantically) related vertices (i.e., words). In other words, each vertex set V_i is intended to contain only words related to a specific sense of q. As a result S is the **sense inventory** for the query q and each S_i is a sense cluster.

We now introduce each graph-based WSI algorithm in detail.

Curvature. The curvature algorithm aims at quantifying how strongly the neighbors of a vertex are related to each other. To measure this degree of correlation, the *curvature* coefficient for a vertex w is calculated as follows:

$$curv(w) = \frac{\text{\# triangles } w \text{ participates in}}{\text{\# triangles } w \text{ could participate in}} \tag{4}$$

where a triangle is a cycle of length 3. The numerator of Equation (4) is trivially calculated as the number of links between neighbors of w, and the denominator is calculated by counting all the possible pairs of neighbors. According to Equation (4), the curvature coefficient can assume values between 0 and 1. A vertex whose neighbors are highly connected (i.e., with a high value of curvature) is assumed to be part of a component that represents a specific meaning of the target query. Conversely, a vertex with low curvature acts as a connection between different meanings.

The curvature algorithm is designed to identify the meaning components by means of the removal of all vertices whose curvature is below a certain threshold σ. For example, we can attribute two different meanings to the word *Napoleon*, namely, a French emperor and an American city. By looking at the graph in Figure 3 we can easily find

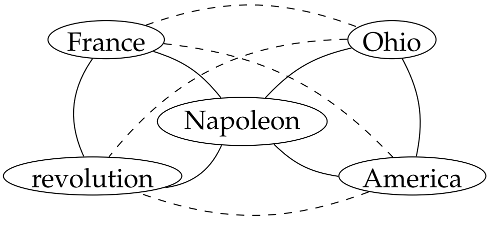

Figure 3
Example of curvature for *Napoleon*.

that *Napoleon* participates in two triangles (represented by continuous lines) and it potentially could also participate in four additional triangles (i.e., those including dashed lines). It follows that $curv(Napoleon) = \frac{2}{6} = 0.33$. The deletion of the vertex *Napoleon* results in two components (respectively, containing the vertices { *France*, *revolution* } and { *Ohio*, *America* }) representing the two mentioned meanings.

SquaT++. The curvature clustering algorithm is based on the hunch that local connectivity is correlated with meaning consistency. We take this idea to the next level by proposing a more elaborate local connectivity approach that exploits three different graph patterns, namely: triangles (i.e., cycles of length 3, like in curvature clustering), squares (i.e., cycles of length 4) and diamonds (i.e., graphs with 4 vertices and 5 edges, forming a square with a diagonal), hence the name SquaT++ (Squares, Triangles, and "more"). We determine the strength of the three patterns for a vertex w in the co-occurrence graph as follows:

$$Tri(w) = \frac{\text{\# triangles } w \text{ participates in}}{\text{\# triangles } w \text{ could participate in}} \tag{5}$$

$$Sqr(w) = \frac{\text{\# squares } w \text{ participates in}}{\text{\# squares } w \text{ could participate in}} \tag{6}$$

$$Dia(w) = \frac{\text{\# diamonds } w \text{ participates in}}{\text{\# diamonds } w \text{ could participate in}} \tag{7}$$

where w is a vertex. Then we linearly combine the three measures as follows:

$$SquaT{+}{+}(w) = \alpha \cdot Tri(w) + \beta \cdot Sqr(w) + \gamma \cdot Dia(w) \tag{8}$$

where $\alpha + \beta + \gamma = 1$. Similarly to the curvature algorithm, the sense clusters are obtained by removing all those vertices whose SquaT++ value is below a threshold σ. In Figure 4(a) we show in bold the vertices selected for removal, and in Figure 4(b) the sense clusters obtained after removal, namely: { *videogame*, *simulation*, *software* }, { *iPod*, *apple*, *mac* }, and { *cat*, *animal*, *predator*, *africa*, *savannah* }.

SquaT++ is a generalization of the curvature algorithm in that: (i) it uses the triangle pattern to calculate curvature, and (ii) it disconnects the graph using the same algorithm as curvature. SquaT++ is a novel algorithm, however, that extends the previously proposed SquaT (Navigli and Crisafulli 2010), based on triangles and squares, by introducing a new pattern, namely, the diamond, whose clustering coefficient is linearly combined with the other two. Moreover, in our experiments we tested two versions of SquaT++: the traditional one in which the coefficient is calculated on vertices (like in Equation (8)), and a variant calculated on edges. Our hunch here is that removing low-ranking edges rather than vertices might produce more informative clusters, because no word is removed from the original graph. In what follows, we refer to the vertex version as SquaT++$_V$ and to the variant on edges as SquaT++$_E$, and we refer to the general algorithm as SquaT++.

B-MST. A more global approach to the identification of sense components is the Balanced Maximum Spanning Tree (B-MST), which is based on the computation of

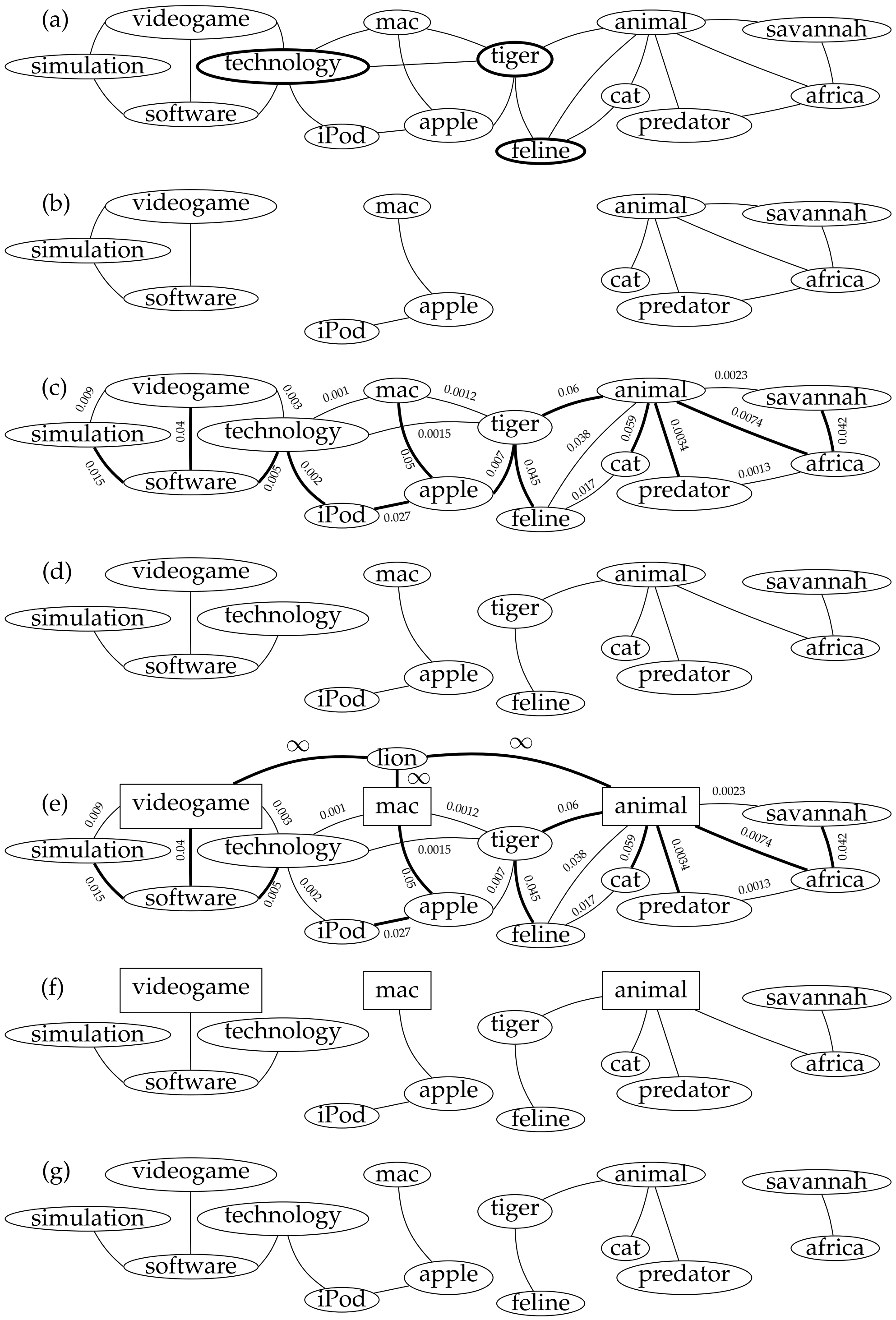

Figure 4
The *lion* example: (a) SquaT++ selection and (b) removal of edges below the threshold; (c) B-MST spanning tree calculation and (d) edge removal; (e) HyperLex hub selection and (f) identification of word senses; (g) Chinese Whispers cluster creation.

the Maximum Spanning Tree (MST) of the co-occurrence graph. Cluster meanings are identified by iteratively removing the edges which represent structurally weak relations, i.e., those with lower weight in the MST. The procedure is as follows:

- Eliminate from G_q all vertices whose degree is 1.
- Calculate the maximum spanning tree T_{G_q} of the graph G_q (e.g., the bold edges in Figure 4(c) represent the maximum spanning tree for our initial graph).
- The original MST algorithm for WSI, proposed by Di Marco and Navigli (2011), iteratively eliminates the minimum-weight edge $e \in T_{G_q}$ whose degree ≥ 2, until N connected components (i.e., word clusters) are obtained or there are no more edges to eliminate. The problem with this approach is that it can generate unbalanced clusters (i.e., a few very large clusters and several small clusters); for this reason we developed the B-MST variant which calculates an appropriate cluster mean cardinality[10] and removes an edge $e \in T_{G_q}$ if its elimination does not lead to connected components with cardinality less than half of the calculated mean value. This additional constraint prevents the creation of very small clusters, while at the same time avoiding artificial equal-size clusters.

Following our *lion* example, and assuming that the value of the only parameter of B-MST (i.e., the maximum number N of meanings to be identified) is set to 3, we obtain the clusters in Figure 4(d).

HyperLex. Another option for sense discovery is that of HyperLex, which identifies the most interconnected vertices in the graph G_q, called **hubs**. Each hub acts as the "root" of a specific component of G_q and, correspondingly, a meaning of the target query q.

First, a list L of the vertices w' of the graph G_q is created and sorted by their absolute count $c(w')$ in decreasing order. Each vertex $w' \in L$ is then selected as hub if it satisfies the following conditions:

$$\begin{cases} \dfrac{degree(w')}{\max_{w'' \in V} degree(w'')} \geq \sigma \\ \dfrac{\sum_{\{w',w''\} \in E} Dice(w', w'')}{degree(w')} \geq \sigma' \end{cases} \tag{9}$$

that is, the normalized degree of vertex w' and the average weight of the edges incident on w' must be, respectively, above the thresholds σ and σ'. Once it has been selected, the hub and all its neighbors are removed from L so as to avoid neighboring vertices from also being selected as hubs. The hub selection process stops when the next vertex in the sorted list does not satisfy either of the Equations (9) or if the list L is empty.

As an example, consider the co-occurrence graph in Figure 2(d). A list of the vertices in the graph is created, sorted by $c(w')$, as shown in Table 4. For the purpose of our

[10] We calculate the mean cardinality of a cluster by dividing the total number of vertices in the graph by the maximum number N of clusters that we want to obtain.

example, let us assume $\sigma = 0.5$ and $\sigma' = 0.015$. The first hub to be selected is *animal*. All its neighbors (*tiger*, *feline*, *cat*, *predator*, *africa*, and *savannah* in our example) are also removed from the list. The next hub is *videogame* (its neighbors *simulation*, *software*, and *technology* are also removed from the list). The last hub is *mac*; after the removal of its neighbor from the list (*apple*) the last vertex to be examined is *iPod*, which cannot be selected as hub because it does not satisfy the second condition of Equation (9). The selected hubs are shown as rectangles in Figure 4(e).

Once the hub selection process is complete, the target query q is added to the set of vertices V of graph G_q and each hub is connected to q with an infinite-weight edge (see vertex *lion* and its edges added to the graph in Figure 4(e)). Then, a maximum spanning tree T_q of the graph is calculated starting from vertex q (see the bold edges in Figure 4(e)). As a result, T_q will include all the infinite-weight edges from q to its direct descendants, namely, the hubs. Vertex q is then removed from the graph so that each subtree rooted at a hub in T_q represents a word sense for the target query q (see Figure 4(f)). In our example, three clusters are produced: { *videogame*, *simulation*, *software*, *technology* }, { *mac*, *apple*, *iPod* }, and { *animal*, *feline*, *tiger*, *cat*, *predator*, *africa*, *savannah* }. Note that, in our example, HyperLex and SquaT++ found the same meanings for the query word *lion* (namely, the animal, the operating system, and the videogame), but produced different clusters (e.g., HyperLex assigns the word *tiger* to the animal cluster whereas SquaT++ removes it from the graph). Finally, notice that in HyperLex the number of senses is dynamically chosen on the basis of the co-occurrences of q and the algorithm's thresholds.

An alternative approach to hub selection as performed in HyperLex consists of using the PageRank algorithm to sort the vertices of the co-occurrence graph and choose the best ranking ones as hubs of the target word (Agirre et al. 2006b). Given that the performance of this variant is comparable to that of HyperLex, in this work we focus on the original version of the induction algorithm.

Chinese Whispers. All the previously presented algorithms work in a top–down fashion, that is, they iteratively remove edges or vertices from an initial co-occurrence graph until a number of partitions are obtained. The last algorithm we consider, called Chinese Whispers, works, instead, bottom–up. The pseudocode, shown in Algorithm 2, consists of the following two steps:

1. First, the algorithm assigns a distinct class i to each vertex v_i and creates a clustering C containing the singleton clusters C_i (lines 1–4 of the algorithm).
2. Second, a series of iterations is performed aimed at merging the clusters (lines 5–11). Specifically, at each iteration the algorithm analyzes each vertex v in random order and assigns it to the majority class among those associated with its neighbors. In other words, it assigns each vertex v to the class c that maximizes the sum of the weights of the edges $\{u, v\}$ incident on v such that c is the class of u, according to the following formula:

$$class(v) := \operatorname*{argmax}_{c} \sum_{\substack{\{u,v\} \in E(G_q) \\ s.t.\ class(u)=c}} Dice(u, v) \tag{10}$$

Algorithm 2 The Chinese Whispers algorithm.

```
Input: a graph $G_q = (V, E)$ to be clustered
Output: a clustering $C$ of the vertices in $V$

 1: for each $v_i \in V$
 2:     $class(v_i) := i$
 3:     $C_i := \{v_i\}$
 4: $C := \{C_i : i = 1, \ldots, |V|\}$
 5: repeat
 6:     $C' := C$
 7:     for each $v \in V$, randomized order
 8:         $class(v) := \operatorname{argmax}_c \sum_{\substack{\{u,v\} \in E(G_q) \\ s.t.\ class(u)=c}} Dice(u, v)$
 9:     for each $i$ do $C_i := \{v \in V : class(v) = i\}$
10:     $C := \{C_i : C_i \neq \emptyset\}$
11: until $C \neq C'$
12: return $C$
```

As soon as an iteration produces no change in the clustering (line 11), the algorithm stops and outputs the final clustering (line 12). In contrast to the previous algorithm, Chinese Whispers is parameter-free. Figure 4(g) shows an output example for this algorithm on the *lion* co-occurrence graph.

3.3 Clustering of Web Search Results

We are now ready to semantically cluster our Web search results R, which we previously transformed into bags of words B (cf. Section 3.1). To this end we use the automatically discovered senses for our input query q (cf. Section 3.2). We adopt different measures, each of which calculates the similarity between a bag of words $b_i \in B$ and the sense clusters $\{S_1, \ldots, S_m\}$ acquired as a result of Word Sense Induction.

Given a result b_i, the sense cluster closer to b_i will be selected as the most likely meaning of r_i. Formally:

$$Sense(r_i) = \begin{cases} \operatorname{argmax}_{j=1,\ldots,m} sim(b_i, S_j) & \text{if } \max_{j=1,\ldots,m} sim(b_i, S_j) > 0 \\ 0 & \text{else} \end{cases} \tag{11}$$

where $sim(b_i, S_j)$ is a generic similarity value between b_i and S_j (0 denotes that no sense is assigned to result r_i). As a result of sense assignment for each $r_i \in R$, we obtain a clustering $\mathcal{C} = (C_1, \ldots, C_m)$ such that:

$$C_j = \{r_i \in R : Sense(r_i) = j\} \tag{12}$$

that is, C_j contains the search results classified with the j-th sense of query q.

We now present three different similarity measures between snippet bags of words and sense clusters (cf. Equation (11)), which we implemented in our framework.

Word Overlap. It calculates the size of the intersection between the two word sets:

$$sim_{WO}(b_i, S_j) = \frac{|b_i \cap V_j|}{|b_i|} \tag{13}$$

where $S_j = (V_j, E_j)$ as defined in Section 3.2.2.

Degree Overlap. It calculates the sum of the degrees in the co-occurrence graph component of S_j of the snippet's words in b_i:

$$sim_{DO}(b_i, S_j) = \frac{\sum_{w \in b_i \cap V_j} degree(w, S_j)}{|b_i| \cdot |E_j|} \tag{14}$$

where $degree(w, S_j)$ is the number of edges incident on w in the S_j component of the co-occurrence graph.

Token Overlap. The third measure is similar in spirit to Word Overlap, but takes into account each token occurrence in the snippet bag of words b_i:

$$sim_{TO}(b_i, S_j) = \frac{\sum_{w \in b_i \cap V_j} c(w, r_i)}{\sum_{w \in b_i} c(w, r_i)} \tag{15}$$

where $c(w, r_i)$ is the number of occurrences of the word w in the result r_i.

3.4 Cluster Sorting

As a natural consequence of the different similarity values between snippet results and a given cluster, first, not all the snippets will have the same degree of relevance for the cluster, and second, the produced clusters will show a different "quality" depending on the relevance of the search results therein. We thus sort the clusters in our clustering $\mathcal{C}$ using a similarity-based notion of cluster "quality." For each cluster $C_j \in \mathcal{C}$, we determine its similarity with the corresponding meaning S_j by calculating the following formula:

$$avgsim(C_j, S_j) = \frac{\sum_{r_i \in C_j} sim(b_i, S_j)}{|C_j|} \tag{16}$$

The formula determines the average similarity between the bags of words b_i of the search results r_i in cluster C_j and the corresponding sense cluster S_j. The similarity function *sim* is the same as that stated in Equation (11) and defined in Section 3.3.

Finally, we rank the elements r_i within each cluster C_j by their similarity $sim(b_i, S_j)$. We note that the ranking and optimality of clusters can be improved with more sophisticated techniques (e.g., Crabtree, Gao, and Andreae 2005; Kurland 2008; Kurland and Domshlak 2008; Lee, Croft, and Allan 2008). This is beyond the scope of this article, however.

4. In Vivo Experiments: Web Search Result Clustering

We now present two extrinsic experiments aimed at determining the impact of WSI when integrated into Web search result clustering. We first describe our experimental set-up (Section 4.1). Next, we present a first experiment focused on the quality of the output search result clusters (Section 4.2) and a second experiment on the degree of diversification of semantically enhanced versus non-semantic search result clustering algorithms (Section 4.3).

4.1 Experimental Set-up

4.1.1 Lexicon. In all our experiments our lexicon was given by the entire WordNet vocabulary (Miller et al. 1990; Fellbaum 1998) augmented with the set of queries in our test data sets.

4.1.2 Corpora. To calculate the co-occurrence strength between words we need a large corpus to extract co-occurrence counts and calculate the Dice values (cf. Equation (1)). To this end we performed separate experiments on two different corpora and constructed the corresponding co-occurrence databases:

- **Google Web1T** (Brants and Franz 2006): This corpus is a large collection of n-grams ($n = 1, \ldots, 5$)—namely, windows of n consecutive tokens—occurring in one terabyte of Web documents as collected by Google. We consider all the co-occurrences for lemmas which appear in the same n-gram (we applied the WordNet lemmatizer to obtain the canonical form of any word sequence).
- **ukWaC** (Ferraresi et al. 2008): This corpus was constructed by crawling the `.uk` domain and obtaining a large sample of Web pages that were automatically part-of-speech tagged using the TreeTagger tool. For this corpus we considered all the co-occurrences of WordNet lemmas that appear in the same sentence.

We selected these two corpora for their very different natures, namely: Google Web1T is a very large corpus, but with very narrow contexts (5-grams) with a minimum occurrence frequency; ukWaC represents a smaller portion of the Web, but with larger contexts. This enabled us to observe the behavior of WSI algorithms when co-occurrences were extracted from different kinds of textual source. In Table 5 we show examples of the contexts available in the two corpora for the same word (i.e., *lion*) and the content words that are found to co-occur with it (shown in italics in Table 5).

Table 5
Example of contexts for the word *lion* in the Web1T and ukWaC corpora (target word in bold, co-occurring words in italics).

corpus	context example
Web1T	*roar* of the **lion** in
ukWaC	*Wilson*'s *Zoo* and its *sad* **lion** had given *way* to the *brave attempt* to *create* an *early* "*Safari Park*".

Table 6
The pseudoword data set.

	pseudoword
1	pizza*blog
2	banana*plush
3	kalashnikov*mollusk*sky
4	hurricane*glue*modem
5	pistol*stair*yacht*semantics
6	potassium*razor*walrus*calendula
7	monarchy*archery*google*locomotive*beach
8	hyena*helium*soccer*ukulele*wife
9	human*orchid*candela*colosseum*movie*guitar
10	journey*harmonica*vine*mustache*rhino*police
11	glossary*river*dad*kitchen*aikido*geranium*italy
12	microbe*hug*ship*skull*beer*giraffe*mathematics

4.1.3 Tuning Set. Given that our graph construction step and our WSI algorithms have parameters, we created a data set to perform tuning. In order to fix the parameter values independently of our application we created this data set by means of pseudowords (Schütze 1992; Yarowsky 1993). A pseudoword is an ambiguous artificial word created by concatenating two or more monosemous words. Each monosemous word represents a meaning of the pseudoword. For example, given the words *pizza* and *blog* we can create the pseudoword *pizza*blog*. The list of pseudowords we used is reported in Table 6.

The powerful property of pseudowords is that they enable the automatic construction of sense-tagged corpora with virtually no effort. In fact, we automatically created our tuning data set as follows:

1. First, we collected the top 100 results retrieved by Yahoo! for each meaning (i.e., monosemous word) of the pseudoword (e.g., *pizza* and *blog* for *pizza*blog*).
2. We created a set of 100 snippets for the "pseudoword" query (e.g., *pizza*blog*) by selecting snippets from each meaning of the pseudoword in a number that was proportional to their total occurrence count. For instance, if *pizza* and *blog* occur, respectively, 73,000 and 27,000 times in the reference corpus (e.g., ukWaC), we selected 73 snippets from *pizza* and 27 from *blog*. As a result we simulated the distribution of the two senses of the pseudoword within the retrieved snippets.
3. Finally, within each of the 100 snippets, we replaced each monosemous word occurrence (e.g., *pizza* and *blog*) with the pseudoword itself (i.e., *pizza*blog*). As a result we obtained a set of 100 snippets for each ambiguous pseudoword.

4.1.4 Parameters. We used our tuning set to select, first, the optimal values of the parameters needed to perform graph construction, and, second, to choose the parameter values specific to each graph-based WSI algorithm. To find the best configurations we performed tuning by combining the three evaluation measures of Adjusted Rand Index, Jaccard Index, and F1 (introduced in Section 4.2.1).

Table 7
The optimal parameter values for graph creation obtained as a result of tuning.

	Web1T						ukWaC					
parameter	Curvature	SquaT++$_V$	SquaT++$_E$	B-MST	HyperLex	Chinese Whispers	Curvature	SquaT++$_V$	SquaT++$_E$	B-MST	HyperLex	Chinese Whispers
max comp. length (ϕ)	2	2	2	2	2	2	2	2	2	2	2	2
min co-occurr. (δ)	5E-2	5E-2	5E-2	5E-2	5E-2	2E-1	2E-1	5E-1	2E-1	2E-1	2E-1	2E-1
min Dice (δ')	1E-2	1E-2	1E-2	1E-2	5E-2	5E-2	1E-4	1E-4	1E-2	1E-2	1E-4	5E-2
min edge weight (θ)	9E-4	9E-4	9E-4	9E-4	9E-4	9E-4	6E-3	3E-3	3E-3	3E-3	7E-3	3E-3
co-occurrence order	1	1	1	1	1	1	1	1	1	1	1	1

Graph construction. Because all our WSI algorithms draw on the co-occurrence graph, we first tuned the parameters for graph construction for each of the two corpora (cf. Section 3.2.1), namely: the maximum length of the compounds extracted from the corpus (ϕ), the minimum number of co-occurrences (δ) and minimum Dice value (δ') for vertex addition, and the minimum weight for a graph edge (θ) and vertex addition using first versus second-order co-occurrences. In Table 7 we show the values for these parameters that optimize the performance of each WSI algorithm on the two corpora.[11] In all our runs we used the Word Overlap as a similarity measure for Web search result clustering.

We observed that the optimal values for many of the parameters used for graph construction were stable across algorithms, whereas they changed across corpora due to the different scales of the two corpora. Instead, the maximum compound length and the co-occurrence order were fixed for all configurations. For the former we observed no performance increase with longer compound lengths. For the latter we found negligible improvements with second-order co-occurrences, at the cost, however, of increasing the size of the resulting graph exponentially. Given the large number of experiments that would be involved, we decided to avoid this additional workload and use first-order co-occurrences in all our experiments.

WSI algorithms. Next, for each graph-based WSI algorithm, we kept the given optimal values fixed for building the co-occurrence graphs for the tuning set queries, while varying the parameter values of the WSI algorithm, using Word Overlap as similarity measure for Web search result clustering. In Table 8 we show the optimal values for each algorithm when using Web1T (third column) and ukWaC (fourth column) to build the co-occurrence graph. Chinese Whispers is not shown as it is parameter-free (cf. Section 3.2.2). For SquaT++, together with the σ threshold, we also tuned the three coefficient values α, β, and γ, that is, we needed to find the best values for the coefficients in Equation (8). The optimal coefficient combinations are shown in Table 9 for SquaT++ on vertices and edges, when using the two corpora for graph construction. The values indicate that all the three graph patterns provide a positive contribution to the algorithm's performance, with the same coefficients for SquaT++ on vertices and

[11] To this end we used empirically chosen parameters for each WSI algorithm, while delaying the optimal choice of these parameters to the next paragraph.

Table 8
The WSI algorithms' parameters.

		Web1T	ukWaC
Curvature	removal threshold (σ)	0.25	0.35
SquaT++	vertex removal threshold (σ)	0.07	0.2
	edge removal threshold (σ)	0.2	0.25
B-MST	number of clusters (N)	4	4
HyperLex	min hub degree (σ)	0.05	0.06
	min edge weight (σ')	0.004	0.01

Table 9
Optimal values for the three graph patterns used in SquaT++.

	Web1T			ukWaC		
	α	β	γ	α	β	γ
SquaT++$_V$	0.34	0.16	0.50	0.34	0.50	0.16
SquaT++$_E$	0.34	0.16	0.50	0.34	0.50	0.16

edges. Interestingly, we observe that, whereas the contribution of triangles (weighted by α) is the same across corpora, the respective weights of squares (β) and diamonds (γ) are flipped. After inspection we found that the graphs obtained with Web1T are less interconnected than those produced with ukWac. Consequently, diamonds are sparser but more reliable in the Web1T setting, whereas they are much more frequent, and thus noisier, in the ukWaC setting.

4.1.5 Test Sets. We conducted our in vivo experiments on two test sets of ambiguous queries:

- **AMBIENT** (AMBIguous ENTries), a data set that contains 44 ambiguous queries.[12] The sense inventory for the meanings (i.e., subtopics)[13] of queries is given by Wikipedia disambiguation pages. For instance, given the *beagle* query, its disambiguation page in Wikipedia provides the meanings of dog, Mars lander, computer search service, beer brand, and so forth. The top 100 Web results of each query returned by the Yahoo! search engine were tagged with the most appropriate query senses according to Wikipedia (amounting to 4,400 sense-annotated search results). To our knowledge, this is currently the largest data set of ambiguous queries available on-line. In fact, other existing data sets, such as those from the TREC Interactive Tracks, are not focused on distinguishing the subtopics of a query.

[12] `http://credo.fub.it/ambient`.

[13] In the following we use the terms *subtopic* and *word sense* interchangeably. As stated in the Introduction, this work focuses on query disambiguation along the lines of Word Sense Induction and Disambiguation (Navigli 2009), rather than aspect identification, which concerns subtle distinctions within the same meaning of a query.

Table 10
Statistics on the AMBIENT and MORESQUE data sets.

data set	queries	queries by length				average
		1	2	3	4	subtopics
AMBIENT	44	35	6	3	0	17.9
MORESQUE	114	0	47	36	31	6.6

- **MORESQUE** (MORE Sense-tagged QUEry results), a data set that we developed as an integration of AMBIENT following guidelines provided by its authors.[14] In fact, our aim was to study the behavior of Web search algorithms on queries of different lengths, ranging from one to four words. The AMBIENT data set, however, is composed in the main of one-word queries. MORESQUE provides dozens of queries of length 2, 3, and 4, together with the top 100 results from Yahoo! for each query annotated precisely as was done in the AMBIENT data set. We decided not to discontinue the use of Yahoo! mainly for homogeneity reasons.

Wikipedia has already been used as a sense inventory by, among others, Bunescu and Pasca (2006), Mihalcea (2007), and Gabrilovich and Markovitch (2009). Santamaría, Gonzalo, and Artiles (2010) have investigated in depth the benefit of using Wikipedia as the sense inventory for diversifying search results, showing that Wikipedia offers much more sense coverage for search results than other resources such as WordNet.

We report the statistics on the composition of the two data sets in Table 10. Given that the snippets could possibly be annotated with more than one Wikipedia subtopic, we also determined the average number of subtopics per snippet. This amounted to 1.01 for AMBIENT and 1.04 for MORESQUE for snippets with at least one subtopic annotation. We can thus conclude that multiple subtopic annotations are infrequent. Finally, we analyzed how the different subtopics are distributed over the snippet results for each query. To do this we calculated the standard deviation of the subtopic population for each individual query, which we show in Figure 5. We observed a considerable difference in the standard deviations of shorter and longer queries (e.g., between those from the AMBIENT data set [from 1 to 44 in the figure] and the MORESQUE data set [from 45 to 158]). We further calculated the average standard deviation over the two data sets' queries, obtaining 6.5 for AMBIENT and 13.1 for MORESQUE. Therefore we anticipate that the longer the query length, the more unbalanced will be the distribution of its subtopics over the top-ranking results.

In line with previous experiments on search result clustering, our data set does not contain monosemous queries for two reasons: (i) we are interested in queries with multiple meanings, and (ii) monosemous queries would increase the performance of our experiments because no diversification would be needed for them.

4.1.6 Systems. We performed a comparison of our semantically enhanced search result clustering systems with nonsemantic ones.

[14] http://lcl.uniroma1.it/moresque.

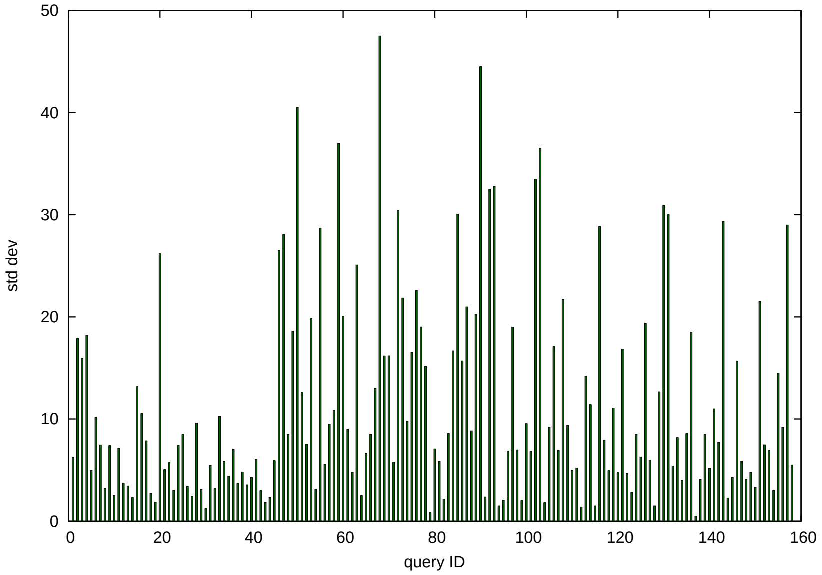

Figure 5
Standard deviations for the subtopic population of the AMBIENT queries (1–44) and the MORESQUE queries (45–158).

Semantically enhanced systems. We integrated our graph-based WSI algorithms (Curvature, SquaT++, B-MST, HyperLex, and Chinese Whispers; cf. Section 3.2) into our search result clustering framework. We tested each algorithm when combined with any of the snippet-to-sense similarity measures introduced in Section 3.3.

Nonsemantic systems. We compared our semantically enhanced systems with four Web clustering engines, namely:

- **Lingo** (Osinski and Weiss 2005): A Web clustering engine implemented in the Carrot open-source framework[15] that clusters the most frequent phrases extracted using suffix arrays.
- **Suffix Tree Clustering (STC)** (Zamir and Etzioni 1998): The original Web search clustering approach based on suffix trees.
- **KeySRC** (Bernardini, Carpineto, and D'Amico 2009): A state-of-the-art Web clustering engine built on top of STC with part-of-speech pruning and dynamic selection of the cut-off level of the clustering dendrogram.
- **Yippy**[16] (formerly Clusty): A state-of-the-art metasearch engine developed by Vivísimo aimed at clustering search results into meaningful topics.

For Lingo and STC we used the Carrot implementation which we integrated into our framework. Conversely, for Yippy we used the on-line output provided by the Web search engine.

[15] `http://project.carrot2.org`.
[16] `http://search.yippy.com`.

4.1.7 Baselines. We compared the four systems against three baselines:

- **Singletons:** Each snippet is clustered as a separate singleton (i.e., the cardinality of the resulting clustering $\mathcal{C}$ is $|\mathcal{C}| = |R|$).
- **All-in-one:** All the snippets are clustered into a single cluster (i.e., $|\mathcal{C}| = 1$).
- **Wikipedia clustering:** Given an input query q, we apply Equation (13) to match the bag of content words of each snippet against that of each Wikipedia page representing a meaning of q (we use the disambiguation page of q as its sense inventory). The snippet is then added to the cluster corresponding to the best-matching Wikipedia page. Given q, we obtain a clustering whose size is determined by the number of meanings in the Wikipedia disambiguation page of q.

The first two baselines help us determine whether the evaluation measures have a bias towards very small (singletons) or big clusters (all-in-one). The third baseline, based on Wikipedia, is a tough one in that—in contrast to our systems—it relies on a predefined sense inventory (which is the same as that used in the manual classification of the test set) to cluster the snippets. Consequently the baseline does not "induce" the senses, but just classifies (or labels) each snippet with the best-matching Wikipedia sense of the input query.

4.2 Experiment 1: Evaluation of the Clustering Quality

4.2.1 Evaluation Measures. In this first experiment our goal is to evaluate the quality of the output produced by our search result clustering systems. Unfortunately, the clustering evaluation problem is a notably hard issue, and one for which there exists no unequivocal solution. Many evaluation measures have been proposed in the literature (Rand 1971; Zhao and Karypis 2004; Rosenberg and Hirschberg 2007; Geiss 2009, inter alia) so, in order to get exhaustive results, we tested three different clustering quality measures, namely, Adjusted Rand Index, Jaccard Index, and F1-measure, which we introduce hereafter. Each of these measures $M(\mathcal{C}, \mathcal{G})$ calculates the quality of a clustering $\mathcal{C}$, output for a given query q, against the gold standard clustering $\mathcal{G}$ for that query. We then determine the overall results on the entire set of queries Q in the test set according to the measure M by averaging the values of $M(\mathcal{C}, \mathcal{G})$ obtained for each single test query $q \in Q$.

Adjusted Rand Index. Given a gold standard clustering $\mathcal{G}$, the Rand Index (RI; Rand 1971) of a clustering $\mathcal{C}$ is a measure of clustering agreement commonly used in the literature, calculated as follows:

$$RI(\mathcal{C}, \mathcal{G}) = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \qquad (17)$$

where TP is the number of true positives (i.e., snippet pairs) that are in the same cluster both in $\mathcal{C}$ and $\mathcal{G}$, TN is the number of true negatives (i.e., pairs which are in different clusters in both clusterings), and FP and FN are, respectively, the number of false positives and false negatives. For the gold standard $\mathcal{G}$ we use the clustering induced by the sense annotations provided in our data sets for each snippet (i.e., each cluster contains the snippets manually associated with a particular Wikipedia page, that is, subtopic, of the query).

Rand Index determines the percentage of snippet pairs that are in the same configuration in both $\mathcal{C}$ and $\mathcal{G}$, but its main weakness is that it does not take chance into account. In fact, the expected value of the RI of two random clusterings is not a constant value (e.g., 0). This issue is addressed by the Adjusted Rand Index (ARI; Hubert and Arabie 1985), which corrects the RI for chance agreement and makes it vary according to expectation:

$$ARI(\mathcal{C},\mathcal{G}) = \frac{RI(\mathcal{C},\mathcal{G}) - E(RI(\mathcal{C},\mathcal{G}))}{\max RI(\mathcal{C},\mathcal{G}) - E(RI(\mathcal{C},\mathcal{G}))} \tag{18}$$

where $E(RI(\mathcal{C},\mathcal{G}))$ is the expected value of the RI. Given two clusterings $\mathcal{C} = (C_1, \ldots, C_m)$ and $\mathcal{G} = (G_1, G_2, \ldots, G_g)$, we first quantify the degree of overlap between $\mathcal{C}$ and $\mathcal{G}$ using the contingency table reported in Table 11, where n_{ij} denotes the number of objects in common between G_i and C_j (i.e., $n_{ij} = |G_i \cap C_j|$) and a_i and b_j represent, respectively, the number of objects in G_i and C_j. Now, Equation (18) can be reformulated as follows (Steinley 2004):

$$ARI(\mathcal{C},\mathcal{G}) = \frac{\sum_{ij}\binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{N}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{N}{2}} \tag{19}$$

Differently from the original RI (which ranges between 0 and 1), the ARI ranges between -1 and $+1$ and is 0 when the index equals its expected value. Given the issues with RI, in our experiments we focused on ARI.

Jaccard Index. The ARI compares a clustering $\mathcal{C}$ with a gold standard $\mathcal{G}$ both in terms of the snippets occurring in the same cluster (TP) and those which are assigned to different clusters (TN). There are typically many TN in a clustering, however; therefore this measure tends to overweight the usefulness of snippets placed in different clusters. The Jaccard Index (JI) is a measure that addresses this issue. JI is calculated as follows:

$$JI(\mathcal{C},\mathcal{G}) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{20}$$

In fact, in contrast to RI (cf. Equation (17)), neither the numerator nor the denominator of JI include the TN term.

Table 11
Contingency table for the clusterings $\mathcal{G}$ and $\mathcal{C}$.

$\mathcal{C}$ $\mathcal{G}$	C_1	C_2	$\cdots$	C_m	**Sums**
G_1	n_{11}	n_{12}	$\cdots$	n_{1m}	a_1
G_2	n_{21}	n_{22}	$\cdots$	n_{2m}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
G_g	n_{g1}	n_{g2}	$\cdots$	n_{gm}	a_g
Sums	b_1	b_2	$\cdots$	b_m	N

F1-Measure. The ARI and the JI calculate the clustering quality using snippet pairs as the basic unit. Instead, a clustering $\mathcal{C}$ can be evaluated by focusing on the precision of the single clusters and the topics recalled by them, that is, we evaluate $\mathcal{C}$ according to its precision (P) and recall (R) against a gold standard $\mathcal{G}$. Precision determines how accurately the clusters of $\mathcal{C}$ represent the topics in the gold standard $\mathcal{G}$, and recall measures how accurately the topics in $\mathcal{G}$ are covered by the clusters in $\mathcal{C}$.

The precision of a cluster $C_j \in \mathcal{C}$ can be calculated as follows (Crabtree, Gao, and Andreae 2005):

$$P(C_j) = \frac{|C_j^t|}{|C_j|} \tag{21}$$

where t is the majority topic in C_j for a given query,[17] and C_j^t is the set of snippets in C_j which are tagged with subtopic t in the gold standard $\mathcal{G}$. The recall of a topic t is, instead, calculated as:

$$R(t) = \frac{|\bigcup_{C_j \in \mathcal{C}^t} C_j^t|}{n_t} \tag{22}$$

where $\mathcal{C}^t$ is the subset of clusters of $\mathcal{C}$ whose majority topic is t, and n_t is the number of snippets tagged with subtopic t in the gold standard. The total precision and recall of the clustering $\mathcal{C}$ are then calculated as:

$$P = \frac{\sum_{C_j \in \mathcal{C}} P(C_j)|C_j|}{\sum_{C_j \in \mathcal{C}} |C_j|}; \quad R = \frac{\sum_{t \in \mathcal{T}} R(t) n_t}{\sum_{t \in \mathcal{T}} n_t} \tag{23}$$

where $\mathcal{T}$ is the set of subtopics in the gold standard $\mathcal{G}$ for the given query. The two values of P and R are then combined into their harmonic mean, namely, the F1 measure (van Rijsbergen 1979):

$$F1(\mathcal{C}, \mathcal{G}) = \frac{2PR}{P+R} \tag{24}$$

Note that, in contrast with ARI, in calculating precision and recall we do not consider untagged gold standard snippets.

4.2.2 Results and Discussion. We show the results of the WSI algorithms in Table 12. With few exceptions, the results obtained on the two corpora are comparable. SquaT++, which extends Curvature with the Square and Diamond patterns, obtains higher performance. Although integrating three different graph patterns is beneficial, the difference between using edges and vertices to do so is mostly marginal.

The first important insight is that the best results, shown in bold in Table 12, are consistent across corpora and similarity measures (i.e., WO, DO, and TO), thus showing the robustness of the WSI algorithms when co-occurrences are extracted from different textual sources. The pairwise evaluation measures (i.e., ARI and JI), however, rank the

[17] The majority topic for a cluster $C_j \in \mathcal{C}$ is the topic t for which there exists the maximum number of snippets in C_j tagged with t.

Table 12
Results of graph-based WSI algorithms with Web1T and ukWaC and various similarity measures (Word Overlap, Degree Overlap, and Token Overlap). The reported measures are Adjusted Rand Index (ARI), Jaccard Index (JI), and F1 (percentages). We also show the average number of clusters per query produced by each algorithm.

Algorithm	Sim.	Web1T				ukWaC			
		ARI	JI	F1	# cl.	ARI	JI	F1	# cl.
	WO	67.03	74.10	58.34	2.3	64.86	72.74	58.84	3.5
Curvature	DO	66.88	73.76	58.67	2.3	64.02	71.04	59.85	3.5
	TO	67.14	74.04	58.36	2.3	65.03	72.46	58.73	3.5
	WO	69.65	75.69	59.19	2.1	69.27	75.55	59.18	2.3
SquaT++$_V$	DO	69.21	75.45	59.19	2.1	68.73	75.14	59.75	2.3
	TO	69.67	75.69	59.19	2.1	69.33	75.55	59.23	2.3
	WO	**69.88**	**75.82**	59.39	2.7	**69.84**	**75.74**	59.70	3.9
SquaT++$_E$	DO	**69.63**	**75.74**	60.99	2.7	**69.86**	**75.35**	63.00	3.9
	TO	**69.88**	**75.83**	59.40	2.7	**69.86**	**75.70**	59.78	3.9
	WO	60.76	71.51	64.56	5.0	61.15	72.24	65.57	5.0
B-MST	DO	66.48	69.37	64.84	5.0	67.60	70.02	67.41	5.0
	TO	63.17	71.21	64.04	5.0	64.18	71.93	65.46	5.0
	WO	60.86	72.05	**65.41**	13.0	56.59	72.00	**70.69**	17.0
HyperLex	DO	66.27	68.00	**71.91**	13.0	65.92	67.31	**76.88**	17.0
	TO	62.82	70.87	**65.08**	13.0	61.64	70.61	**70.42**	17.0
	WO	67.75	75.37	60.25	12.5	68.77	75.45	59.66	6.5
Chinese Whispers	DO	65.95	69.49	70.33	12.5	67.86	72.34	66.16	6.5
	TO	67.57	74.69	60.50	12.5	68.97	75.28	59.79	6.5

WSI algorithms differently from the F1 measure. In fact, when we focus on pairwise evaluation measures, SquaT++ outperforms all other systems on both corpora, with Chinese Whispers ranking second. B-MST and HyperLex obtain lower results. When we look into the precision of the output clusters and the recall of the gold-standard topics (i.e., we calculate F1), however, we observe an inverse trend: HyperLex, B-MST and, to a lesser extent, Chinese Whispers achieve the best performance, whereas Curvature and SquaT++ obtain lower F1. This is because, assuming comparable precision, producing more clusters (as is done by HyperLex, B-MST, and Chinese Whispers) implies more chances to obtain higher recall, thus better diversifying among the topics of the retrieved search results. More specifically, B-MST and, especially, HyperLex benefit from the use of ukWaC in terms of F1 performance, with HyperLex gaining around 5% when moving from Web1T to ukWaC.

Finally, in most cases we observe negligible differences between the three different similarity measures (i.e., WO, DO, TO, cf. Section 3.3), with some exceptions concerning B-MST, HyperLex, and Chinese Whispers.

We now report the best results for our WSI algorithms in Table 13, compared against those of nonsemantic systems (i.e., Lingo, STC, and KeySRC, cf. Section 4.1.6) and our three baselines (i.e., all-in-one, singleton, and Wikipedia, cf. Section 4.1.7). For the WSI algorithms we show the results when using the WO measure, because, first, DO uses graph information and thus cannot be applied to nonsemantic systems, and, second, in

Table 13
A comparison between different search result clustering approaches (percentages). The best results for each of the three classes of algorithms is shown in bold.

	Algorithm	Web1T				ukWaC			
		ARI	JI	F1	#cl.	ARI	JI	F1	# cl.
WSI-based	Curvature	67.03	74.10	58.34	2.3	64.86	72.74	58.84	3.5
	SquaT++$_V$	69.65	75.69	59.19	2.1	69.27	75.55	59.18	2.3
	SquaT++$_E$	**69.88**	**75.82**	59.39	2.7	**69.84**	**75.74**	59.70	3.9
	B-MST	60.76	71.51	64.56	5.0	61.15	72.24	65.57	5.0
	HyperLex	60.86	72.05	**65.41**	13.0	56.59	72.00	**70.69**	17.0
	Chinese Whispers	67.75	75.37	64.25	12.5	68.77	75.45	59.66	6.5
SRC systems	Lingo	−0.53	36.36	16.73	2.0	−0.53	36.36	16.73	2.0
	STC	−7.90	**38.23**	14.96	2.0	−7.90	**38.23**	14.96	2.0
	KeySRC	**14.34**	27.77	**63.11**	18.5	**14.34**	27.77	**63.11**	18.5
Baselines	All-in-one	0.00	47.12	42.40	1.0	0.00	47.12	42.40	1.0
	Singleton	0.00	0.00	**68.17**	100.0	0.00	0.00	**68.17**	100.0
	Wikipedia	**13.83**	**56.02**	14.33	5.7	**13.83**	**56.02**	14.33	5.7

most cases (as remarked earlier) there are negligible differences between the two other similarity measures (i.e., WO and TO, see Table 12).

Our first finding here is that WSI-based search result clustering outperforms all other approaches across all evaluation measures on the two corpora, except for KeySRC and the singleton baseline when using the F1 measure. We note, however, that although KeySRC outperforms the WSI algorithms based on graph patterns in terms of F1, it attains very low ARI and JI results. Even worse, the singleton baseline produces trivial, meaningless clusterings, as measured by ARI and JI. The all-in-one baseline, instead, obtains non-zero JI (thanks to the true positives taken into account), but again zero ARI. Further, its F1 is lower than singleton, because of its lower recall. The Wikipedia baseline fares well compared with the other baselines in terms of ARI and JI, but achieves lower F1, again because of low recall. Finally, KeySRC consistently outperforms the other SRC systems in terms of ARI and F1.

In order to perform a fair comparison of our systems with Yippy we used a modified version of our test set that retains only the Yahoo! results that were also returned by Yippy. The average number of results over all queries in the resulting data set is 24.4, with a minimum and maximum number of 3 and 56 results per query, respectively.

We report the results on the reduced data set in Table 14. Among the classical search result clustering systems, Yippy performs worse in terms of ARI and JI. Instead, when we focus on the precision and recall of the output clusters, Yippy outperforms all other nonsemantic systems, while lagging behind all WSI algorithms (which use Web1T). One finding here is that, even in the presence of a smaller number of snippets per query, semantic systems perform best, whereas other approaches, which rely (like KeySRC) on the availability of a sufficient number of snippets, fall short.

4.3 Experiment 2: Evaluation of the Clustering Diversity

4.3.1 Evaluation Measure. Most of today's search engines return a flat list of search results. We thus performed a second experiment aimed at quantifying the impact of our Web search result clustering systems on flat-list search engines. In other words, our goal was

Table 14
A comparison of different SRC approaches, including Yippy (on a reduced version of the AMBIENT+MORESQUE data set; WSI systems use Web1T).

	Algorithm	ARI	JI	F1	# cl.
WSI-based	Curvature	63.21	75.22	64.86	2.1
	SquaT++$_V$	**64.25**	75.29	64.99	2.0
	SquaT++$_E$	64.13	75.96	65.30	2.3
	B-MST	53.72	76.51	70.82	5.0
	HyperLex	55.93	**76.63**	**72.04**	7.9
	Chinese Whispers	55.41	74.83	70.52	8.4
SRC systems	Lingo	−1.58	35.65	17.00	2.0
	STC	**7.91**	**38.34**	15.04	2.0
	KeySRC	−0.01	31.80	32.90	3.3
	Yippy	3.80	14.81	**64.52**	14.9
Baselines	All-in-one	0.00	45.66	48.30	1.0
	Singleton	0.00	0.00	**72.19**	24.4†
	Wikipedia	**10.00**	**52.05**	15.60	5.8

† Corresponding to the average number of snippet results in the reduced data set.

to determine how many different meanings of a query are covered in the top-ranking results shown to the user. One natural way of measuring such performance is given by **S-recall@K (Subtopic recall at rank *K*)** and **S-precision@r (Subtopic precision at recall *r*)** (Zhai, Cohen, and Lafferty 2003). S-recall@K counts the number of different subtopics retrieved for *q* in the top *K* results returned:

$$\text{S-recall@K} = \frac{|\cup_{i=1}^{K} subtopics(r_i)|}{m} \tag{25}$$

where $subtopics(r_i)$ is the set of subtopics manually assigned to the search result r_i and m is the number of subtopics for query q in the gold standard. In order to cut out some noise, we calculated the S-recall@K considering only the subtopics assigned to at least two snippets.

S-precision@r instead determines the ratio of different subtopics retrieved for q in the first K_r documents, where K_r is the minimum number of top results for which the system achieves recall r. Formally:

$$\text{S-precision@r} = \frac{|\cup_{i=1}^{K_r} subtopics(r_i)|}{K_r} \tag{26}$$

So whereas S-recall@K aims at determining the performance of a system at retrieving the largest number of topics for the query q in the K top-ranking results, S-precision@r quantifies the ratio of distinct subtopics covered by the minimal set of results returned for which the system obtains a specific recall r. Note that unambiguous queries would perform with S-precision@r = S-recall@K = 1 for all values of r and K.[18]

[18] Here again we focus on the polysemy of queries in the traditional (computational) linguistic sense. See Section 2.6 for a discussion.

Table 15
S-recall@K on all queries (percentages). The results for WSI algorithms are reported for both Web1T and ukWaC.

		K									
	System	3	4	5	6	7	8	9	10	15	20
Web1T	Curvature	48.2	53.5	57.1	60.5	64.6	67.4	69.4	72.6	**81.5**	**86.2**
	SquaT++$_V$	47.1	52.0	55.5	59.3	61.9	65.6	68.4	70.4	79.4	86.2
	SquaT++$_E$	47.6	51.9	56.2	59.6	62.6	64.5	67.0	69.0	78.5	84.4
	B-MST	49.1	**55.8**	59.4	62.5	65.6	67.8	70.0	72.3	80.0	85.5
	HyperLex	**50.4**	55.5	**60.5**	**63.4**	**66.2**	**69.3**	**71.2**	**72.9**	78.9	84.9
	Chinese Whispers	48.8	53.3	58.3	62.2	65.4	68.5	70.8	72.8	78.9	84.5
ukWaC	Curvature	47.2	51.8	56.8	59.5	62.5	65.4	67.0	68.4	76.3	83.4
	SquaT++$_V$	47.1	51.9	56.7	59.4	62.9	65.6	68.1	70.3	78.8	84.4
	SquaT++$_E$	47.9	51.2	55.1	58.6	61.5	64.8	67.8	69.6	78.9	84.9
	B-MST	49.9	55.3	61.0	63.9	66.9	70.7	73.7	75.6	83.3	87.5
	HyperLex	**51.1**	**59.3**	**64.6**	**67.3**	**71.3**	**73.9**	**74.9**	**76.6**	**83.4**	**87.6**
	Chinese Whispers	49.7	54.5	57.7	61.2	64.0	66.7	69.5	71.4	79.4	84.0
	KeySRC	39.5	46.1	48.7	51.4	54.3	57.1	59.6	61.7	68.2	72.5
	EP	36.1	41.4	44.6	50.8	53.5	55.0	57.1	59.2	67.9	**73.3**
	Yahoo!	**42.3**	**47.6**	**51.4**	**54.6**	**57.3**	**59.4**	**61.0**	**63.4**	**69.1**	**73.3**

These two measures are only suitable, however, for systems returning ranked lists (such as Yahoo! and Essential Pages). In order to apply them to search result clustering systems, we flatten each clustering to a list of search results. To do so, given a clustering $\mathcal{C} = (C_1, C_2, \ldots, C_m)$, we add to the initially empty list the first element[19] of each cluster C_j $(j = 1, \ldots, m)$; then we iterate the process by selecting the second element of each cluster C_j such that $|C_j| \geq 2$, and so on. The remaining elements returned by the search engine, but not included in any cluster of $\mathcal{C}$, are appended to the bottom of the list in their original order.

4.3.2 Results and Discussion. The results in terms of S-recall@K are shown in Table 15. The first key finding is that, independently of the adopted corpus for graph construction, each of the WSI algorithms outperforms all nonsemantic systems, including the state-of-the-art search resulting clustering engine (KeySRC), Essential Pages (see Section 2.4), and the Yahoo! baseline. This result provides strong evidence that inducing senses for a given ambiguous query is beneficial for the diversification of snippet results. Not all WSI algorithms perform the same, however: In fact, we observe that exploiting local graph patterns (as done by Curvature and SquaT++) typically leads to worse results compared with other graph-based approaches. We do not observe substantial differences between Curvature and SquaT++ on edges and vertices. We hypothesize that the lack of significant difference in the diversification performance of the three pattern-based WSI algorithms is due to the lower number of clusters they produce (in the order of around two to three clusters, cf. Table 12).

[19] Recall that the snippets within a cluster are sorted by relevance, cf. Section 3.4.

The best performance on both corpora is, instead, obtained by HyperLex, tailed by B-MST and Chinese Whispers. HyperLex is more complex and requires the tuning of many parameters (Agirre et al. 2006a), however. Interestingly, we observe that the ranking of WSI algorithms according to S-recall@K closely matches that obtained with the F1 measure for clustering quality. Finally, among the nonsemantic alternatives, Yahoo! fares well and surpasses KeySRC and EP.

To get futher insights into the performance of the best semantic systems, in Figure 6 we graphed the values of S-recall@K for representative systems, namely, B-MST,

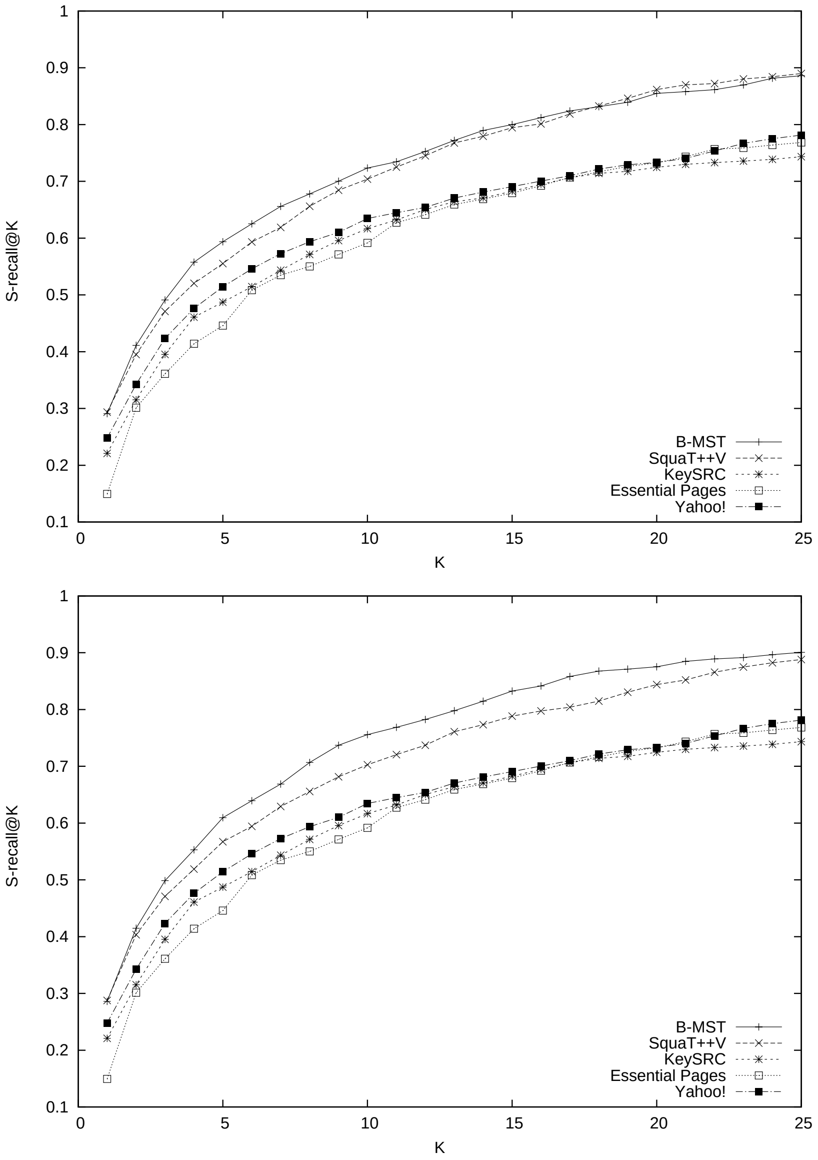

Figure 6
S-recall@K trend for B-MST, SquaT++$_V$, KeySRC, Essential Pages, and Yahoo! on Web1T (top) and ukWaC (bottom).

SquaT++$_V$, and the three nonsemantic systems. The results shown in the figure are those obtained with Web1T (top) and ukWaC (bottom). We can see that SquaT++$_V$ lags behind B-MST especially for low values of K. As also remarked previously, Yahoo! tends to perform better than KeySRC.

As regards S-precision@r, shown in Table 16, again all WSI algorithms outperform nonsemantic systems. The general trend observed for S-recall@K is confirmed here: HyperLex generally achieves the best values of S-precision@r, with good performance for all other semantic systems. All in all, HyperLex has the best balance between recall and precision, with better diversification performance on ukWaC, and therefore looks like the most suitable choice. B-MST, however, is much simpler and requires just one parameter (i.e., the number of clusters), which can also be exploited by the user to get finer- or coarser-grained search result groups. As was previously done for S-recall@K, we also graphed the values of S-precision@r for the same representative systems in Figure 7.

5. In Vitro Experiment: Evaluating the Induced Senses

Although the primary aim of this work was to demonstrate a relevant, end-to-end application of sense discovery techniques, we performed an additional in vitro experiment aimed at verifying the quality of the discovered senses independently of the task in which they are used.

When performing in vitro evaluations, no single intrinsic measure provides a clear hint as to which algorithm performs best (Manandhar et al. 2010). In fact, some measures favor large clusters, whereas others are based on the expectaction that the WSI algorithm will discover more fine-grained sense distinctions. To provide further insights into the clusters produced by our graph-based WSI algorithms, we performed

Table 16
S-precision@r on all queries (percentages). The results for WSI algorithms are reported for both Web1T and ukWaC.

				r		
	System	50	60	70	80	90
Web1T	Curvature	46.0	33.8	30.7	25.2	21.9
	SquaT++$_V$	45.9	34.5	28.3	24.0	21.0
	SquaT++$_E$	42.8	35.3	29.1	23.6	20.4
	B-MST	43.6	35.3	29.3	25.7	21.6
	HyperLex	46.5	**38.0**	**31.3**	**26.5**	**22.5**
	Chinese Whispers	**49.4**	35.2	28.9	24.2	21.9
ukWaC	Curvature	37.9	33.2	27.4	24.3	20.4
	SquaT++$_V$	45.0	34.4	28.8	25.5	22.2
	SquaT++$_E$	42.0	34.0	29.3	25.2	20.5
	B-MST	49.3	36.7	**33.5**	26.3	22.5
	HyperLex	**51.4**	**40.0**	32.4	**27.6**	**22.6**
	Chinese Whispers	45.1	36.6	30.1	24.5	20.5
	KeySRC	29.3	22.3	17.7	15.4	12.0
	EP	33.6	24.9	**18.9**	**16.1**	**13.2**
	Yahoo!	**36.1**	**25.7**	18.7	15.5	12.6

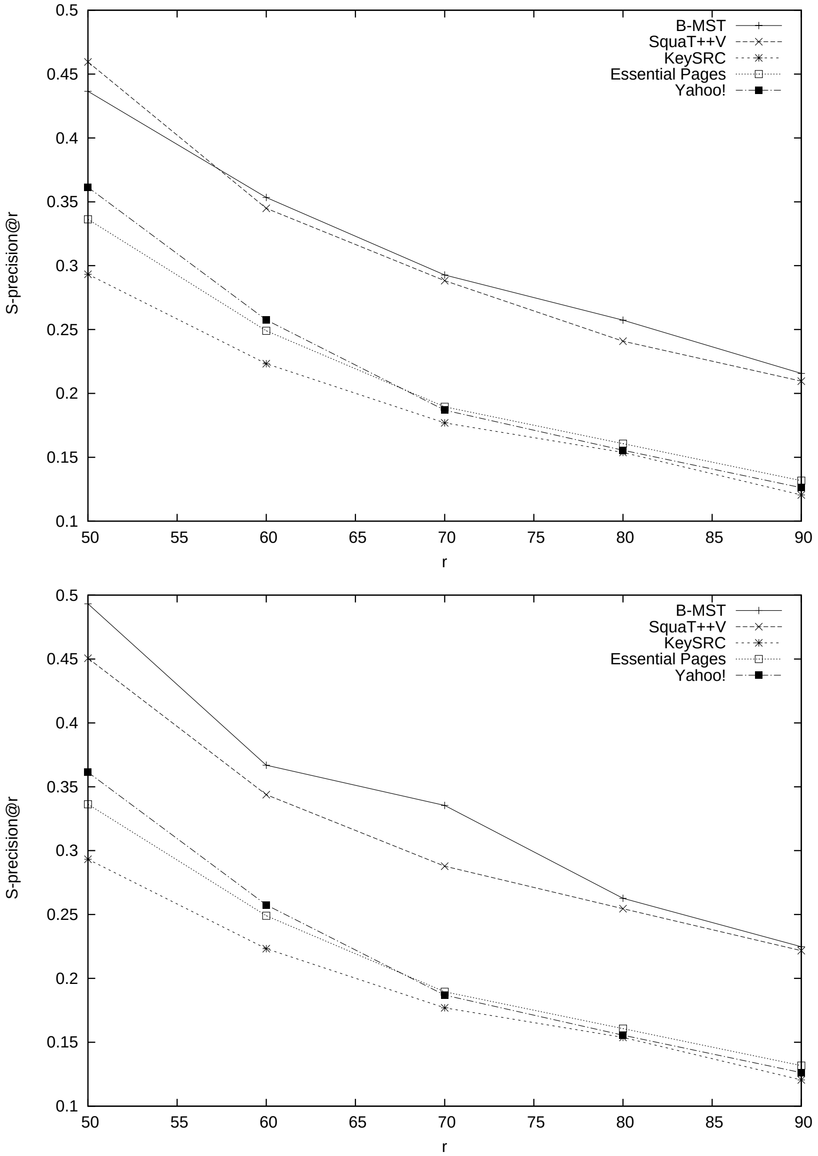

Figure 7
S-precision@r trend for B-MST, SquaT++$_V$, KeySRC, Essential Pages, and Yahoo! on Web1T (top) and ukWaC (bottom).

a qualitative evaluation of the output clusters. To this end we randomly selected 17 queries from our query data set. For each query, we submitted in random order the output of three representative WSI algorithms on the ukWaC corpus, namely, Curvature, HyperLex, and B-MST, to five annotators.

We show an excerpt of the evaluation procedure for the query *excalibur* in Table 17. On the left side of the table we propose an example of an anonymized set of three clusterings (i.e., one for each algorithm, shown in columns 2–4) presented to our annotators. Each algorithm produced a group of clusters, each of which consisted of a set of words strictly related to the meaning conveyed by the cluster itself, as discussed in

Table 17
An example of the manual evaluation procedure for the query *excalibur*: We show a clustering triple proposed to the evaluator (left side) and an example of produced ranking (right side).

	Clustering A	Clustering B	Clustering C
Cluster 1	movie book casino	movie DVD video book	hotel movie review offer
Cluster 2	sword stone artillery	sword lineage stone	
Cluster 3	comic comic strip	hotel chicago casino	

rank	algorithm
1st	B
2nd	A
3rd	C

Section 3.2.2. The annotators were asked to rank the three clusterings according to their own preference (ties were allowed). On the right side of Table 17 we show an example of ranking for the three clusterings. In the example, clustering B was deemed to be more representative, because it better models three meanings of *excalibur*, namely: the film-novel meaning, the sword meaning, and the hotel casino meaning, whereas clustering A mixes the movie and the casino meaning within cluster 1, and, even worse, clustering C just provides a singleton cluster.

Finally, for each query, and for the entire set of 17 queries, we calculated the average ranking obtained by each WSI algorithm. The overall results are shown in Table 18 (last row): 1.7 for HyperLex, 1.8 for B-MST, and 2.4 for Curvature. This experiment corroborates the findings obtained from our extrinsic experiments: Curvature is the worst-ranking system (probably because of the low number of induced senses), whereas HyperLex and B-MST are more apt to discriminate between the meanings of an input query. It is worth noting that the annotators often assigned the same rank to the clusters produced by B-MST and HyperLex, confirming our extrinsic finding that the two algorithms tend to have a similar behavior, compared with local graph pattern WSI.

6. Time Performance Analysis

Finally, because we are interested in the real-world application of the WSI techniques we discussed, we decided to collect statistics about the execution times of each system on the AMBIENT and MORESQUE data sets. We carried out this performance analysis on a workstation using Sun Java 1.6 VM running on OpenSuse 11.4 (64 bit) with 16 GB PC3-15000 RAM, Intel Xeon E3-1240@3.30 GHz, and 1.5 TB hard disk space.

In the graph construction step, in common with all WSI algorithms, most (i.e., about 80%) of the computational load is due to the interaction with the database management system (DBMS, we used MySQL 5.1), and the remaining CPU time is used for populating the graph. On average constructing a co-occurrence graph takes 10–12 seconds per query. We note, however, that our algorithms were not engineered to work in an enterprise, possibly distributed, environment, with a commercial DBMS. Moreover, a fully engineered architecture might appropriately precalculate and cache the graphs concerning the most frequent queries.

Table 18
Results of the manual evaluations. The average scores assigned by the assessors to each one of the 17 queries are shown in columns 2, 3, and 4. In the last row we report the average results over all queries.

query id	B-MST	HyperLex	Curvature
1	2.0	1.0	3.0
2	1.4	2.0	2.4
3	2.2	1.6	2.2
4	1.4	2.4	2.2
5	2.6	2.0	1.4
6	1.2	1.8	3.0
7	2.2	1.4	2.1
8	2.2	2.4	1.4
9	1.8	2.4	1.6
10	1.8	1.6	2.2
11	1.6	1.4	3.0
12	1.6	1.8	2.6
13	1.8	1.2	3.0
14	1.8	1.2	3.0
15	1.8	1.8	2.0
16	1.6	1.4	3.0
17	1.8	1.2	3.0
all	1.8	1.7	2.4

The average time performance of WSI algorithms including sense discovery and snippet clustering (but excluding graph construction) are shown in Table 19, expressed in average number of seconds per query for both corpora. These numbers are compared with the time performance of nonsemantic systems (bottom part of the table).

We observe that, among pattern-based algorithms, SquaT++ has a high runtime cost, due to the heavy calculation of three different graph patterns. SquaT++$_E$ is particularly onerous in the presence of large amounts of edges, which is the case for ukWaC. Curvature, instead, has a lower cost, because the triangle pattern is less

Table 19
Execution times expressed in seconds. For WSI algorithms, the reported times include the sense discovery and snippet clustering steps and excludes the graph construction step.

	Algorithm	Web1T	ukWaC
WSI-based systems	Curvature	0.34	0.34
	SquaT++$_V$	28.98	14.45
	SquaT++$_E$	21.49	169.13
	B-MST	0.24	0.27
	HyperLex	0.16	0.13
	Chinese Whispers	0.28	0.35
SRC systems	Lingo	0.27	
	STC	0.20	
	KeySRC	1.00†	

† Estimated by the authors of KeySRC.

onerous to compute. Interestingly, the algorithms which we experimentally found to perform best (i.e., B-MST, HyperLex, and Chinese Whispers) have a much lower computational load compared with graph-pattern based algorithms. We found that HyperLex is particularly fast, with an average time of 0.1 seconds per query. Finally, we observe that the cost of the best WSI algorithms is not very far off that of nonsemantic SRC systems.

7. Conclusions

In this article we have presented a novel approach to Web search result clustering based on the automatic discovery of word senses from raw text. Key to our approach is the idea of, first, automatically inducing senses for the target query and, second, clustering the search results based on their semantic similarity to the word senses induced.

A sizeable body of work looking at the benefit of word senses for Web search already exists at the intersection between lexical semantics and information retrieval. That research, however, has focused almost exclusively on classical Word Sense Disambiguation, with contrasting and often inconclusive results. In this article, instead, we provide clear indication on the usefulness of a looser notion of sense to cope with ambiguous queries.

In fact, our experiments on data sets of queries of different lengths show that our approach outperforms all nonsemantic approaches to Web search result clustering. The main advantage of using Word Sense Induction lies in its dynamic production of word senses that cover both concepts (e.g., *beagle* as a specific breed of dog) and instances (e.g., *beagle* as a specific instance of a space lander). This is in contrast with static dictionaries such as WordNet that are typically used in Word Sense Disambiguation and which, by their very nature, mainly encode concepts.

Not only have we shown that graph-based WSI, when applied to search result clustering, surpasses its nonsemantic alternatives, but we have also provided an end-to-end evaluation framework that enables fair comparison of WSI algorithms. As a result, we are able to overcome many of the issues with the evaluation of clustering algorithms (von Luxburg, Williamson, and Guyon 2012), including the lack of a single unbiased intrinsic measure (Manandhar et al. 2010). Moreover, new WSI algorithms can be added at any time and compared with those already integrated into the framework. Building upon this, we are currently organizing a Semeval-2013 task for the extrinsic evaluation of WSI algorithms.[20] As of today, we are releasing a new data set of 114 ambiguous queries and 11,400 sense-annotated snippets.[21] Given the present paucity of ambiguous query data sets available (Sanderson 2008), we hope our data set will be useful in future comparative experiments.

Thanks to its modular structure, our framework can easily be extended in many other ways, including the addition of new snippet similarity measures, text corpora, query data sets, evaluation measures, and so on. Although our graphs are centered on words (as vertices), we are also interested in testing new graph construction procedures based on the use of collocations as vertices, as done by Korkontzelos and Manandhar (2010). Furthermore, the framework is independent of the target language, in that it just requires a large-enough corpus for co-occurrence extraction in that language and some basic tools for processing text (i.e., a stopword list, a lemmatizer, and a compounder).

[20] `http://www.cs.york.ac.uk/semeval-2013/task11/`.

[21] The data set is available at `http://lcl.uniroma1.it/moresque/`.

As future work, the framework might be integrated with distributional semantics models and techniques (Baroni and Lenci 2010; Erk, Padó, and Padó 2010; Mitchell and Lapata 2010; Boleda, im Walde, and Badia 2012; Clarke 2012; Silberer and Lapata 2012, inter alia).

Finally we note that, although in this article our framework was applied to polysemous queries only, nothing prevents it from being used to perform experiments at different levels of sense granularity. A qualitative evaluation of preliminary experiments in aspect identification (cf. Section 2.6), which requires the detection of very fine-grained subsenses of possibly monosemous queries, showed that WSI also seems to perform well in this task. Given the high number of monosemous queries submitted to Web search engines, we believe that further investigation in this direction may well reveal additional benefits of WSI for Web Information Retrieval.

Acknowledgments

The authors gratefully acknowledge the support of the ERC Starting Grant MultiJEDI no. 259234 and the CASPUR High-Performance Computing Grants 515/2011 and 118/2012. Thanks go to Google for providing the Web1T corpus for research purposes, Claudio Carpineto and Massimiliano D'Amico for producing the output of KeySRC and Essential Pages, and Stanislaw Osinski and Dawid Weiss for their help with Lingo and STC. Additional thanks go to Jim McManus and the three anonymous reviewers for their helpful comments.

References

Agirre, Eneko, David Martínez, Oier López de Lacalle, and Aitor Soroa. 2006a. Evaluating and optimizing the parameters of an unsupervised graph-based WSD algorithm. In *Proceedings of the 1st Workshop on Graph-Based Algorithms for Natural Language Processing*, pages 89–96, New York.

Agirre, Eneko, David Martínez, Oier López de Lacalle, and Aitor Soroa. 2006b. Two graph-based algorithms for state-of-the-art WSD. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 585–593, Sydney.

Agirre, Eneko and Aitor Soroa. 2007. UBC-AS: A graph based unsupervised system for induction and classification. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 346–349, Prague.

Agrawal, Rakesh, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. 2009. Diversifying search results. In *Proceedings of the 2nd International Conference on Web Search and Web Data Mining*, pages 5–14, Barcelona.

Baroni, Marco and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Basile, Pierpaolo, Annalina Caputo, and Giovanni Semeraro. 2009. Exploiting disambiguation and discrimination in Information Retrieval systems. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 03*, pages 539–542, Washington, DC.

Bennett, Paul N. and Nam Nguyen. 2009. Refined experts: Improving classification in large taxonomies. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 11–18, Boston, MA.

Bernardini, Andrea, Claudio Carpineto, and Massimiliano D'Amico. 2009. Full-subtopic retrieval with keyphrase-based search results clustering. In *Proceedings of 2009 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 206–213, Milan.

Biemann, Chris. 2006. Chinese whispers—an efficient graph clustering algorithm and its application to Natural Language Processing problems. In *Proceedings of the 1st Workshop on Graph-Based Algorithms for Natural Language Processing*, pages 73–80, New York.

Boleda, Gemma, Sabine Schulte im Walde, and Toni Badia. 2012. Modeling regular polysemy: A study on the semantic classification of Catalan adjectives. *Computational Linguistics*, 38(3):575–616.

Branson, S. and A. Greenberg. 2002. Clustering Web search results using suffix tree methods. In *Technical Report CS276A Final Project*, Stanford University.

Brants, Thorsten and Alex Franz. 2006. Web 1T 5-gram, ver. 1, ldc2006t13. In *Linguistic Data Consortium*, Philadelphia, PA.

Brody, Samuel and Mirella Lapata. 2009. Bayesian Word Sense Induction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 103–111, Athens.

Bruza, Peter, Robert McArthur, and Simon Dennis. 2000. Interactive Internet search: Keyword, directory and query reformulation mechanisms compared. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 280–287, Athens.

Bunescu, Razvan C. and Marius Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 9–16, Trento.

Carbonell, Jaime and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 335–336, Melbourne.

Carmel, David, Haggai Roitman, and Naama Zwerdling. 2009. Enhancing cluster labeling using Wikipedia. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 139–146, Boston, MA.

Carpineto, Claudio, Massimiliano D'Amico, and Andrea Bernardini. 2011. Full discrimination of subtopics in search results with keyphrase-based clustering. *Web Intelligence and Agent Systems*, 9(4):337–349.

Carpineto, Claudio, Stanislaw Osiński, Giovanni Romano, and Dawid Weiss. 2009. A survey of Web clustering engines. *ACM Computing Surveys*, 41(3):1–38.

Carpineto, Claudio and Giovanni Romano. 2004. Exploiting the potential of concept lattices for Information Retrieval with CREDO. *Journal of Universal Computer Science*, 10(8):985–1013.

Chandar, Praveen and Ben Carterette. 2010. Diversification of search results using webgraphs. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 869–870, Geneva.

Chapelle, Olivier, Yi Chang, and Tie-Yan Liu. 2011. Future directions in learning to rank. *Journal of Machine Learning Research—Proceedings Track*, 14:91–100.

Chen, Harr and David R. Karger. 2006. Less is more: Probabilistic models for retrieving fewer relevant documents. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 429–436, Seattle, WA.

Chen, Jiyang, Osmar R. Zaïane, and Randy Goebel. 2008. An unsupervised approach to cluster web search results based on word sense communities. In *Proceedings of 2008 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 725–729, Sydney.

Cheng, David, Santosh Vempala, Ravi Kannan, and Grant Wang. 2005. A divide-and-merge methodology for clustering. In *Proceedings of the 24th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 196–205, Baltimore, MD.

Clarke, Daoud. 2012. A context-theoretic framework for compositionality in distributional semantics. *Computational Linguistics*, 38(1):41–71.

Clough, Paul, Mark Sanderson, Murad Abouammoh, Sergio Navarro, and Monica Lestari Paramita. 2009. Multiple approaches to analysing query diversity. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 734–735, Boston, MA.

Crabtree, Daniel, Xiaoying Gao, and Peter Andreae. 2005. Improving Web clustering by cluster selection. In *Proceedings of 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 172–178, Compiegne.

Cutting, Douglass R., David R. Karger, Jan O. Pedersen, and John W. Tukey. 1992. Scatter/Gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318–329, Copenhagen.

Di Giacomo, Emilio, Walter Didimo, Luca Grilli, and Giuseppe Liotta. 2007. Graph visualization techniques for Web clustering engines. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):294–304.

Di Marco, Antonio and Roberto Navigli. 2011. Clustering Web search results with maximum spanning trees. In *Proceedings of the XIIth International Conference of the Italian Association for Artificial Intelligence*, pages 201–212, Palermo.

Dorow, Beate, Dominic Widdows, Katarina Ling, Jean-Pierre Eckmann, Danilo Sergi, and Elisha Moses. 2005. Using curvature and Markov clustering in graphs for lexical acquisition and word sense discrimination. In *Proceedings of the Meaning-2005 Workshop*, Trento.

Erk, Katrin, Sebastian Padó, and Ulrike Padó. 2010. A flexible, corpus-driven model of regular and inverse selectional preferences. *Computational Linguistics*, 36(4):723–763.

Fellbaum, Christiane, editor. 1998. *WordNet: An Electronic Database*. MIT Press, Cambridge, MA.

Ferraresi, Adriano, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating ukWaC, a very large Web-derived corpus of English. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4)*, pages 47–54, Marrakech.

Furnas, G. W., T. K. Landauer, L. M. Gomez, and S. T. Dumais. 1987. The vocabulary problem in human-system communication. *Commununications of ACM*, 30(11):964–971.

Gabrilovich, Evgeniy and Shaul Markovitch. 2009. Wikipedia-based semantic interpretation for natural language processing. *Journal of Artificial Intelligence Research (JAIR)*, 34:443–498.

Geiss, Johanna. 2009. Creating a gold standard for sentence clustering in multi-document summarization. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 96–104, Singapore.

Gelgi, Fatih, Hasan Davulcu, and Srinivas Vadrevu. 2007. Term ranking for clustering Web search results. In *Proceedings of 10th International Workshop on the Web and Databases*, Beijing, China.

Gonzalo, Julio, Anselmo Penas, and Felisa Verdejo. 1999. Lexical ambiguity and Information Retrieval revisited. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 195–202, College Park, MD.

Harris, Zellig. 1954. Distributional structure. *Word*, 10:146–162.

Hubert, L. and P. Arabie. 1985. Comparing partitions. *Journal of Classification*, 2(1):193–218.

Kamvar, Maryam and Shumeet Baluja. 2006. A large scale study of wireless search behavior: Google mobile search. In *Proceedings of the 2006 Conference on Human Factors in Computing Systems*, pages 701–709, Montréal.

Ke, Weimao, Cassidy R. Sugimoto, and Javed Mostafa. 2009. Dynamicity vs. effectiveness: Studying online clustering for Scatter/Gather. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 19–26, Boston, MA.

Kim, Sang-Bum, Hee-Cheol Seo, and Hae-Chang Rim. 2004. Information Retrieval using word senses: Root sense tagging approach. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 258–265, Sheffield.

Korkontzelos, Ioannis and Suresh Manandhar. 2010. UoY: Graphs of unambiguous vertices for word sense induction and disambiguation. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 355–358, Uppsala.

Krovetz, Robert and William B. Croft. 1992. Lexical ambiguity and Information Retrieval. *ACM Transactions on Information Systems*, 10(2):115–141.

Kurland, Oren. 2008. The opposite of smoothing: A language model approach to ranking query-specific document clusters. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 171–178, Singapore.

Kurland, Oren and Carmel Domshlak. 2008. A rank-aggregation approach to searching for optimal query-specific clusters. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 547–554, Singapore.

Lee, Kyung Soon, W. Bruce Croft, and James Allan. 2008. A cluster-based resampling method for pseudo-relevance feedback. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 235–242, Singapore.

Lin, Dekang. 1998. Automatic retrieval and clustering of similar words. In *Proceedings* *of the* 17*th* *International Conference on Computational Linguistics*, pages 768–774, Montreal.

Lin, Dekang and Patrick Pantel. 2002. Discovering word senses from text. In *Proceedings of the* 8*th* *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 613–619, Edmonton.

Liu, Shuang, Clement Yu, and Weiyi Meng. 2005. Word Sense Disambiguation in queries. In *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management*, pages 525–532, Bremen.

Liu, Tie-Yan, Yiming Yang, Hao Wan, Hua-Jun Zeng, Zheng Chen, and Wei-Ying Ma. 2005. Support vector machines classification with a very large-scale taxonomy. *SIGKDD Explorations*, 7(1):36–43.

Liu, Ying, Wenyuan Li, Yongjing Lin, and Liping Jing. 2008. Spectral geometry for simultaneously clustering and ranking query search results. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 539–546, Singapore.

Ma, Hao, Michael R. Lyu, and Irwin King. 2010. Diversifying query suggestion results. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, AAAI 2010, pages 1,399–1,404, Atlanta, GA.

Maarek, Yoelle, Ron Fagin, Israel Ben-Shaul, and Dan Pelleg. 2000. Ephemeral document clustering for Web applications. IBM Research Report RJ 10186. Haifa.

Manandhar, Suresh, Ioannis P. Klapaftis, Dmitriy Dligach, and Sameer S. Pradhan. 2010. SemEval-2010 task 14: Word sense induction & disambiguation. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 63–68, Uppsala.

Mandala, Rila, Takenobu Tokunaga, and Hozumi Tanaka. 1998. The use of WordNet in Information Retrieval. In *Proceedings of the COLING-ACL workshop on Usage of Wordnet in Natural Language Processing*, pages 31–37, Montréal.

Matsuo, Yutaka, Takeshi Sakaki, Kôki Uchiyama, and Mitsuru Ishizuka. 2006. Graph-based word clustering using a Web search engine. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 542–550, Sydney.

Mihalcea, Rada. 2007. Using Wikipedia for automatic Word Sense Disambiguation. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 196–203, Rochester, NY.

Mihalkova, L. and R. Mooney. 2009. Learning to disambiguate search queries from short sessions. In *Proceedings of Machine Learning and Knowledge Discovery in Databases (2)*, pages 111–127, Bled.

Miller, George A., R. T. Beckwith, Christiane D. Fellbaum, D. Gross, and K. Miller. 1990. WordNet: an online lexical database. *International Journal of Lexicography*, 3(4):235–244.

Mitchell, Jeff and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429.

Navigli, Roberto. 2009. Word Sense Disambiguation: A survey. *ACM Computing Surveys*, 41(2):1–69.

Navigli, Roberto. 2012. A quick tour of Word Sense Disambiguation, induction and related approaches. In *Proceedings of the 38th Conference on Current Trends in Theory and Practice of Computer Science*, pages 115–129, Spindleruv Mlýn.

Navigli, Roberto and Giuseppe Crisafulli. 2010. Inducing word senses to improve Web search result clustering. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 116–126, Boston, MA.

Navigli, Roberto and Simone Paolo Ponzetto. 2012. The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Ngo, Chi Lang and Hung Son Nguyen. 2005. A method of Web search result clustering based on rough sets. In *Proceedings of 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 673–679, Compiegne.

Nguyen, Cam-Tu, Xuan-Hieu Phan, Susumu Horiguchi, Thu-Trang Nguyen, and Quang-Thuy Ha. 2009. Web search clustering and labeling with hidden topics. *ACM Transactions on Asian Language Information Processing*, 8(3):1–40.

Osinski, Stanislaw and Dawid Weiss. 2005. A concept-driven algorithm for clustering search results. *IEEE Intelligent Systems*, 20(3):48–54.

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

Reisinger, Joseph and Marius Pasca. 2011. Fine-grained class label markup of search queries. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1,200–1,209, Portland, OR.

Rosenberg, Andrew and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 410–420, Prague.

Sanderson, Mark. 1994. Word Sense Disambiguation and Information Retrieval. In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 142–151, Dublin.

Sanderson, Mark. 2000. Retrieving with good sense. *Information Retrieval*, 2(1):49–69.

Sanderson, Mark. 2008. Ambiguous queries: Test collections need more sense. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 499–506, Singapore.

Santamaría, Celina, Julio Gonzalo, and Javier Artiles. 2010. Wikipedia as sense inventory to improve diversity in Web search results. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL 2010, pages 1,357–1,366, Uppsala.

Schütze, Hinrich. 1992. Dimensions of meaning. In *Proceedings of the 1992 ACM/IEEE Conference on Supercomputing*, pages 787–796, Los Alamitos, CA.

Schütze, Hinrich. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–124.

Schütze, Hinrich and Jan Pedersen. 1995. Information Retrieval based on word senses. In *Proceedings of SDAIR'95*, pages 161–175, Las Vegas, NV.

Silberer, Carina and Mirella Lapata. 2012. Grounded models of semantic representation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1,423–1,433, Jeju Island.

Smadja, Frank, Kathleen R. McKeown, and Vasileios Hatzivassiloglou. 1996. Translating collocations for bilingual lexicons: A statistical approach. *Computational Linguistics*, 22(1):1–38.

Song, Ruihua, Zhenxiao Luo, Jian-Yun Nie, Yong Yu, and Hsiao-Wuen Hon. 2009. Identification of ambiguous queries in Web search. *Information Processing and Management*, 45:216–229.

Steinley, Doug. 2004. Properties of the Hubert-Arabie adjusted Rand index. *Psychological Methods*, 9(3):386–396.

Stokoe, Christopher, Michael J. Oakes, and John I. Tait. 2003. Word Sense Disambiguation in Information Retrieval revisited. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 159–166, Toronto.

Swaminathan, Ashwin, Cherian V. Mathew, and Darko Kirovski. 2009. Essential pages. In *Proceedings of 2009 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 173–182, Milan.

Udani, Goldee, Shachi Dave, Anthony Davis, and Tim Sibley. 2005. Noun sense induction using Web search results. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 657–658, Salvador.

Van de Cruys, Tim and Marianna Apidianaki. 2011. Latent semantic word sense induction and disambiguation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1,476–1,485, Portland, OR.

van Rijsbergen, C. J. 1979. *Information Retrieval*, second edition. Butterworths, London.

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

von Luxburg, Ulrike, Robert C. Williamson, and Isabelle Guyon. 2012. Clustering: Science or art? *Journal of Machine Learning Research—Proceedings Track*, 27:65–80.

Voorhees, Ellen M. 1993. Using WordNet to disambiguate word senses for text retrieval. In *Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 171–180, Pittsburgh, PA.

Wang, Xuanhui, Deepayan Chakrabarti, and Kunal Punera. 2009. Mining broad latent query aspects from search sessions. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 867–876, Paris.

Wang, Xuanhui and ChengXiang Zhai. 2007. Learn from Web search logs to organize search results. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 87–94, Amsterdam.

Widdows, Dominic and Beate Dorow. 2002. A graph model for unsupervised lexical acquisition. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 1–7, Taipei.

Wu, Fei, Jayant Madhavan, and Alon Y. Halevy. 2011. Identifying aspects for Web-search queries. *Journal of Artificial Intelligence Research (JAIR)*, 40:677–700.

Xue, Gui-Rong, Dikan Xing, Qiang Yang, and Yong Yu. 2008. Deep classification in large-scale text hierarchies. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 619–626, Singapore.

Xue, Xiaobing and Xiaoxin Yin. 2011. Topic modeling for named entity queries. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 2009–2012, New York.

Yarowsky, David. 1993. One sense per collocation. In *Proceedings of the ARPA Workshop on Human Language Technology*, pages 266–271, Princeton, NJ.

Zamir, Oren and Oren Etzioni. 1998. Web document clustering: A feasibility demonstration. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 46–54, Melbourne.

Zamir, Oren, Oren Etzioni, Omid Madani, and Richard M. Karp. 1997. Fast and intuitive clustering of Web documents. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 287–290, Newport Beach, CA.

Zeng, Hua-Jun, Qi-Cai He, Zheng Chen, Wei-Ying Ma, and Jinwen Ma. 2004. Learning to cluster Web search results. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 210–217, Sheffield.

Zhai, ChengXiang, William W. Cohen, and John Lafferty. 2003. Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 10–17, Toronto.

Zhang, Benyu, Hua Li, Yi Liu, Lei Ji, Wensi Xi, Weiguo Fan, Zheng Chen, and Wei-Ying Ma. 2005. Improving Web search results using affinity graph. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR 2005, pages 504–511, Salvador.

Zhang, Xiaodan, Xiaohua Hu, and Xiaohua Zhou. 2008. A comparative evaluation of different link types on enhancing document clustering. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 555–562, Singapore.

Zhao, Ying and George Karypis. 2004. Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning*, 55(3):311–331.