Lang Resources & Evaluation (2018) 52:733–770
https://doi.org/10.1007/s10579-018-9415-1

ORIGINAL PAPER

A comparison of graph-based word sense induction clustering algorithms in a pseudoword evaluation framework

Flavio Massimiliano Cecchini[1] · Martin Riedl[2] · Elisabetta Fersini[1] · Chris Biemann[2]

Published online: 24 March 2018

Abstract This article presents a comparison of different Word Sense Induction (WSI) clustering algorithms on two novel pseudoword data sets of semantic-similarity and co-occurrence-based word graphs, with a special focus on the detection of homonymic polysemy. We follow the original definition of a pseudoword as the combination of two monosemous terms and their contexts to simulate a polysemous word. The evaluation is performed comparing the algorithm's output on a pseudoword's ego word graph (i.e., a graph that represents the pseudoword's context in the corpus) with the known subdivision given by the components corresponding to the monosemous source words forming the pseudoword. The main contribution of this article is to present a self-sufficient pseudoword-based evaluation framework for WSI graph-based clustering algorithms, thereby defining a new evaluation measure (TOP2) and a secondary clustering process (hyperclustering). To our knowledge, we are the first to conduct and discuss a large-scale systematic pseudoword evaluation targeting the induction of coarse-grained homonymous word senses across a large number of graph clustering algorithms.

✉ Flavio Massimiliano Cecchini
flavio.cecchini@unicatt.it

Martin Riedl
http://lt.informatik.uni-hamburg.de/

Elisabetta Fersini
fersini@disco.unimib.it

Chris Biemann
http://lt.informatik.uni-hamburg.de/

[1] DISCo, Università degli Studi di Milano - Bicocca, Viale Sarca 336, Ed. U14, 20126 Milan, Italy

[2] Informatikum, Universität Hamburg, Vogt-Kölln-Straße 30, 22527 Hamburg, Germany

Keywords Word sense induction · Graph clustering · Pseudowords · Evaluation

1 Introduction: graph-based word sense induction and pseudowords

Word Sense Induction (WSI) is the branch of Natural Language Processing (NLP) concerned with the unsupervised detection of all possible senses that a term can assume in a text document. It could also be described as "unsupervised Word Sense Disambiguation" (Navigli 2009). Since ambiguity and arbitrariness are constantly present in natural languages, WSI can help improve the analysis and understanding of text or speech (Martin and Jurafsky 2000). At its core we find the notion of distributional semantics, exemplified by the statement by Harris (1954): "*Difference of meaning correlates with difference of distribution.*"

In this article, we focus on graph-based WSI methods. Graphs provide an intuitive mathematical representation of the relations between words (Biemann and Quasthoff 2009). A graph can be defined and built in a straightforward way, but allows for a very deep analysis of its structural properties. This and the discrete nature of a graph, contrary to the continuous generalisations represented by vector spaces of semantics (Turney and Pantel 2010), favour the identification of significative patterns and subregions, among other things allowing the final number of clusters to be left unpredetermined, an ideal condition for WSI.

The main contribution of this article is threefold:

1. we present two parallel pseudoword ego word graph[1] data sets, based on *semantic similarities* and *co-occurrences*;
2. on them, using these data sets, we compare the performances of six WSI clustering algorithms;
3. we define a new ad hoc evaluation measure for this task, called TOP2.

In the process, we highlight the properties of pseudoword ego graphs with respect to their components, the different structures of word graphs originating from semantic similarities and co-occurrences, and the biased behaviour of some existing and widely-used significance measures.

This article is structured as follows. Section 2 is a brief overview of previous works that deal with WSI and pseudowords. In Sect. 3 we give a definition and a motivation for pseudowords. Section 4 introduces the framework of pseudoword evaluation, focused on our task of homonymy detection. Section 5 details the specific components that we used to implement our pseudoword evaluation framework, including a brief description of the considered clustering algorithms. Lastly, Sect. 6 comments the results of the comparisons, and Sect. 7 concludes the article.

[1] An *ego word graph* of a word *w* is a graph that represents the context of *w* in the corpus; alternatively, it can be seen as the neighbourhood of *w* in a word graph that globally represents the corpus. See Sect. 5.1.1 for the definition of ego word graph in our framework.

2 Related works

Pseudowords were first proposed in Gale et al. (1992) and Schütze (1992) as a way to create artificial ambiguous words by merging two (or more) random words. We will discuss them more in depth in Sect. 3. We note that ours is not the first approach making use of pseudowords: we are aware of the works Otrusina and Smrž (2010) and Pilehvar and Navigli (2013, 2014) (which find an application e.g., in Başkaya and Jurgens 2016) on a pseudoword generation method that better models polysemous words with an arbitrary degree of polysemy. The aim of these works is to use pseudowords molded into the "form" of existing polysemous words, following the semantic structure of WordNet.[2] Given a word with more than one *synset* in WordNet, for each such *synset* the goal is to find a monosemous term (i.e., with only one *synset*) that is as close as possible to that *synset* according to some distance on the WordNet graph. This term will then be selected as one of the *pseudosenses* of the pseudoword. However, we want to remark on the different nature of our article. Here we compare the behaviours of different clustering algorithms on two data sets of pseudowords built to emulate homonymy, and relate these behaviours to the structure of the pseudowords' ego word graphs. As homonymy is more clear-cut than generic polysemy, we deem that the efficacy of a WSI algorithm should be first measured in this case before being tested in a more fine-grained and ambiguous situation. Homonymy recognition itself is relevant for many tasks like lexical substitution, where the ability to distinguish completely different senses and uses is more important than the subtle distinction between facets of the same meaning. Also, the task we define does not depend on the arbitrary granularity of an external lexical resource, which might be too fine-grained for our purpose. Further, the sense distinctions e.g., in WordNet might not be mirrored in the corpus, and conversely, some unforeseen senses might be observed.

We also mention (Marco and Navigli 2013), where pseudowords are used as part of a complex evaluation framework investigating the quality of web search results. There, different WSI algorithms (among them Chinese Whispers, considered also in the present work; see Sect. 5.1.3) are tuned on a small pseudoword data set before being tested with regard to the organisation and diversification of results of web queries. This way, the algorithms are quite indirectly compared in terms of quality of their clusterings. Collaterally, the authors also sketch out an "*in vitro*" evaluation of the clusterings relying on the rankings obtained from human annotators. However, the fact that pseudowords are not central to that setting, and that the performances of graph-based WSI algorithms are principally interpreted in the light of desirable properties of web search results (also noticing that the graph-building step varies for each algorithm and relies on a corpus different than the retrieved web text snippets), let us assert that in the end the cited work has a very different nature than our article, even if some techniques are similar. In particular, the authors want to cluster word instances and do not induce a representation of word senses: this becomes more apparent when new data are introduced, as instances needs to be re-clustered every time this happens.

[2] http://wordnetweb.princeton.edu/perl/webwn, Miller (1995).

Another work, more closely related to ours, where pseudowords play a major role as a means to evaluate a WSI algorithm is proposed by Bordag (2006), where a triplet-based approach is presented. While the methodology and the data sets of our work may surely be seen as an expansion of Bordag's paper, we can affirm that the pseudoword evaluation framework presented in this article, while describing a generic method for algorithm evaluation, is more focused on the task of graph-based homonymy detection rather than on the generic assessment of the validity of an algorithm. Moreover, the algorithms are not evaluated against themselves (i.e., the sense-identifying clusters yielded by an algorithm for two words v and w are compared to those yielded by the same algorithm for the pseudoword v_w), but instead compared to the objective ground truth underlying each pseudoword. In our setting, this is achieved by defining more precise restrictions on the pseudoword components. Further, we also consider semantic similarities alongside mere co-occurrences (see again Sect. 5.1.1) to build pseudoword ego graphs.

Finally, in our opinion current WSI tasks have some shortcomings. A fundamental problem is the vagueness regarding the granularity (fine or coarse) of the senses that have to be determined. As a consequence, the definition of an adequate evaluation measure becomes difficult, as many of them have been showed to be biased towards few or many clusters.[3] Further, small data sets often do not allow obtaining significant results. Pseudoword evaluation, on the contrary, presents an objective and self-contained framework where the classification task is well characterised and gives the opportunity to define an ad hoc evaluation measure, at the same time automating the data set creation. Therefore, we tackle the following research questions: what are the limitations of a graph-based pseudoword evaluation for homonymy detection? How does the structure of a pseudoword ego word graph depend on its components? How do different clustering strategies compare on the same data set, and what are the most suited measures to evaluate their performances?

3 Pseudowords

A *pseudoword* is an artificial lexical construct used to help in the creation of data sets for the evaluation of Word Sense Induction clustering algorithms.

Pseudowords were independently first proposed in Gale et al. (1992) and Schütze (1992). The underlying idea is to treat usually two, but sometimes three or more, different words in a given data set as one and the same word: we treat the occurrences of the first word as if they were also occurrences of the other word. From the point of view of Word Sense Induction, for which we are interested in modelling a word's context, this is equivalent to joining the contexts of both words. We are effectively creating a new, non existent word from the conflation of two distinct terms, a *pseudoword* which potentially carries all the senses of its original

[3] See for example the results at task 14 of SemEval 2010 (Manandhar et al. 2010), where adjusted mutual information was introduced to correct the bias: https://www.cs.york.ac.uk/semeval2010_WSI/task_14_ranking.html.

components; these are also called *pseudosenses*. A pseudoword is thus an artificial lexical object that is the mere sum of its parts. We can take *banana_door* as an example: this pseudoword encompasses all and only the meanings of *banana* and *door*. So, if our data set included the sentences

- This afternoon I ate a *banana*.
- Remember to close the *door*!

we would treat *banana* and *door* as instances of the same fictive word *banana_door*. In our notation, the pseudoword originating from the combination of words A and B will be written as A_B, with an underscore joining the two components, which represents the equal importance of both components. We do not resort to a hyphen, since this might incorrectly lead to interpret a pseudoword (a mere sum of its components) as a compound word (a new word with its own meaning).

From the above discussion, if we denote a generic context[4] of a word v as $C(v)$, a set whose elements are words, we will identify the context of v_w with

$$C(v\_w) = C(v) \cup C(w).$$

In other words, the context of a pseudoword simply consists in the union of the respective contexts of its components.

Going back to the previous example, we could extract the following contexts:

$$\begin{aligned} C(\text{banana}) &= \{afternoon, eat\} \\ C(\text{door}) &= \{close, remember\} \\ C(\text{banana\_door}) &= \{afternoon, close, eat, remember\}. \end{aligned}$$

As mentioned in Sect. 1, from the perspective of distributional semantics word senses are determined by a word's context. It follows immediately that the senses of a pseudoword are themselves determined by the union of the senses of its components (its pseudosenses), and that consequently a pseudoword can roughly simulate ambiguity. Pseudowords bring two main advantages to WSI:

1. they allow us to greatly expand data sets of ambiguous words using only few building blocks and no human intervention;
2. knowing the sense distribution of the components translates into knowing the pseudosense distribution of a pseudoword.

With regard to the first point, we observe that if we have n words with known distributions, there are $\binom{n}{2}$ pseudowords that can be generated from them, for a nearly quadratic increase of available terms.

[4] In this example a context is informally understood as the lemmatised versions of content words co-occurring with the target word. A formal definition of the kind of context used in our work will be given in Sect. 5.1.1.

The second point is crucial for the evaluation frameworks that have been proposed in the past and for the one that we want to propose in this article: it means that, knowing the sense distribution of words v and w, we already know the ideal clustering of the context of v_w and that we can use this information to gauge the performance of a WSI algorithm on v_w. Despite the potentialities we have shown, there are also issues that we need to confront before better defining the implementation of pseudoword evaluation in Sect. 4.

We recall that proper homonymy arises when two semantically and etymologically unrelated senses are expressed by the same spoken or written sign, i.e., word, such as *count* in the sense of a *nobleman*, as opposed to *the act of enumerating* (Lyons 1968). This word is ambiguous because we can not assign it a meaning without first having some clarifying context. At the opposite side of the ambiguity spectrum lie polysemous words, whose different senses are more or less strictly related to each other and often revolve around a common concept. Combining random words will generate pseudowords that fall in between these two extremes. If v and w only possess unrelated senses, v_w will be equivalent to a case of proper homonymy. Sometimes, however, some kind of overlap between the semantic spheres of v and w is likely to exist, up to the point that the senses of v_w are ultimately blurred and not easily distinguishable anymore. For example, using two near-synonyms like *door* and *gate* will produce a pseudoword which is substantially equivalent to any one of its two components. On another note, *door* and *window* will surely share an important amount of contexts, as they both express a metaphorical or architectural concept of *opening* in a building, and the resulting pseudoword will probably represent this more abstract notion. Therefore, even if the previous statement about the knowledge of a pseudoword's sense distribution still remains true, we notice that haphazardly mixing words will possibly yield rather unpredictable and unwanted results, in the sense that it is not clear how to define *a priori* the type of ambiguity that a generic pseudoword simulates. Further, if we perform the process of pseudoword creation using three or more component words, this issue will naturally sharpen.

The study by Nakov and Hearst (2003) shows that the performances of Word Sense Disambiguation (WSD) algorithms used to disambiguate the senses of totally random pseudowords might act as an "optimistic" upper bound with respect to the same performances on true polysemous words: while pseudowords might not be a case of perfect homonymy, their senses are still easier to separate in most cases, as they lack the usual kind of correlation between sense categories and sense distributions of a real polysemous term. Introducing some sorts of restrictions, like requiring a minimum frequency for each component or taking components with similar distributions, we can obtain a pseudoword closer to the real case, and in fact a drop in the disambiguation performances is noted. From a set-theoretic point of view, the quality of a pseudoword greatly depends on the cardinality of the intersection $C(v) \cap C(w)$. The smaller its cardinality, the better v_w will approximate a case of homonymy. This point will be investigated in Sects. 4.1 and 4.1.1, and further in Sect. 5.2. Essentially, we can claim that on the scale between clear-cut homonymy and fuzzy polysemy random pseudowords lie closer to the former; on the other hand, the more words are combined together, the more we approach the latter.

4 Pseudoword evaluation framework

In Sect. 3 we point out how the kind of ambiguity modelled by a pseudoword v_w is not always clearly predictable. To keep the complexity of such lexical interactions as limited as possible, we want to restrict ourselves to the simplest and most unambiguous case of a pseudoword, namely: the **combination of two monosemous terms**. This way, we want to simulate a *disemous* (i.e., with two senses) case of homonymy.

Our motivation is that homonymy is usually more clear-cut than generic polysemy: the senses will be more distinguishable in terms of contexts, and their respective distributions will be independent from each other with good approximation. On the contrary, the senses of a polysemous words tend to have more intricate relationships and the line between one and the other is often hazy; WordNet's fine-grainedness is an example of how some subtle distinctions between different meanings may appear arbitrary at times. For this reason, as already stated, we deem that the efficacy of a WSI algorithm should first be measured in the case of a well-defined homonymy before being tested in a more fine-grained and vague circumstance. Also, we do not want our evaluation to depend on the subjective granularity of an external lexical resource, like in Pilehvar and Navigli (2014).

Considering only monosemous components for our pseudowords allows us to perform the following analysis of the structure of a pseudoword.

4.1 Structure of the ego graph of a disemous pseudoword

Since we want to evaluate graph-based clustering algorithms for WSI, we will associate a weighted word graph to each generic word z in our corpus and each pseudoword v_w. Such a graph is called an *ego graph*, and we write it respectively as G_z and G_{vw}. This kind of graph represents the context, i.e., a portion of the text, found in the corpus in relation to that specific pseudoword or word: each node represents a word and edges the relationships between them. More details on what kinds of ego graphs we use in our framework and how we construct them are given in Sect. 5.1.1. Here, we want to focus on the particular structure of a pseudoword's ego graph G_{vw} with respect to the components' ego graphs G_v and G_w.

As discussed in Sect. 3, the knowledge about v and w translates into being able to tell if a node u of G_{vw} belongs to either ego graph G_v or G_w, or if it can be found in both. This means respectively that u is more strongly tied to the contexts of either v or w, or instead that it acts neutrally with respect to both.

More formally, we denote the node sets of G_v, G_w and G_{vw} respectively as V_v, V_w and V_{vw}, and we write

$$V'_v = V_v \cap V_{vw}$$
$$V'_w = V_w \cap V_{vw}.$$

The sets V'_v and V'_w thus represent all the terms appearing in the graphs G_v and G_w (i.e., the terms that we associate respectively to v and w) that also appear in G_{vw}. We

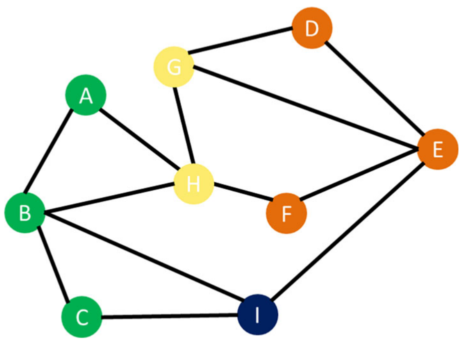


Fig. 1 The partition of a pseudoword's ego graph, with $\alpha = \{A, B, C\}$, $\beta = \{D, E, F\}$, $\gamma = \{G, H\}$, $\delta = \{I\}$. Therefore, the ground truth clustering is $\mathcal{T} = \{\{A, B, C\}, \{D, E, F\}\}$. If we had another clustering $\mathcal{C} = \{\{A, B, D, G, H\}, \{C, E, F, I\}\}$, for evaluation purposes we would just consider $\mathcal{C}' = \{\{A, B, D\}, \{C, E, F\}\}$

use this notation because we observe that not every element of V_v and V_w will necessarily appear in G_{vw}: when "combining" the two contexts, terms which are less strongly related to either component will be left out in favor of other ones more significant to the pseudoword (cf. Sect. 5.1.1).

Now we can express the pseudoword's ego graph node set V_{vw} as

$$V_{vw} = \alpha \cup \beta \cup \gamma \cup \delta, \tag{1}$$

with

$$\begin{aligned} \alpha &= V'_v \setminus V'_w \\ \beta &= V'_w \setminus V'_v \\ \gamma &= V'_v \cap V'_w \\ \delta &= V_{vw} \setminus (V'_v \cup V'_w). \end{aligned} \tag{2}$$

Elements in α are nodes in V_{vw} that are specific only to v, and the same goes for elements in β with respect to w. The subset γ gathers elements associated to both components v and w and gives a measure of how much the contexts of the two words overlap in our corpus. The last set δ contains all terms that appear only in the combined context of v and w that underlies the pseudoword. It is indeed possible that such words are not significant enough (cf. Sect. 5.1.1) to appear in G_v or G_w, but merging their contexts lets them gain enough relevance to appear in G_{vw}.

For the purpose of evaluation, we can not take elements of δ into consideration, as in our model they are not relatable to either v or w: they are absent from the word graphs that model their respective contexts. The same can be said for nodes in γ, which act as neutral, non discriminating elements and can not be taken to influence the evaluation process: we do not want to penalise an algorithm for associating elements in γ with elements in α rather than in β, as both choices are equivalent.

Therefore, we will identify just α and β as constituting the desired underlying partition[5] of V_{vw}. We define the *ground truth clustering* of V_{vw} as

$$\mathcal{T} = \{\alpha, \beta\}. \tag{3}$$

We want to compare the clustering $\mathcal{C}$ obtained by any given clustering algorithm on G_{vw} to the ground truth clustering $\mathcal{T}$; to this end, we have to consider a reduced version of $\mathcal{C}$, more precisely

$$\mathcal{C}' = \{C \backslash \{\gamma \cup \delta\} \mid C \in \mathcal{C}\}, \tag{4}$$

i.e., what remains of the clustering $\mathcal{C}$ if all the neutral and unrelated elements are ignored. Figure 1 illustrates this situation, while Sect. 5.2.1 shows a practical application of the analysis conducted here with regard to the structure of a pseudoword ego graph.

4.1.1 Collapsed pseudowords

Looking back at partition (1) of the node set V_{vw} of G_{vw}, we observe that it is entirely possible that either relation

$$\alpha = \emptyset \quad \text{or} \quad \beta = \emptyset$$

holds. This means equivalently that either relation

$$V'_v \subseteq V'_w \quad \text{or} \quad V'_w \subseteq V'_v$$

holds. We will observe this case when the terms in the selected context of pseudoword *v_w* which are associated to the component *v* are all also already associated to the other component *w*: one component totally overshadows the other one. If we assume the case $\alpha = \emptyset$, then *w* will be the dominant component word, and G_{vw} will mostly or totally collapse onto a near-copy of G_w. Therefore, G_{vw} loses its interest as a potential disemous word and no real Word Sense Induction can be performed on it.

Definition 1 We define a *collapsed pseudoword* as a pseudoword *v_w* for which, in the notation of (1), either $\alpha = \emptyset$ or $\beta = \emptyset$ holds.

We decide to ignore all collapsed pseudowords for evaluation purposes. We note that this phenomenon is linked to the difference in frequency between the components of a pseudoword (cf. Sect. 5.1.2 for our definition of frequency class). In fact, we observe that a pseudoword never collapses if its components have a comparable frequency in the corpus, and conversely, that the majority of the more uneven combinations, with a larger gap between the respective frequencies, tend to collapse; further, some words are seemingly more dominant than others. Still, many non-collapsed pseudowords remain very skewed towards one component (cf. Sect.

[5] If $\gamma \neq \emptyset$ and/or $\delta \neq \emptyset$, we are actually considering a *non-exhaustive* partition or a *subpartition*, i.e., a collection of disjoint, non-empty subsets whose union is not necessarily the whole set.

5.2). Despite pseudowords being often criticised for their supposed artificialness and for not obeying the true sense distribution of a proper polysemic word, we note that a similar skewness is very realistic in the case of homonymy, where sense distributions tend to be dominated by a most frequent sense (MFS). In coarse-grained Word Sense Disambiguation evaluations, the MFS baseline is often in the range of 70–80% (Navigli et al. 2007), not unlike for the pseudowords in our data sets. An example of collapsed pseudoword in our data set is presented in Sect. 5.2.1.

4.2 Pseudoword evaluation goal

We now state the goal of pseudoword-based evaluation formally:

Pseudoword evaluation goal Given:

- a corpus,
- two different words v and w in the corpus and their ego graphs G_v and G_w,
- the pseudoword v_w resulting from their combination and its corresponding ego graph G_{vw} (cf. Sect. 5.1.1),
- a graph-based clustering algorithm $\mathfrak{A}$,
- a measure μ to gauge the significance of clustering overlap,

we want to assess how well the clustering $\mathfrak{A}(G_{vw}) = \mathcal{C}$ coincides with the desired ground truth clustering $\mathcal{T} = \{\alpha, \beta\}$, defined on the basis of relation (1), by means of the score

$$\mu(\mathcal{C}', \mathcal{T}),$$

with $\mathcal{C}'$ defined by (4).

We will perform this pseudoword evaluation for different clustering algorithms and significance measures, on two different, parallel ego graph data sets. The choice of the evaluation metric μ is very delicate and can introduce a certain bias, as discussed in Sect. 6.1.2. Motivated by this fact, in Sect. 5.1.5 we will define a new, ad hoc measure for this evaluation setting, called TOP2.

5 Evaluation implementation, techniques and data set analysis

In this section we will present the elements that compose our particular implementation of the pseudoword evaluation framework. We will also describe the new techniques that we introduce and perform a brief analysis of our two pseudoword ego graph data sets, one for co-occurrences and the other one for semantic similarities.

5.1 Framework details

5.1.1 Corpus, distributional thesauri, LMI *and word graphs*

Our corpus is a combination of the Leipzig Corpora Collection[6] (LCC) (Richter et al. 2006) and the Gigaword corpus (Parker et al. 2011), and consists of 105 million English newspaper sentences. The corpus has been parsed using the Stanford Parser (De Marneffe et al. 20016) to extract syntactic dependencies. Following the approach found in *JoBimText*[7] (Biemann and Riedl 2013), we use a term-context frequency-weighted version of pointwise mutual information called *lexicographer's mutual information* (or *local mutual information*—LMI) (Evert 2004; Kilgarriff et al. 2004) to compute similarity scores between any two co-occurring words or between a word and a syntactic context in which it appears.

LMI is a variant of *pointwise mutual information* (PMI) (Cover and Thomas 2012), which is expressed between two random variables X and Y in terms of their respective individual and joint probability distributions P_X, P_Y and P_{XY} as

$$\log \frac{P_{XY}(x,y)}{P_X(x)P_Y(y)}, \tag{5}$$

for their respective observations x and y. PMI measures how significant the co-occurrent observation of two values assumed by X and Y is: the higher, the more meaningful their co-occurrence is, whereas a value of 0 represents independence. However, in the case where our observations are e.g., two words v and w in a text, the formula (5) has been shown to overestimate the significance of co-occurrences of infrequent terms. To counterbalance this phenomenon, LMI is defined as the product of PMI with the frequency of the observed pair (v, w). If we denote their individual frequencies by c_v and c_w, the frequency of their co-occurrence by c_{vw} and the total number of words in the text as n, we will write their lexicographer's mutual information as

$$\mathrm{LMI}(v,w) = c_{vw} \log\left(N \frac{c_{vw}}{c_v c_w}\right). \tag{6}$$

This way, it will be given much more importance to word pairs that also appear a significant number of times in the text than to random infrequent co-occurrences [see Riedl (2016) on this issue].

The goal is to create both a first-order and a second-order *distributional thesaurus* for each word w in the corpus, i.e., a ranking of words from the most to the least similar to w according to the LMI computed respectively for co-occurrences or for syntactic contexts. A word in a second-order distributional thesaurus of w does not necessarily co-occur with w in a sentence of the corpus. We notice that, for each word, we take its lemmatised version and that we distinguish the difference between lowercase and uppercase letters as found in the corpus.

6 http://corpora.uni-leipzig.de.

7 https://sourceforge.net/p/jobimtext/wiki/Home/.

For a given term w, its *first-order distributional thesaurus* will use LMI to rank every word that co-occurs with w in a **sentence**. Based on these rankings, we choose a number n of significantly (syntagmatically) similar words to consider for each resulting distributional thesaurus. Then, for every word w, we build the graph $G_w = (V, E)$ with the n highest ranked entries in its thesaurus as its node set v, so that $|V| = N$. An edge will be drawn between $u, v \in V$ if u is part of the first n entries of v's thesaurus or, viceversa, v appears in the first n entries for u. We notice that both conditions will not necessarily be satisfied at the same time, but, if this happens, since LMI is symmetric the two scores will be the same. Knowing this, we define a weight mapping on G_w by setting the weight of (u, v) equal to the LMI co-occurrence score of the couple. We call the graph G_w the (co-occurrence-based) weighted *ego graph* of w. We remove w from G_w, as, following the same reasons found in Widdows and Dorow (2002), we want to focus on the relations between terms co-occurring with w, and not on the relations between them and w. We might use some extra care to exclude stop words, which are already penalised by the PMI part of LMI, but which nonetheless keep appearing in the distributional thesaurus because of their high frequency.

The passage to the second order takes place by using LMI on the first-order relations given by the extracted syntactic dependencies. The resulting *second-order distributional thesaurus* induces a semantic similarity ego word graph that is shaped analogously as the previously detailed co-occurrence-based ego graph.

Whichever its form, an ego graph G_w based on co-occurrences or semantic similarities can be clustered to induce the word senses of w in form of sense clusters. We consider a word w to be *monosemous* with respect to a given clustering algorithm if that clustering of G_w yields a single cluster. We might call this a "bottom-up approach", as it relies just on instruments already present in our evaluation setting and does not introduce new ones from outside. In particular, we use a fine-grained version[8] of Chinese Whispers (described in Sect. 5.1.3). Additionally, we check that w exists in WordNet and has only a single synset there. This double criterion is quite strict and, while an external look-up is not indispensable to our setting, it is helpful for avoiding the bias coming from algorithms which favour a coarse clustering (cf. Sect. 6.1.2).

Lastly, we notice that word graphs have been shown to be of the small-world type (Watts and Strogatz 1998; Ferrer i Cancho and Solé 2001) and often also scale-free (Barabási and Albert 1999). This lends them a peculiar structure that can be exploited for WSI purposes, but that might also create unwanted behaviours by part of a clustering algorithm. However, in this article we will not further analyse this issue.

5.1.2 Pseudoword components

To create our pseudoword ego graph data sets, we have chosen to focus only on **nouns** for the pseudosenses. This choice is dictated both by practical reasons, such

[8] Precisely, the implementation found in https://sourceforge.net/p/jobimtext/wiki/Sense_Clustering/ with parameters: `-n 200 -N 200`.

as the fact that nouns are easier to lemmatise than e.g., verbs, and by the more theoretical observation that nouns are open-class words[9] that are not subordinated to other word classes (as is e.g., for adjectives or adverbs) and with a more immediate reference to real or abstract entities than verbs. Also, keeping the focus on just one word class reduces the noise deriving from syntagmatic relations between different classes: if we are simulating homonymy for nouns, we only want to mix nouns as pseudosenses. For similar reasons, we will restrict the elements of semantic-similarity-based ego graphs to nouns only (as the comparison between two nouns is more appropriate than the one e.g., between a noun and a verb), whereas this restriction is absent for co-occurrences.

We divide all the nouns in the corpus into 5 logarithmic frequency classes, based on the frequency of the most common one, i.e., *year*, which appears 6,179,666 times. We take the binary logarithm of this value, approximately 22.6, and assess the frequency class (FC) of a word according to where its logarithmicised frequency falls with respect to its quintiles.[10] For example, if a word w appears 10,000 times in the corpus, the binary logarithm is approximately 13.3 and w will belong to frequency class 3. Conversely, a member of frequency class 3 will have a frequency between ca 526 and 12,077. In general, the member of one of our frequency classes is expected to be 23 times as frequent in the corpus as a member of the next-lower class.

For each frequency class, we extract random candidates and retain only 10 of them which are found to be monosemous according to the criteria presented in Sect. 5.1.1. The number of their combinations, and thus of total different pseudowords, amounts to $\binom{50}{2} = 1225$. We then proceed to compute all their respective first-order and second-order ego graphs. For pseudocomponents v and w, we substitute all their occurrences in the corpus with v_w and go normally through the steps described in Sect. 5.1.1 to get a semantic-similarity and a co-occurrence-based ego graph G_{vw}. We remark that this graph-building process has to be performed separately for each pseudoword, as we want to merge only two contexts at a time. That is, we notice that if a word z co-occurs with v or w, the pseudoword v_w does not have any sensible interpretation as an entry of the distributional thesaurus of another possible pseudoword u_z. We limit the size of all the distributional thesauri, and consequently of the ego graphs, to $N = 500$ nodes, be they of regular words or pseudowords. For very infrequent terms, it is possible for G_{vw} to have a smaller order.

We end up with 50 monosemous words evenly distributed among 5 frequency classes that will serve as pseudosenses:

Frequency class 1:	*ackee, barque, bobolink, bufflehead, carryall, euonymus, leatherette, pennywhistle, pneumococcus, tautog*
Frequency class 2:	*afro, barman, catsup, dimwit, grosgrain, hyperthyroidism, philanderer, southerly, tranquilliser, yellowcake*

[9] On this topic cf. Lyons (1968).

[10] The quintiles are the four values that divide a quantity in five parts: in this case, they are the multiples of ca 4.52, i.e., 4.52, 9.04, 13.56 and 18.08.

Frequency class 3:	*astrologer, bullfight, curio, flamboyance, hairpiece, lightbulb, showtime, tailwind, ugandan, wedlock*
Frequency class 4:	*artistry, heartache, heartbreak, marshmallow, pillow, reliability, reptile, shawl, tequila, tofu*
Frequency class 5:	*ailment, beer, clothing, courage, credibility, dioxide, garment, paycheck, psychologist, treasurer*

Of the ensuing 1225 pseudowords, for evaluation purposes we have to discard a total of 249 collapsed pseudowords (Sect. 4.1.1) from the semantic-similarity-based ego graph data set and 143 from the co-occurrence-based one, mostly consisting of combinations of terms of FC 1 with terms of FC 5 (the greatest possible difference in frequency classes); some terms in FC 5 (like *beer* or *courage*) are particulary dominant. This brings the actual count of pseudowords used in our evaluation down to respectively 976 and 1082 pseudowords.

5.1.3 Clustering algorithms

We will evaluate and compare the performances of six clustering algorithms used in the field of WSI: three selected from literature, the Markov cluster algorithm, Chinese Whispers, MaxMax, and three defined by us in other works, the gangplank clustering algorithm, a form of aggregative clustering algorithm and a new curvature-based clustering algorithm. We give a short description of each in the following paragraphs. Additionally, we will briefly touch upon the well-known WSI algorithm HyperLex (Véronis 2004), which we also subject to our evaluation framework, albeit their interaction is problematic (as also discussed in Sect. 6.1.1).

Markov Cluster Algorithm The *Markov cluster algorithm* (MCL) (van Dongen 2000) uses the concept of random walk on a graph, or Markov chain: the more densely intra-connected a region in the graph, the higher the probability to remain inside it starting from one of its nodes and moving randomly to another one. The strategy of the algorithm is then to perform a given number n of steps of the random walk, equivalent to taking the n-th power of the corresponding Markov random field's adjacency matrix. Subsequently, entries of the matrix are raised to a given power to further increase strong connections and weaken less significant ones. This cycle is repeated an arbitrary number of times, and, as weaker connections tend to disappear, the resulting matrix is interpretable as a graph clustering. Not rooted in the NLP community, MCL was used for the task of WSI on co-occurrence graphs in Widdows and Dorow (2002). We use two implementations of the algorithm: one with inflation factor 1.4 and another one with inflation factor 2. In both cases, the expansion factor is 2 and the maximum number of iterations is 100.

Chinese Whispers The *Chinese Whispers* (CW) algorithm is originally described in Biemann (2006). It is inspired by MCL as a simplified version of it and similarly simulates the flow of information in a graph. Initially, every node in the graph starts as a member of its own class; then, at each iteration every node assumes the prevalent class among those of its neighbours, measured by the weights on the edges incident to it. This algorithm is not deterministic and may not stabilise, as nodes are accessed in random order. However, it is extremely fast and quite successful at distinguishing denser subgraphs. The resulting clustering is generally relatively coarse. Besides its use for Word Sense Induction, in Biemann (2006) CW was also tested on the tasks of language separation and word class induction.

MaxMax (MM) is described in Hope and Keller (2013) and applied to the task of WSI on weighted word co-occurrence graphs. It is a soft-clustering algorithm that rewrites the word graph G as an unweighted, directed graph, where edges are oriented by the principle of *maximal affinity*: the node u dominates v if the weight of (u, v) is maximal among all edges departing from v. Clusters are then defined as all the maximal quasi-strongly connected subgraphs of G (Ruohonen 2013), each of which is represented by its root. Clusters can overlap because a node could be the descendant of two roots at the same time. The algorithm's complexity is linear in the number of the edges and its results are uniquely determined.

Gangplank clustering algorithm The *gangplank clustering algorithm* (GP) is introduced in Cecchini and Fersini (2015), where its use for the task of WSI on co-occurrence graphs is shown. There, the concept of *gangplank edges* is defined: these are edges that can be seen as weak links between nodes belonging to different, highly intra-connected subgraphs of a graph, and thus help induce a cluster partitioning of the node set. In its proposed implementation, the computation of gangplank edges and the subsequent clustering of G is actually performed on a second-order graph of G, a *distance graph* D_G which represents the distances between nodes of G according to a weighted version of Jaccard distance adapted to node neighbourhoods. Further, a recompacting step to avoid singleton clusters is also implemented.

Aggregative clustering algorithm An *aggregative clustering algorithm* (AGG) is proposed in Cecchini et al. (2015). It consists of a k-medoid algorithm (Lloyd 1982) with the addition of a medoid seeding step. The same weighted Jaccard distance used on word graphs for the gangplank clustering algorithm allows us to consider a word graph as a metric space: there, starting from an initial seed word, new cluster seeds are determined according to a radius parameter σ, which influences the sizes of the clusters and thus the coarseness of the clustering. Successively, the k-medoid algorithm is iterated until convergence. We note that the number of clusters is never pre-determined. Our implementation will use two types of radii: a fixed one of $\sigma = 0.97$, and a variable one set as the 75th percentile of the distance values in the word graph. Like for GP, a recompacting step is performed.

Curvature-based clustering algorithm The *curvature-based clustering algorithm* (CURV) also originates from the notion of Jaccard distance on a graph found in Cecchini and Fersini (2015); Cecchini et al. (2015) and is further detailed in Cecchini (2017). The observation is that the weighted and the unweighted Jaccard distances introduced there behave differently, and the curvature on a weighted, undirected graph is defined as the difference between the two. A positive difference is interpreted as implying a significative semantic similarity between two nodes. Clusters are obtained as the connected components of the graph induced by positive curvatures (seen as edges of a "curvature graph"). Like for GP and AGG, a recompacting step is performed.

HyperLex The clustering algorithm *HyperLex* (HL) is explicitly devised for WSI and first appears in Véronis (2004). Its implementation is split into two phases: the graph-building step and the actual clustering step. In the first step, a co-occurrence-based word graph for a target word is built out of a corpus of web pages containing it. After a paragraph-based text pre-processing, the weights, interpreted as distances, between node couples in the resulting graph G are computed according to the estimated likelihood of their co-occurrence in a paragraph. The algorithm then proceeds to identify roots, i.e., nodes of G with high degree and whose mean weight on the edges towards its neighbours does not exceed a given threshold σ (0.8 both in the original and in the present paper). This process roughly produces an independent dominating set of G.[11] These roots define sense clusters, taken as all their successors in the directed minimum spanning tree of G. HyperLex boasts a nearly linear time complexity with respect to nodes and edges. Despite its straightforward nature, we argue that its inclusion in our evaluation framework is unfortunately unsatisfactory, as discussed in Sect. 6.1.1.

Besides the six main selected clustering algorithms (all but HyperLex), we also introduce for all of them a secondary level of clustering that we call *hyperclustering*, detailed in the next section. We note here that HyperLex is not considered for hyperclustering, since it is normally meant for larger graphs than the hypergraphs we obtain on our data sets, on which its parameters do not interact well. We will discuss this issue more in detail in Sect. 6.1.1.

5.1.4 Hyperclustering

All possible clusterings of a set v are represented by its *partition lattice*, with the partial ordering given by partition refinement (Grätzer 2011). We denote this lattice with $\mathcal{C}(V)$ and notice that it behaves quite differently from the power set $\mathcal{P}(V)$ of all subsets of v. We can see $\mathcal{C}(V)$ as a hierarchy, with its supremum $\{V\}$ (only one, big cluster) at its top and the infimum $\{\{v_1\}, \{v_2\}, \ldots, \{v_{|V|}\}\}$ (each element forming its own cluster) at its bottom.

The aim of *hyperclustering* is to define a process by which the same algorithm that produces a clustering $\mathcal{C}$ can reuse it to build a sequence of coarser clusterings. Starting from $\mathcal{C}$, if $\leq$ represents the ordering by refinement in the partition lattice, we want to define an operator Hyp such that

$$\mathcal{C} \leq \mathrm{Hyp}(\mathcal{C}).$$

The iterated application of Hyp will move the clustering “upwards” in the hierarchy of $\mathcal{C}(V)$, up to a maximal element that may or may not be the supremum. We want Hyp to recompact $\mathcal{C}$, preserving its clusters as building blocks that are combined to form new, bigger clusters. This way, we can exploit the natural higher precision of smaller clusters to improve the overall recall of $\mathcal{C}$ without losing information about the already found associations. With what may sound like a pun, the operator Hyp “clusters the clusters of a clustering”.

To perform a clustering on $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ we first have to represent the clusters C_i as nodes of a graph and to define edges connecting them. We will call the

[11] On this graph-theoretical topic, see e.g., Haynes et al. (1998).

resulting graph the *hypergraph*[12] $H = (W, F)$ of $\mathcal{C}$. For the node set we simply define $W = \mathcal{C}$. The edges and their weights depend on the structure of the graph $G = (V, E)$ of which $\mathcal{C}$ is a clustering. We set

$$(C_i, C_j) \in F \quad \Leftrightarrow \quad \exists (v, w) \in E \quad \mathrm{s.t.v} \in \mathrm{C_i},\ \mathrm{w} \in \mathrm{C_j}.$$

In other words, two nodes of H are linked if there exists an edge in G linking the two corresponding subgraphs induced by the two clusters. This condition is by itself quite weak: a word graph over a large corpus will have many connections, so that H will be most probably very dense too. Therefore, we want to give a weighted structure to H. To do this, we consider all edges connecting C_i to C_j (we could see it as taking the bipartite subgraph of G induced by C_i and C_j) and take their mean weight. Formally, we define the interconnecting edge set

$$I(C_i, C_j) = \{(v, w) \in E \mid v \in C_i,\ w \in C_j\}, \tag{7}$$

and subsequently the weight mapping

$$\mu(C_i, C_j) = \frac{1}{|I(C_i, C_j)|} \sum_{(v,w) \in I(C_i, C_j)} p(v, w),$$

where $p : E \times E \longrightarrow \mathbb{R}^+$ is the weight mapping associated to G.

We express the clustering $\mathcal{C}$ of $G = (V, E)$ (more precisely of its node set v) obtained by algorithm $\mathfrak{A}$ equivalently as $\mathfrak{A}(G)$. We also denote the hypergraph relative to $\mathcal{C}$ as $H(\mathcal{C})$. Then, we define the hyperclustering of $\mathcal{C}$ as

$$\mathrm{Hyp}(\mathcal{C}) = \mathfrak{A}(H(\mathcal{C})). \tag{8}$$

The operator $\mathrm{Hyp}(\cdot) = \mathfrak{A}(H(\cdot))$ defines the hyperclustering $\mathrm{Hyp}(\mathcal{C}) = \{Q_1, \ldots, Q_m\}$, which directly induces a new clustering of G. Namely, the sets defined as

$$K_i = \bigcup_{C \in Q_i} C \quad \forall Q_i \in \mathrm{Hyp}(\mathcal{C})$$

form a new partition $\mathcal{K} = \{K_1, \ldots, K_m\}$ of v, which is by definition a coarsening of $\mathcal{C}$. To ease notation, we identify $\mathcal{K}$ with $\mathrm{Hyp}(\mathcal{C})$, and can therefore write $\mathcal{C} \leq \mathrm{Hyp}(\mathcal{C})$, as desired.

5.1.5 Evaluation metrics and TOP2

We will compare the output of a clustering algorithm to the ground truth clustering of a pseudoword ego graph (cf. Sect. 4.1) by means of three evaluation metrics. Two of them, normalised mutual information (NMI) (Cover and Thomas 2012; Strehl and Ghosh 2002) and BCubed F-measure (Bagga and Baldwin 1998), are commonly

[12] Despite some similarities, our definition of hypergraph is different than the common graph-theoretical concept that goes by the same name, namely that of a graph $G = (V, E)$ whose edges can be generic subsets of v. See Berge and Minieka (1973) for more details about the subject.

used in the field. The third one is TOP2, a new evaluation measure loosely inspired by NMI and developed specifically for the task at hand, taking into account the nature of our disemous pseudoword ego graphs. Basically, we exploit the fact that we know what the ground truth looks like and that it always consists of exactly two clusters. Our intention is to penalise clusterings that stray too far from this bipartition, rewarding however an algorithm which manages to concentrate the most significant information in two clusters, even if its clustering is otherwise rather dispersive. We define TOP2 as the average of the harmonic means of homogeneity and completeness of the two clusters that better represent the two original components of the pseudoword.

Let $\mathcal{C}$ denote a clustering of pseudoword v_w's ego graph. Recalling the relations (1) and (2) of Sect. 4.1, we define

$$C_v = \arg\max_{C \in \mathcal{C}} |C \cap \alpha| \quad \text{and} \quad \mathrm{C_w} = \arg\max_{C \in \mathcal{C}} |\mathrm{C} \cap \beta|, \tag{9}$$

the two clusters of $\mathcal{C}$ that have the greatest overlap with v's and w's distributional thesaurus respectively; C_v and C_w are the best approximations of α and β. In a sense, we are reducing the clustering $\mathcal{C}$ to its subset $\{C_v, C_w\}$. *A priori*, arg max defines a set of maximizing arguments; since we want to consider just one of them, if it has more than one we will choose randomly.

We define the *precision* or *purity* $p_v(C)$ of a cluster C with respect to component v, and analogously $p_w(C)$ with respect to w, as

$$p_v(C) = \frac{|C \cap \alpha|}{|C|} \quad \text{and} \quad p_w(C) = \frac{|C \cap \beta|}{|C|} \tag{10}$$

and the *recall* or *completeness* $c_v(C)$ or $c_w(C)$ as

$$c_v(C) = \frac{|C \cap \alpha|}{|\alpha|} \quad \text{and} \quad c_w(C) = \frac{|C \cap \beta|}{|\beta|}. \tag{11}$$

We notice that the definitions in (11) are always well defined, since we choose to discard collapsed pseudowords (see Definition 1) and therefore we will always have $|\alpha|, |\beta| \neq 0$. Additionally, by definition (9) and the general definition of clustering,[13] the values p and c in (10) for C_v and C_w will be always strictly greater than 0.

For both clusters C_v and C_w, we compute the harmonic mean of purity and completeness with respect to the component they are representing. We write the harmonic mean of two positive real numbers $x, y \in \mathbb{R}^+$ as $h(x, y)$ and remember that

$$h(x, y) = \frac{2}{\frac{1}{x} + \frac{1}{y}}.$$

So, given $h(p_v(C_v), c_v(C_v))$ and $h(p_w(C_w), c_w(C_w))$ we define the TOP2 score as their macroaverage

[13] We define the *clustering* of a set $\mathcal{S}$ as a finite collection of non-empty subsets of $\mathcal{S}$ whose union is the whole $\mathcal{S}$. In this paper, we often assume a clustering to also be a partition, i.e., that the subsets are all disjoint, but for some algorithms like MaxMax this is not always the case.

$$\textsc{top}2(C_v, C_w) = \frac{h(p_v(C_v), c_v(C_v)) + h(p_w(C_w), c_w(C_w))}{2}. \tag{12}$$

Notice that in (12) the first cluster is always valued with respect to v, and the second one with respect to w.

We observe that following the definition of representative clusters of (9) might lead to the unfortunate case $C_v = C_w$, where the algorithm concentrates and confuses all the information about the two distinct senses of the pseudoword into a single cluster, that we denote as C_{vw}. We want to penalise this behaviour. To this end we also consider the second best matches for (9), which we denote respectively as C'_v and C'_w. The final TOP2 score is then defined as the best one, i.e., yielding the greatest value, of the two cases $\{C'_v, C_{vw}\}$ or $\{C_{vw}, C'_w\}$.

The other case to take into account is when the algorithm produces only one cluster, i.e., when $\mathcal{C} = \{C\}$. As we can not resort to second best representative clusters, we look at the virtual clusterings $\{C, \emptyset\}$ and $\{\emptyset, C\}$ (the order makes a difference as with respect to which component purity and completeness are computed). Putting the harmonic mean of purity and completeness relative to the empty set equal to 0, we again take as the final TOP2 score the best one between the two considered clusterings; we notice that in this case the TOP2 score will never be greater than 0.5.

The TOP2 score can be immediately generalised to the case where we know that the ground truth will consist of any fixed number $n \geq 2$ of clusters, but we will not go into the details of it in this article.

5.2 Pseudoword analysis

In this section we will get more insight into the structure of the pseudoword ego graphs. We will not consider collapsed pseudowords, defined in Sect. 4.1.1. We distinguish two main parameters that characterise a pseudoword ego graph: *balance* and *relative overlap*.

The first aspect is the *balance*, or conversely the *skewness*, of a pseudoword. We represent it by means of the ratio

$$\rho = \max\left(\frac{|\beta|}{|\alpha|}, \frac{|\alpha|}{|\beta|}\right),$$

which is always greater than or equal to 1. In other words, we consider all the nodes that specifically belong to just one of the two components of v_w, i.e., the elements contained in the sets α or β (see Sect. 4.1), and measure the ratio of the most represented component to the least represented one (this ratio is well defined because we exclude the case of collapsed pseudowords of Definition 1). If a pseudoword's ego graph is perfectly balanced, $\frac{|\beta|}{|\alpha|} = \frac{|\alpha|}{|\beta|} = 1$. Otherwise, a value ρ will tell us that, for every word of the smaller component, G_{vw} contains ρ of the larger one. We take the mean of these ratios for each of the 15 different frequency class combinations, and show the results in Fig. 2: once for semantic similarities and once for co-occurrences, first ordered from less to more unbalanced frequency

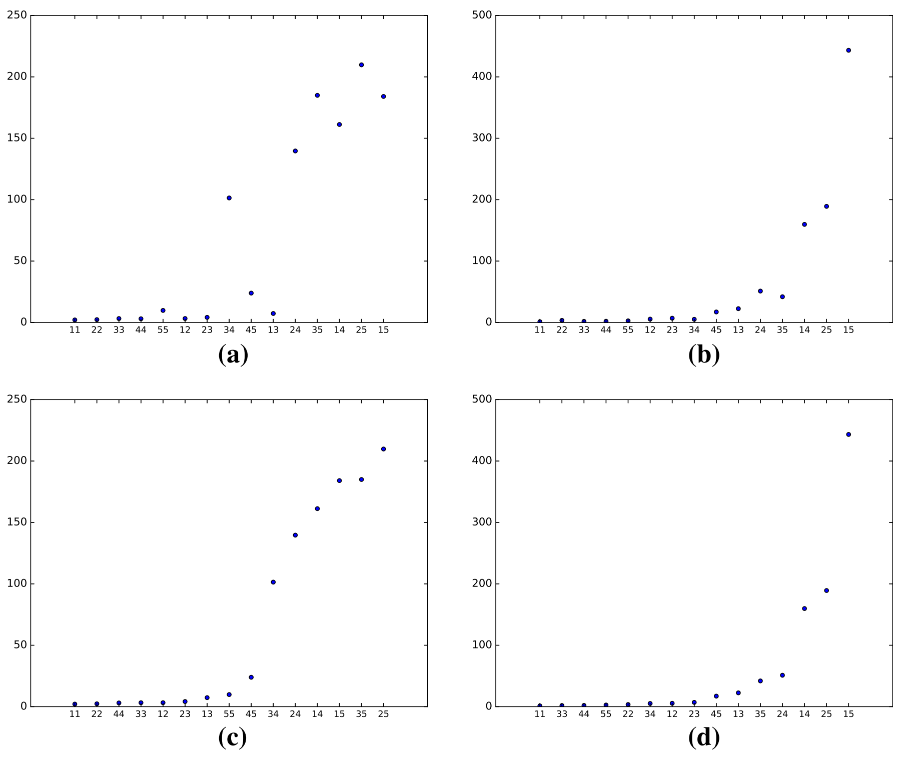


Fig. 2 The mean ratio ρ of the most represented component to the least represented component (either $|\alpha|/|\beta|$ or $|\beta|/|\alpha|$; y-axis) in pseudoword ego graphs based on semantic similarities and co-occurrences, by frequency class combination (x-axis) . **a** Ratio for semantic similarities, **b** ratio for co-occurrences, **c** ratio for semantic similarities, ordered from lowest to highest , **d** ratio for co-occurrences, ordered from lowest to highest.

class combination (e.g., 14 is more unbalanced than 22) and then reordered from smaller to greater values of the ratio.

We evidence the general, expected trend to have more unbalanced words when quite different frequency classes (FC) get mixed. Conversely, combinations of components in the same FC are mostly very balanced. This is the same behaviour that originates the prevalence of collapsed pseudowords (cf. Definition 1) among unbalanced FC combinations, which is actually the limit case where $\rho \rightarrow \infty$. The dominance of some terms in FC 5 makes 55 stick out as the more unbalanced of the most balanced combinations. We observe that more than half of our pseudowords are quite unbalanced towards one of their pseudosenses. Our starting assumption is that the greater ρ, the more biased towards the dominant term a clustering algorithm will be: in practice, we expect it to find just a single cluster most of the time (or equivalently a very big cluster together with other very small, unmeaningful clusters). In Sect. 6.1.2 we will see that BCubed seems to be insensible to such skewness.

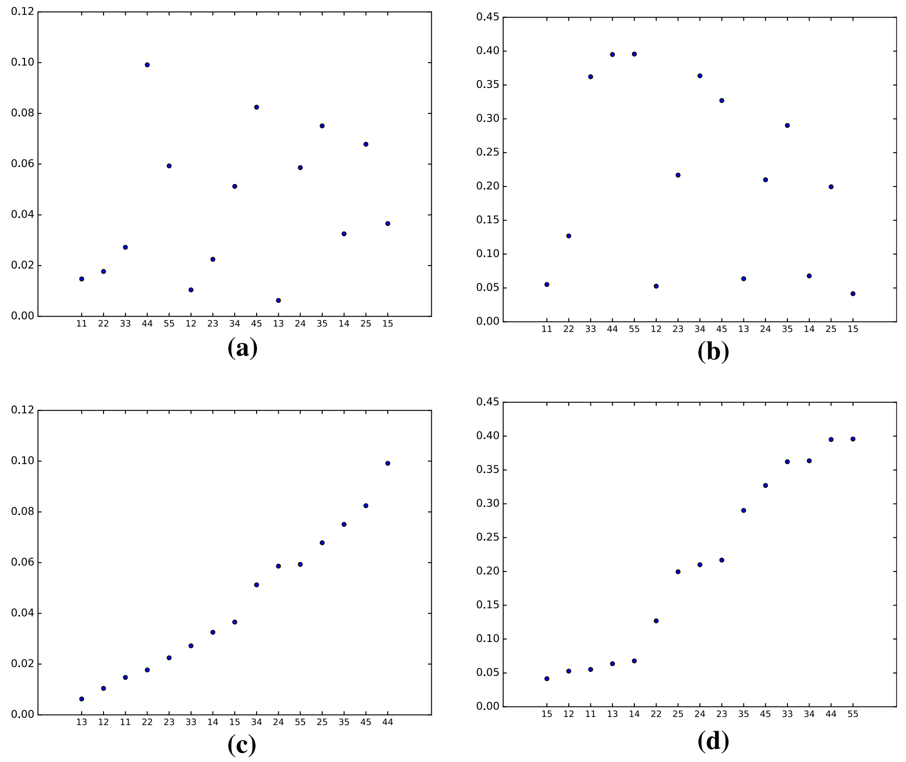


Fig. 3 Mean relative overlap κ given by the number $|\gamma|$ of nodes common to both components with respect to the total number of nodes (y-axis), by frequency class combination (x-axis) . **a** Relative size of γ for semantic similarities, **b** relative size of γ for co-occurrences, **c** relative size of γ for semantic similarities, ordered from lowest to highest, **d** Relative size of γ for co-occurrences, ordered from lowest to highest.

We note that co-occurrence-based ego graphs seem to be generally slightly more balanced than the equivalent similarity-based ones. We explain this fact by remarking that co-occurrences are sparser than syntactic relationships, and thus that LMI scores (cf. Sect. 5.1.1) tend to more evenly distributed, and components have a lesser tendency to predominate one over the other.

Figure 3 shows the mean cardinality of the node set γ per frequency class combination and relative to the total size of the ego graph. More precisely, given a graph $G = (V, E)$, we consider the ratio

$$\kappa = \frac{|\gamma|}{|V|}.$$

The set γ, in the notation of (1) (see Sect. 4.1), consists of the terms in G_{vw} that are common to the distributional thesauri of v and w and its size measures the *overlap* of v and w. The more similar the meanings of the two components, the larger we expect the overlap to be. In the case of the pseudoword *beer_tequila*, for example, 305 nodes, more than half of the total 492 nodes in its semantic-similarity-based ego graph, are found in the distributional thesauri of both *beer* and *tequila*, for a κ score of 0.62; the same number rises to 332 on 484 for co-occurrences (and the pseudoword collapses, so that the resulting score $\kappa = 0.69$ is not considered for the statistics shown). In Fig. 3a, b a pattern is clearly visible: the higher the mean FC of a combination and the greater its smaller member, the more terms the two components will have in common. Figure 3c, d show a roughly linear increase.

The generic trend of higher-frequency combinations to have a relatively bigger $|\gamma|$ is also directly related to the frequency of the single components. With a simplistic reasoning, the more frequent a term, the richer its distributional thesaurus, and the richer two distributional thesauri, the more probable that their elements will overlap at a higher degree. Very unbalanced pseudowords are likely to have relatively smaller $|\gamma|$'s: the unrelatedness of the two components is the cause both of the prevalence of the most frequent one and of a negligible overlap.

In the end, the analysis conducted here reveals that both our pseudoword data sets represent a good variety of artificial words that simulate real homonymy, with many different mixtures of balance and relative overlap values.

5.2.1 Examples of parameter computations

We will exemplify the computations of the pseudoword parameters seen in the previous section and the analysis of Sect. 4.1 with the aid of two very different pseudowords: *beer_tequila* and *barque_pennywhistle*. The word *beer* is of frequency class 5, *tequila* of frequency class 4, and both *barque* and *pennywhistle* belong to frequency class 1. For the first couple we expect a large overlap, while the elements in the second couple look unrelated[14]. We will first examine their parameters in the semantic-similarity-based data set.

The pseudoword *beer_tequila* is unbalanced in favour of *beer*: the set α contains 139 words like

`{Heineken, tofu, sirloin, soft-drink, fajita,...}`,

mostly about beer brands, food and beverages, while β, for *tequila*, has only 15, with a more restricted range:

`{Beam, vermouth, Burgundy, Sauvignon,...}`.

The balance parameter is then

$$\rho = \max\left(\frac{139}{15}, \frac{15}{139}\right) \cong \max(0.11, 9.27) = 9.27,$$

far from 1 and demonstrating the skewness of this pseudoword.

The set γ is very large here, with cardinality 305, for a high relative overlap $\kappa = \frac{305}{492} \cong 0.62$. The set γ contains terms such as

`{broth, tapa, syrup, roast, lobster, bourbon,...}`.

The set δ, which contains "new information", has 33 elements, including terms not directly related to food or beverages, like

[14] *barque* is another word for *ship*, while *pennywhistle* is a small, inexpensive flute.

`{detergent, petroleum, shampoo, julep,...}`.

The behaviour of this pseudoword changes in the co-occurrence-based data set, where the high frequency class of *beer* and the very similar senses of the two components make it collapse. Out of 485 nodes, the common terms in γ are 332, more than for the semantic-similarity-based ego graph, for an overlap of $\kappa = 0.68$, but $|\beta| = 0$ and $|\delta| = 0$, so that this ego graph is a near-replica of the first-order distributional thesaurus of just *beer*. We notice that the terms in α and the common terms in γ approximately repeat the same themes as in the semantic-similarity-based ego graph, as respectively in

`{Oktoberfest, Schlitz, license, belgian-style,...}`

and

`{produce, gallon, drunk, distilled, go, lunch,...}`.

The pseudoword *barque_pennywhistle* is more balanced than *beer_tequila* in the semantic-similarity-based data set, a fact due to the same, low frequency class of both components: we have 37 words for barque, among which

`{ship, yacht, plane, Fock, sailboat, galleon,...}`,

and 96 for pennywhistle, among which

`{kazoo, organ, horn, tuba, concertina, solo,...}`,

for a balance parameter $\rho = 2.60$. There is no overlap: the set γ of words common to both distributional thesauri is empty, so that $\kappa = 0$, as could be expected from words referring to two such dissimilar objects. Still, we have 11 terms in δ. Terms like *bell* or the imitative *twang* might be put in relation both with the nautical sphere (the ship's bell, the sound of a tight rope) and with sounds and musical instruments. Other terms like *poetry* are more difficult to interpret.

Passing to co-occurrences does not change the picture much. The ego graph is even more balanced now: 109 nodes for *barque* against 108 for *pennywhistle*, obtaining $\rho = 1.01$. For *barque* we have e.g.,

`{participate, 179-foot, brig, aircraft, Sodano,...}`,

and for *pennywhistle*

`{musical, jazz, most, drum, pipe, balladeer,...}`.

We note two things. First, the word *most*, normally considered a stop word, testifies the more noisy nature of co-occurrence-based ego graphs (we also recall from Sect. 5.1.1 that our co-occurrence ego graphs can be composed of any kinds of word classes); second, the nature of co-occurrences is shown by the apparently out-of-place *Sodano*, the name of a former Cardinal Secretary State of the Vatican, which appears because of the *barque of St. Peter*, a metaphor for the Catholic Church.

The remaining 6 nodes of the few total 223 nodes (due to the low frequency classes of both components) of the network belong to γ, for a vanishing relative overlap of $\kappa = 0.03$, in line with $\kappa = 0$ for semantic-similarity-based ego graphs. The common words are

`{same, way, time, own, more, include}`,

which can be considered stop words (e.g., *same*) or very generic words (e.g., *time*). We observe that they will be ignored during our evaluation, since we do not consider the sets γ and δ, as motivated in Sect. 4.1.

6 Results

In this section we will present and compare the main results on the two pseudoword ego graph data sets, obtained by the clustering algorithms detailed in Sect. 5.1.3 with the additional application of hyperclustering (cf. Sect.5.1.4), evaluated with the measures mentioned and described in Sect. 5.1.5.

Throughout this section we will refer to a clustering obtained directly from an ego graph as the *basic clustering* (consequently consisting of *basic clusters*), opposed to its hyperclustering (consisting of hyperclusters), and will use the following notation for algorithms and evaluation measures:

MCL2:	Markov cluster algorithm, expansion 2, inflation 2
MCL14:	Markov cluster algorithm, expansion 2, inflation 1.4
CW:	Chinese Whispers
MM:	MaxMax
GP:	gangplank clustering algorithm
AGG97:	aggregative clustering algorithm, $\sigma = 0.97$
AGG75P:	aggregative clustering algorithm, $\sigma =$ 75th percentile of all distance values on the pseudoword graph
CURV:	curvature-based clustering algorithm
BSL:	one-cluster-per-word baseline
HL:	HyperLex with mean distance threshold $\sigma = 0.8$
BC-F:	BCubed F-score (harmonic mean of BCubed precision and recall)
NMI:	normalised mutual information
TOP2:	TOP2 score, valuing the two best clusters
NOR:	basic clustering
HYP:	hyperclustering

6.1 Overall mean scores and numbers of clusters

Firstly, we display the overall scores achieved by each clustering algorithm in both cases of semantic-similarity- and co-occurrence-based ego graphs. The numbers in Tables 1 and 2 represent the mean scores over all pseudowords together with their respective 95% confidence intervals. The best mean scores for each evaluation measure are highlighted. As we point out in Sect. 4.1.1, we do not consider collapsed pseudowords in our evaluation. We write side by side the scores achieved by a basic clustering and its corresponding hyperclustering.

Similarly, Table 3 shows how many clusters are found by each clustering algorithm on average, and compares these numbers with the reduced size of the respective hyperclusterings.

Despite the biases that we will put in evidence in Sect. 6.1.2, Chinese Whispers emerges as the overall best system for semantic-similarity-based ego graphs. Its high TOP2 score reflects the fact that, even if it often conflates all elements into one single cluster, its average two clusters represent the two pseudosenses quite faithfully. This is also mirrored in the drop of scores for its hyperclustering: once the

Table 1 Overall mean scores over all pseudowords in the semantic-similarity-based ego graph data set, for all clustering algorithms, both for basic clusterings and their respective hyperclusterings

	BC-F		NMI		TOP2	
	NOR	HYP	NOR	HYP	NOR	HYP
MCL2	67.8 ± 1.2	*76.5* ± 0.9	39.4 ± 1.6	27.5 ± 1.7	67.1 ± 1.5	56.2 ± 1.7
MCL14	93.0 ± 0.6	**91.0 ± 0.7**	53.0 ± 2.6	36.3 ± 2.8	72.4 ± 1.8	61.2 ± 1.9
CW	**94.7** ± 0.5	85.1 ± 0.7	**53.2** ± 2.7	0.5 ± 0.4	**73.9** ± 1.6	40.5 ± 0.5
MM	18.8 ± 0.5	*75.9* ± 0.8	27.3 ± 0.9	***37.9*** ± 1.8	39.7 ± 0.8	***67.2*** ± 1.5
AGG97	68.8 ± 1.1	*85.1* ± 0.7	42.6 ± 1.8	0.2 ± 0.3	67.6 ± 1.3	41.2 ± 0.4
AGG75P	58.8 ± 1.2	*81.6* ± 0.7	39.9 ± 1.8	14.2 ± 1.7	66.1 ± 1.4	41.2 ± 1.6
CURV	70.2 ± 1.0	*70.3* ± 1.0	4.9 ± 0.3	4.7 ± 0.3	34.6 ± 1.0	34.2 ± 1.0
GP	55.0 ± 1.2	*78.4* ± 0.8	30.4 ± 1.4	*35.9* ± 2.0	58.6 ± 1.2	*64.5* ± 1.4
BSL	85.1 ± 0.7	–	0.0 ± 0	–	41.1 ± 0.4	–
HL	88.6 ± 0.5	87.3 ± 0.5	12.3 ± 1.2	0.0 ± 0.0	28.0 ± 0.6	24.4 ± 0.6

For each evaluation measure, the best score (respectively for normal clustering and hyperclustering) achieved by an algorithm is boldfaced. Italic values show an improvement from the normal clustering to the hyperclustering. The 95% confidence interval is also reported for each mean value

Table 2 Overall mean scores over all pseudowords in the co-occurrence-based ego graph data set, for all clustering algorithms, both for basic clusterings and their respective hyperclusterings

	BC-F		NMI		TOP2	
	NOR	HYP	NOR	HYP	NOR	HYP
MCL2	35.3 ± 0.7	*45.7* ± 1.0	9.4 ± 0.3	**8.5** ± 0.3	33.4 ± 0.5	*34.6* ± 0.7
MCL14	69.1 ± 0.9	*77.2* ± 0.7	5.4 ± 0.3	4.0 ± 0.2	39.3 ± 0.7	35.6 ± 0.9
CW	88.7 ± 0.5	**90.3** ± 0.4	4.1 ± 0.4	0.1 ± 0.0	25.6 ± 1.1	*34.9* ± 0.7
MM	35.2 ± 0.7	*51.0* ± 0.7	**11.1** ± 0.4	5.8 ± 0.3	34.2 ± 0.6	*36.7* ± 0.7
AGG97	50.3 ± 0.8	*81.5* ± 0.7	**11.1** ± 0.5	3.9 ± 0.4	41.7 ± 0.6	39.9 ± 0.8
AGG75P	55.9 ± 0.7	*82.6* ± 0.6	10.1 ± 0.5	3.8 ± 0.4	**42.8** ± 0.7	38.0 ± 0.9
CURV	77.5 ± 0.9	77.5 ± 0.9	4.8 ± 0.3	4.5 ± 0.3	35.8 ± 1.0	35.3 ± 1.0
GP	58.2 ± 2.0	*76.9* ± 1.0	4.2 ± 0.4	2.1 ± 0.3	35.4 ± 0.5	**40.6** ± 0.5
BSL	**90.5** ± 0.4	–	0.0 ± 0	–	38.8 ± 0.5	–
HL	89.7 ± 0.3	*90.5* ± 0.3	0.7 ± 0.1	0.0 ± 0.0	33.2 ± 0.6	38.8 ± 0.3

For each evaluation measure, the best score (respectively for normal clustering and hyperclustering) achieved by an algorithm is boldfaced. Italic values show an improvement from the normal clustering to the hyperclustering

peak has been reached, performances can only decline, and more so if we obtain just one single hypercluster. The MCL and aggregative variants also perform analogously.

For co-occurrences, the picture is not as distinct, although the aggregative clustering algorithm with variable radius based on the 75th percentile and the gangplank clustering algorithm seem to perform slightly better than the others.

Table 3 Overall mean numbers of clusters over all pseudowords for all clustering algorithms, comparing basic clusterings and their respective hyperclusterings

	Similarities		Co-occurrences	
	NOR	HYP	NOR	HYP
MCL2	18.7 ± 0.8	7.8 ± 0.3	50.2 ± 0.9	38.8 ± 0.6
MCL14	2.9 ± 0.1	1.6 ± 0.0	14.1 ± 0.3	9.0 ± 0.3
CW	2.0 ± 0.1	1.1 ± 0.0	4.0 ± 0.1	1.1 ± 0.0
MM	43.8 ± 0.8	5.0 ± 0.1	42.9 ± 0.6	8.3 ± 0.2
AGG97	5.4 ± 0.2	1.0 ± 0.0	7.9 ± 0.2	1.9 ± 0.0
AGG75P	6.8 ± 0.2	1.0 ± 0.0	6.5 ± 0.2	1.8 ± 0.0
CURV	10.4 ± 0.3	9.7 ± 0.3	8.7 ± 0.2	8.1 ± 0.2
GP	7.5 ± 0.2	2.5 ± 0.0	6.3 ± 0.3	2.0 ± 0.1
BSL	1	1	1	1
HL	1.2 ± 0.0	1	1.3 ± 0.0	1

The 95% confidence interval is also shown for each mean value

Aided by the more dispersive nature of co-occurrence word graphs, already observed in the analyses of Sects. 4.1, 4.1.1 and 5.2, and by the bias that will be discussed in Sect. 6.1.2, our trivial baseline outclasses all other algorithms in terms of BCubed measure, and has an average TOP2 score. For the same reason, scores for co-occurrences are generally lower than for semantic similarities. Further, in the case of co-occurrences we see that hyperclustering can indeed help a basic clustering find a better focus, independently from the fine-grainedness of an algorithm. This principally backs up the impression that co-occurrence clustering is a harder task to tackle, a fact that might be tied to the small-world nature of word graphs (mentioned in Sect. 5.1.1).

We want to focus on two interesting phenomena that arise from our pseudoword-based comparisons. The first one is the pronounced decrease in performance of Chinese Whispers on the co-occurrence-based data set with respect to semantic similarities. There, among basic clusterings, Chinese Whispers regularly obtains the best scores for most frequency class combinations of the pseudoword components, and is followed closely by the MCL variants, especially the one with inflation parameter 1.4. A drop in TOP2 performances is already seen in CW's hyperclustering, because of its tendency to coincide with the trivial baseline. This behaviour intensifies on co-occurrence-based ego graphs: its clusterings very often consist of one big cluster, possibly accompanied by unmeaningful micro clusters, and, while BCubed measures do not penalise this fact, TOP2 makes it apparent with a sharp drop, especially for the most unbalanced frequency class combinations (such as 14 or 25). We will further analyse this point in Sect. 7.

The second phenomenon is the surprising behaviour of MaxMax. Its basic clustering is by far the worst system together with CURV: it tends to score quite low, confronted with the other algorithms, for all three evaluation measures. This is surely connected to its extreme fragmentation, as it very often induces more than 40 clusters. Even if their precision is high, their recall is very low, and TOP2 can not find two clusters that are representative enough. The scenario is totally changed by the intervention of hyperclustering: then, the number of clusters becomes sensible and

MaxMax often achieves the best scores among hyperclusterings, especially for NMI and TOP2. This means that MM adapts particularly well to hyperclustering and that the highly precise smaller building blocks are effectively combined in meaningful bigger entities. Still, this does not seem to hold for every dispersive clustering algorithm: on the co-occurrence-based data set, MCL2 even surpasses MaxMax as the most fine-grained algorithm, but its hyperclusterings remain very splintered. We want to see a sort of paradigmatic difference in their clustering approaches: whereas MCL *isolates* denser combinations, MaxMax *congregates* the nodes in certain types of subgraphs. Such difference is accentuated by the hyperclustering: MaxMax goes on condensing its clustering, while MCL retains its divisive nature. We formalise this concept in Sect. 7.

6.1.1 HyperLex and possible limits of our evaluation framework

In Sect. 5.1.3 we give a very short description of HyperLex, a clustering algorithm explicitly devised for graph-based Word Sense Induction. However, its inclusion in our pseudoword evaluation framework is problematic. This is the reason why we show the results obtained by HyperLex in Table 1 and onwards, albeit without further discussion. Since HyperLex is a well-known algorithm in the field, we think that this issue nonetheless deserves some additional explanations; in fact, a brief discussion of the encountered problems might be helpful in singling out some possible limits of our evaluation framework.

We want first to highlight the relevant points which might represent difficulties for our approach:

1. HyperLex uses paragraphs, and not sentences, as the text units for detecting word co-occurrences;
2. edge weights in the resulting word graph G are expressed as distances based on estimated probabilities, and not as similarities. Edges are then pruned according to a given threshold before clustering;
3. clustering occurs by means of the clustering coefficient[15] of a node, estimated through a mean edge weight;
4. in the end, clusters are defined by the structure of a minimum spanning tree of G starting from some selected root nodes.

In all these points, the concept of distance is crucial and is exploited to obtain the final result. We also remark that HyperLex is explicitly geared towards co-occurrence-based word graphs (even if we might argue that choosing paragraphs instead of sentences as text units steers the graph-building process towards a more topical, rather than merely syntactical, flavour). For these reasons, our main concern is that we are not able to guarantee a fair evaluation of HyperLex (or of similarly designed clustering methods) in our framework while staying true to its working

[15] A *clustering coefficient* of a node or a graph can be defined in different ways. The first definition of a local clustering coefficient is found in Watts and Strogatz (1998); a global one based on triangles is in Feld (1981); Karlberg (1997).

principles. In particular, we express this more formally by retracing the previous points.

Point 1: Our corpus consists only of sentences and is not organised in paragraphs (cf. Sect. 5.1.1). This means that, even on our co-occurrence-based data set, we are dealing with different types of co-occurrences.

Points 2, 3: LMI is a similarity measure and not a distance (see again Sect. 5.1.1). HyperLex works on the assumption that no edge in G has a weight greater than a given threshold (0.9) after the pruning phase. In our evaluation framework, we want the word graphs to be the same for each algorithm at the start of the process, so that we can compare them on the same structures and ground truths. Therefore we can not perform an *a posteriori* pruning. We are aware of the possibility of "reversing" the algorithm to make it work with similarities instead of distances (we find e.g., a similar readaptation in Marco and Navigli (2013)), but recomputing weights on our edges would also change the structure of the underlying word graph.

Point 3: Further, the previous point becomes even more apparent in the case of our semantic-similarity-based data set. There, edges represent second-order similar contexts, without necessarily corresponding to sentence or paragraph-level co-occurrences. As a consequence, on these word graphs we would not be able to compute all co-occurrence-based distances needed by HyperLex, and we have to resort to a substitute value, like e.g., 1. However, this way we are again radically altering the structure of the word graphs.

Points 3, 4: HyperLex is designed to deal with quite large graphs, and as such it is badly suited for hyperclustering (see Sect. 5.1.4) on our data sets. Since our hypergraphs are too small in size, quite often, even when tuning the algorithm's parameters, no hypercluster is found.

All in all, HyperLex is not as modular as the other clustering algorithms we consider in this work, and instead requires a specific graph-building step that we can not well enough replicate in our current setting. Also, the optimal tuning of the various parameters of HyperLex is beyond the scope of this work.

We think that the strength of our framework is that we can highlight the behaviour of different algorithms when they are confronted on the base of word graphs with identical structures, all the while gaining more insight into their single dynamics. We do this by keeping the graph-building and the clustering processing steps separate; unfortunately, the line between them is extremely blurred with HyperLex (and with other similarly principled systems too), and that is why we deem to include it only collaterally in this work.

6.1.2 Measure biases

In the process of our pseudoword evaluation, we claim to have gathered further evidence of the biases of BCubed measures and NMI.

BCubed measures (Bagga and Baldwin 1998; Amigó et al. 2009), due to their nature as averages over all single clustered elements, stress the similarity between the *internal* structures, i.e., the distribution of elements inside each cluster, of two clusterings and disregard their *external* structures, i.e., their respective sizes and the distribution of cardinalities among clusters. The fact that many pseudowords in both our data sets are very unbalanced (in the sense explored in Sect. 5.2) and that e.g., CW tends to produce coarse clusterings originates extremely high BCubed scores for this algorithm, as this measure does not give enough relevance to the smaller of the two ground truth clusters of (3). In fact, if the algorithm manages to find only one cluster, the BCubed score is the same as for the one-cluster-per-word trivial baseline BSL, which itself corresponds to the value $\frac{\max(|\alpha|,|\beta|)}{|\alpha|+|\beta|}$, in the notation of identity (1): the more unbalanced the pseudoword, the higher this value. The same considerations are valid for MCL with parameters (2, 1.4) and less pronouncedly for the aggregative clustering algorithm with radius $\sigma = 0.97$. We might argue that if a pseudoword is nearly collapsed onto one single component, a high BCubed score effectively recognises this and rewards the algorithm for not being too fragmentary. Still, we want an algorithm to isolate both components, and we deem that TOP2 is better suited to evaluate this in the case of clear-cut homonymy.

On the other hand, NMI tendentially penalises a small number of clusters too heavily, and it seems sometimes quite difficult to interpret, as its variations are too brisk; in general, though, NMI seems more adherent to the desired kind of evaluation (as discussed in Sect. 5.1.5 for TOP2) than BCubed. Still, this briskness does not take account of the skewness of a pseudoword at all. We see this by the example of two pseudowords, in the case of similarity-based ego graphs, with very different balance parameters: *paycheck_reptile*, fairly balanced with $\rho = 2.0$ (160 terms for *paycheck*, 322 for *reptile*), and *artistry_bufflehead*, with a very high $\rho = 35.62$ (413 terms for *artistry* and only 13 for *bufflehead*). We will consider the algorithm AGG75P and its hyperclustering. For *paycheck_reptile*, 2 good clusters are found, for an excellent NMI score of 0.96; for *artistry_bufflehead*, the clustering is quite fragmented and yields 9 clusters that score an NMI of 0.28. Now, both hyperclusterings are greatly condensed to one single hypercluster. Then, for both the NMI score is exactly 0, as is always the case when one single cluster is found. However, the TOP2 score behaves differently: in the first case we obtain 0.39, while in the second one 0.48. Despite being both under the threshold of $\frac{1}{2}$ (see again Sect. 5.1.5), the higher, second one mirrors the intuition that it is still "less wrong" to assign a single cluster to a very unbalanced sense distribution (like for *artistry_bufflehead*) than to a more uniform one (like for *paycheck_reptile*). In the end, normalised mutual information is too much influenced by small differences in size between the compared clusterings and too susceptible to outliers.

6.2 Examples of clusterings

We briefly want to show the differences between the basic clusterings of our main six systems (MCL, CW, MM, AGG, GP, CURV) and their variants on the ego graph of a same pseudoword, in order to give a small direct insight into their nature. We will also briefly look at the effects of hyperclustering. As our example, we chose the similarity-based ego graph of *euonymus_carryall*, a combination of two words belonging to the lowest frequency class 1. Its graph is connected and consists of only 336 nodes out of a possible maximum of 500; this is due to the rarity of such words. The graph is balanced, with a ratio ρ (cf. Sect. 5.2) between *euonymus* and *carryall* of 1.09. The baseline has here a BCubed F-score of 0.67, a TOP2 score of 0.31 and of course a NMI of exactly 0.

Chinese Whispers finds two clusters, which seem to correspond on one hand to the sense of *plant*, as in

`{nandina, hemlock, reed, iris,...}`,

and on the other hand to the sense of *transportable container*, as in

`{can, receptacle, scarf, compartment,...}`.

As a consequence, all scores rate very high, respectively 0.97 for BCubed F-score, 0.90 for NMI and 0.99 for TOP2.

The gangplank algorithm gives us 8 clusters. Their precision is quite high and it is possible to distinguish *carryall*-type clusters like

`{apartment, package, sheeting, canopy}`,

and *euonymus*-type ones, although the distinction made between two clusters of the same type is not always clear, like for the plant-related terms of

`{clethra, tree, spiraea}` and

`{loosestrife, gilt, multiflora, bugger, leucothoe,...}`.

The gangplank scores are: BCubed F-score 0.70, NMI 0.61, TOP2 0.78.

After hyperclustering, we get 3 more clear-cut clusters, similar to those obtained by CW. One is just a micro cluster of three terms with no relevance. Scores improve consequently: BC-F 0.94, NMI 0.80, TOP2 0.90.

The Markov cluster algorithm with parameters (2, 2) returns a more fragmented result, with 11 clusters. Some of them appear to be very specific, like

`{bike, humvee, skateboard, car, Cruiser}`,

together with a pair of singletons like

`{wheel}` or

`{look}`.

We can still separate *euonymus* from *carryall*, but recall penalises the scores and we have a BCubed F-score of 0.73, a NMI of 0.63 and a TOP2 of 0.84: in this case higher than GP, but worse than GP's hyperclustering.

When using 1.4 as the inflation parameter, though, MCL nearly acts as the hyperclustering of its version with inflation parameter 2. We retrieve just 2 clusters where the two pseudosenses are clearly distinct, and we obtain extremely high scores: BCubed F-score 0.97, NMI 0.9, TOP2 0.92. Its hyperclustering stays unvaried.

MaxMax takes the tendency of the more fragmented MCL version even further and produces 21 clusters, some of them consisting only of two or three terms, like

`{pack, equip}` and

`{style, lash, look}`,

and overall with very restricted meaning areas. The biggest cluster comprises 56 elements. Despite the very high precision, its scores are the lowest ones between basic clusterings, with a BCubed F-score of 0.34, a NMI of 0.48 and a TOP2 of 0.47. However, the hyperclustering causes here a major improvement. The clustering becomes condensed in 3 clusters, with the *carryall* sense split among one bigger and one smaller cluster:

`{humvee, bike, car, cart, Cruiser, skateboard,...}`,
`{lawn, eraser, moss, brush, reed, grass, wedge,...}` and
`{eyeglass, overalls, cap, parka, sweater,...}`.

This new clustering reaches a BCubed F-score of 0.62, a NMI of 0.37 and a TOP2 of 0.66. It is remarkable how normalised mutual information unpredictably decreases, despite the evident overall better look. MaxMax's hyperclustering is capable of achieving even better improvements than for *euonymus_carryall.*

Between the two aggregative clustering variants, one with variable radius (AGG75P) and the other with a fixed radius of 0.97 (AGG97), the latter has a more dispersed clustering, with 6 clusters against the 3 found by the former. This is explained by the fact that the ego graph of *euonymus_carryall* is very balanced, so that distances are more homogenously distributed, producing a very high 75th percentile. The clusters of AGG97 are quite balanced and the TOP2 score is still quite good: 0.72, compared to a BCubed F-score of 0.63 and a NMI of 0.67. Precision is good, but the pseudosenses are split among the clusters. AGG75P is more focused, with only one smaller spurious cluster pertaining to *carryall*:

`{humvee, bike, car, cart, Cruiser, skateboard,...}`,
`{lawn, eraser, moss, brush, reed, grass, wedge,...}` and
`{eyeglass, overalls, cap, parka, sweater,...}`.

This variant thus obtains much better results indeed: BCubed F-score 0.93, NMI 0.82, TOP2 0.90.

The two respective hyperclusterings also behave differently: AGG97 merges everything into one big cluster, because of its high fixed radius, while AGG75P, by its own definition, will always find at least two different clusters in a hypergraph with at least three nodes and different weights on its edges.

The curvature-based clustering algorithm identifies 9 clusters. Their distribution is very skewed: there are two bigger clusters with respectively 283 and 29 terms:

`{equip, apple, twig, style, lunchbox, toothpick, pachysandra,...}`,
`{nandina, brake, wheel, rosebush, poppy,...}`.

The rest is subdivided among the remaining micro clusters. The clusters appear quite confused, and scores are consequently low: BCubed F-score 0.61, NMI 0.06, TOP2 0.42. The scores stay substantially unvaried for the hyperclustering, since the only change is that two minor clusters are merged, namely:

`{cultivar, gizmo, daylily, weed, boxwood, hibiscus}` to
`{rucksack, restraint, leaf, brooch, tree, bulb, button-downs}`.

Table 4 Average values of the mean absolute deviations for each clustering algorithm over both ego graph data sets

	Similarities	Co-occurrences
MCL2	51.8	13.0
MCL14	92.0	54.6
CW	66.6	160.5
MM	10.6	12.4
AGG97	66.3	59.4
AGG75P	50.7	69.6
CURV	80.8	90.8
GP	40.6	12.2
BSL	0	0
HL	16.2	69.6

7 Discussion of results and conclusions

In this final Section, we link the performances of the algorithms to the nature of the word graphs they are run on, and introduce some quantities that describe the clustering tendencies of the algorithms, explaining how they interact with different word graphs and pseudoword properties.

7.1 Discussion of Results

The analysis of the algorithms' behaviours on both the semantic-similarity and on the co-occurrence-based ego graph data sets in Sect. 6, together with the more detailed insight into the nature of their clusterings of Sect. 6.2, allow us to make some final considerations about the functioning and the desired properties of a clustering algorithm for the specific task we have chosen to study.

First, we observe that coarse-grainedness is an advantage in our evaluation framework: in fact, in the ideal case we want an algorithm to produce exactly two clusters, and the TOP2 score penalises outputs which deviate too much from this ideal number. Chinese Whispers is on average the most coarse-grained among our systems, and it is true that it manages to achieve very good TOP2 scores. However, this happens at the expense of stability: very often the more coarse-grained algorithms ignore smaller or not well-represented differences and conflate all elements into just one cluster. Their good average scores arise from the means of both extremely high and very low values.

On the other hand, more fine-grained clustering algorithms might be more sensitive to minor, coherent subregions in a word graph and be stabler in terms of score variance, but their lack of focus and more dispersed clusterings give them inferior average scores; the best example for this is MaxMax. Considering semantic-similarity-based ego graphs, the curvature-based clustering algorithm is also stably floating around low scores. Still, we recognise somewhat different splitting behaviours that we want to summarise using the *mean absolute deviation* (MAD), a measure of dispersion,[16] together with the mean number of clusters in a clustering (Table 3). We apply the MAD to the cardinalities of the clusters in a given clustering, and for each algorithm we take the mean of the MAD scores of all its clusterings on either data set.[17] The mean absolute deviation tells us how unbalanced the distribution of elements in a clustering is with respect to a hypothetical uniform clustering where all clusters are of equal size. The higher the MAD, the more skewed the clustering: this means that there will be few very big clusters counterposed to many smaller clusters. In Table 4 we show the MAD values for our algorithms on both data sets. The mean number of clusters, instead, tells us how fragmented a clustering is. Figure 4 graphically shows the position of our algorithms with respect to these two statistics, which we will comment comparing them to their respective TOP2 scores.

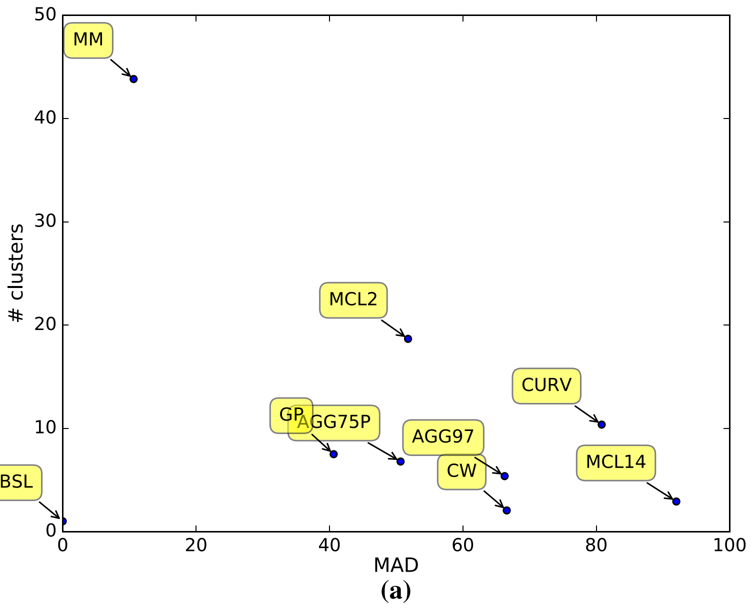
We observe for example that CURV, though only relatively fine-grained, has quite high mean absolute deviation values, much higher than MaxMax, which however suffers from an extremely high fragmentation. On the contrary, GP is more balanced, because, although having a higher MAD, it is less fine-grained, which lends to its clustering more compactness than the two other aforementioned systems. However, at the same time, it is exactly this same compactness that is detrimental to its TOP2 scores, since the recall of its representative clusters will be relatively low: we know that in most cases of homonymy one term is prevalent over the other, as mentioned in Sect. 4.1.1.

We claim that the best clustering shape that can be obtained by an algorithm for the task of homonymy detection, and more generally of Word Sense Induction, is one that admits a pronounced, even if not exasperated, degree of skewness without being too much fragmented, i.e., with too many clusters. In the charts of Fig. 4, this would correspond to a rather undefined region towards their bottom center-right quarter, where the best systems tend to concentrate (cf. Tables 1 and 2), especially in Fig. 4b (AGG75P, AGG97, MCL14 and CURV). As a matter of fact, CW's performances suffer on the co-occurrence-based ego graph data set (Fig. 4b) because its good shape seen on the semantic-similarity-based data set is lost, or, in other terms, because, while its mean clustering size remains stably low, its MAD value gets too high.

An interesting factor emerges from hyperclustering: we might call it the *scalability*, or conversely *rigidity*, of a clustering algorithm. Namely, we already noticed how MaxMax's hyperclustering markedly improves with respect to its basic clustering. At the same time, we observe that MaxMax's hyperclustering greatly rescales the size of the basic clustering. On the contrary, the basic clusterings of MCL2 and especially of the curvature-based algorithm are largely unaffected by hyperclustering. This hints to different properties of the algorithms, particularly when noting that MCL2 is as fragmented as MM on the co-occurrence-based data set (cf. Table 3). In the notation of Sect. 5.1.4, we express this ratio as

[16] The mean absolute deviation of a data set of observations is the average of the absolute values of the differences between the mean data set value and the observations (Dixon and Massey 1957).

[17] We could normalise the MAD score with respect to the number of total clustered elements. However, since the order of our ego graphs is nearly constant, we just take the absolute mean deviations. The same goes for the mean number of clusters.



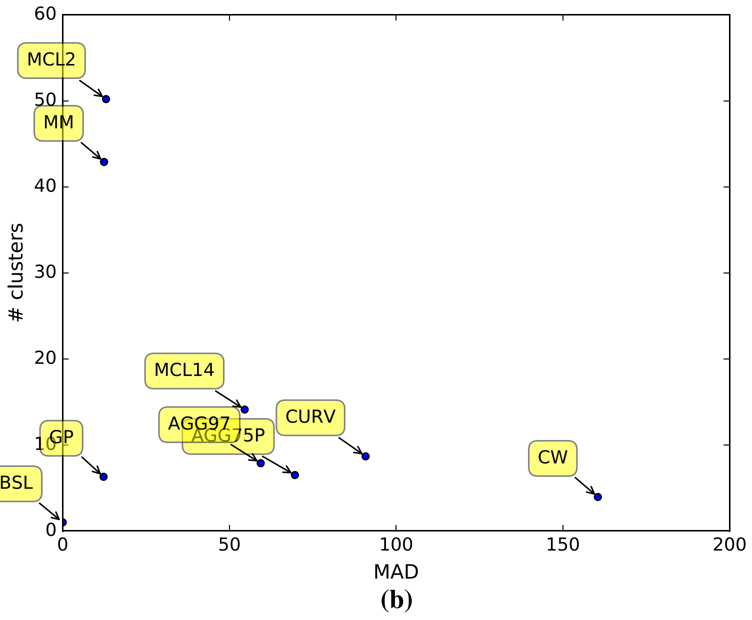


Fig. 4 Distribution of the clustering algorithms with respect to mean absolute deviation (MAD, x-axis) on the cluster size distribution in their clusterings and mean number of produced clusters (y-axis). **a** Semantic-similarity-based data set, **b** Co-occurrence-based data set.

Table 5 Mean ratios of the number of clusters in a basic clustering to the same number in its hyperclustering (rigidity), for each algorithm, over both ego graph data sets

	Similarities	Co-occurrences
MCL2	2.5	1.3
MCL14	1.9	1.7
CW	1.8	3.5
MM	9.3	5.6
AGG97	5.3	4.3
AGG75P	3.8	3.9
CURV	1.1	1.1
GP	3.1	2.6
HL	1.2	1.3

$$\zeta = \frac{|\mathcal{C}|}{|\mathrm{Hyp}(\mathcal{C})|} \geq 1 \tag{13}$$

and summarise its values for each algorithm in Table 5.

We can say that the closer ζ to 1, the greater the *invariance* of an algorithm with respect to hyperclustering. This invariance is surely tied to coarseness, albeit it does not directly depend on it. We might claim that clustering algorithms with low invariance and a relatively great fragmentation, like MM or GP, benefit the most from hyperclustering. This is due to the fact that the algorithm is capable of *rescaling* itself and efficiently use its relatively small and limited basic clusters as building blocks for more significant clusters.

Conversely, we recognise a generic turning point in terms of coarseness beyond which hyperclustering ceases to be useful: when a clustering is already coarse enough, i.e., close to the supremum of the partition lattice (cf. Sect. 5.1.4), the hyperclustering exasperates this coarseness, lowering its quality and the associated scores. Chinese Whispers is a clear example thereof: on the semantic-similarity-based ego graph data set its hyperclustering very often, if not always coincides with the trivial baseline, and indeed its scores drastically reduce, especially for the more balanced pseudowords.

7.2 Final conclusions

The first main contribution of this article is to present a self-sufficient pseudo-word-based evaluation framework for WSI graph-based clustering algorithms, thereby defining a new evaluation measure (TOP2, cf. Sect. 5.1.5) and a secondary clustering process (hyperclustering, cf. Sect. 5.1.4).

The second main contribution is a thorough comparison of well and less known WSI graph-based clustering algorithms, each with a different approach to the task, on the presented evaluation framework. Even if we observe some clustering algorithms excelling at some evaluation measure more than others, in the end we are effectively not able to proclaim one of them as the "overall best WSI clustering algorithm". Here we are also refraining from any conclusion about HyperLex, as motivated in Sect. 6.1.1.

The type of an ego graph, either based on (second-order) semantic similarities or on (first-order) co-occurrences, influences the values of the mean absolute deviation and to a much lesser extent the invariance ratio ζ defined by (13) in Sect. 7.1: different clustering algorithms react differently to first-order or second-order relations. As we have stressed many times throughout the analysis of the previous sections, co-occurrence-based word graphs represent more unpredictable, generally less significant connections between words than similarity-based word graphs. They are also more subject to noise in the form of words with little relevance to WSI or even punctuation, depending on the pre-processing that was performed on the original raw text. Co-occurrence word graphs in our data set are sensibly denser than their similarity-based counterparts and their small-world nature (cf. Sect. 5.1.1) is more pronounced, and this is a cause of generally more fragmented and more blurred clusters, corresponding to overall lower scores. In such a setting, we can distinguish a disadvantage for clustering algorithms that rely on the abstract concept of *information flow*, like the MCL variants or CW, compared to others that are based on distance, like the AGG variants, or that just check some local properties, like MM or CURV, especially for the more balanced pseudowords. We see a reason for this in the fact that co-occurrences have a local essence, and that we can not as easily associate them semantic reasonings as with similarities, like an information flow implies; we rather observe that the emerging syntactic relations possess a different nature and are in a complementary syntagmatic vs. paradigmatic relation (the work De Saussure (1916) sets the grounds of this distinction), mirrored by the complementary relation between Word Sense Discrimination and Word Sense Induction. Even if they are often used as synonyms, this study seems to put in evidence the need to approach these tasks with different instruments.

As a final remark, we want to highlight the paramount importance of text pre-processing in the outcome of WSI clustering algorithms. This determinant step is often not given enough relevance, even if it greatly influences text modellisations, a fact that indirectly emerges from our work. In the light of this, besides believing that these connections need further investigation, we think that it might be useful for WSI evaluation campaigns or frameworks to be performed not only on a shared data set, but also on the same given type of structure.

References

Amigó, E., Gonzalo, J., Artiles, J., & Verdejo, F. (2009). A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, *12*(4), 461–486.

Bagga, A., Baldwin, B. (1998). Algorithms for scoring coreference chains. In *Proceedings of the first international Conference on Language Resources and Evaluation (LREC'98), workshop on linguistic coreference* (pp. 563–566). European Language Resources Association, Granada, Spain.

Barabási, A. L., & Albert, R. (1999). Emergence of scaling in random networks. *Science*, *286*(5439), 509–512.

Başkaya, O., & Jurgens, D. (2016). Semi-supervised learning with induced word senses for state of the art word sense disambiguation. *Journal of Artificial Intelligence Research*, *55*, 1025–1058.

Berge, C., & Minieka, E. (1973). *Graphs and hypergraphs* (Vol. 7). Amsterdam: North-Holland.

Biemann, C. (2006). Chinese whispers: An efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of the first workshop on graph based methods for natural language processing* (pp. 73–80), New York, NY, USA.

Biemann, C., & Quasthoff, U. (2009). Networks generated from natural language text. In N. Ganguly, A. Deutsch & A. Mukherjee (Eds.), *Dynamics on and of complex networks: Applications to biology, computer science, and the social sciences* (pp. 167–185). Springer.

Biemann, C., & Riedl, M. (2013). Text: Now in 2D! a framework for lexical expansion with contextual similarity. *Journal of Language Modelling*, *1*(1), 55–95.

Bordag, S. (2006). Word sense induction: Triplet-based clustering and automatic evaluation. In *Proceedings of the 11th conference of the European chapter of the association for computational linguistics* (pp. 137–144). EACL, Trento, Italy.

i Cancho, R. F., & Solé, R. (2001). The small world of human language. *Proceedings of the Royal Society of London Series B: Biological Sciences*, *268*(1482), 2261–2265.

Cecchini, F. M. (2017). Graph-based clustering algorithms for word sense induction. Ph.D. thesis, Università degli Studi di Milano-Bicocca.

Cecchini, F. M., Fersini, E. (2015) . Word sense discrimination: A gangplank algorithm. In *Proceedings of the second Italian conference on computational linguistics CLiC-it 2015* (pp. 77–81). Trento, Italy.

Cecchini, F. M., Fersini, E., & Messina, E. (2015). Word sense discrimination on tweets: A graph-based approach. In *KDIR 2015—Proceedings of the international conference on knowledge discovery and information retrieval* (Vol. 1, pp. 138–146). IC3K, Lisbon.

Cover, T., & Thomas, J. (2012 [1991]). Elements of information theory. Wiley, Hoboken, NJ.

De Marneffe, M. C., MacCartney, B., & Manning, C. (2006) . Generating typed dependency parses from phrase structure parses. In *Proceedings of the fifth international conference on language resources and evaluation (LREC'06)*, 2006 (pp. 449–454). European Language Resources Association, Genoa.

De Saussure, F. (1916) . Cours de linguistique générale. Payot&Rivage, Paris, France (1995 [1916]). Critical edition of 1st edition

Di Marco, A., & Navigli, R. (2013). Clustering and diversifying web search results with graph-based word sense induction. *Computational Linguistics*, *39*(3), 709–754.

Dixon, W., & Massey, F, Jr. (1957). *Introduction to statistical analysis*. New York, NY: McGraw-Hill.

van Dongen, S. (2000). Graph clustering by flow simulation. Ph.D. thesis, Universiteit Utrecht

Evert, S. (2004) . The statistics of word cooccurrences: Word pairs and collocations. Ph.D. thesis, Universität Stuttgart

Feld, S. L. (1981). The focused organization of social ties. *American Journal of Sociology*, *86*(5), 1015–1035.

Gale, W., Church, K., & Yarowsky, D. (1992) . Work on statistical methods for word sense disambiguation. In *Technical Report of 1992 fall symposium—Probabilistic approaches to natural language*, pp. 54–60. AAAI, Cambridge, Massachusetts, USA

Grätzer, G. (2011). *Lattice theory: Foundation*. New York: Springer.

Harris, Z. (1954). Distributional structure. *Word*, *10*(2–3), 146–162.

Haynes, T. W., Hedetniemi, S., & Slater, P. (1998). *Fundamentals of domination in graphs*. Boca Raton, FL: CRC Press.

Hope, D., & Keller, B. (2013). MaxMax: A graph-based soft clustering algorithm applied to word sense induction. In *Proceedings of the 14th international conference on computational linguistics and intelligent text processing* (pp. 368–381). Samos, Greece

Karlberg, M. (1997). Testing transitivity in graphs. *Social Networks*, *19*(4), 325–343.

Kilgarriff, A., Rychlý, P., Smrž, P., & Tugwell, D. (2004). The sketch engine. In *Proceedings of the eleventh Euralex Conference* (pp. 105–116). Lorient, France.

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, *28*(2), 129–137.

Lyons, J. (1968). *Introduction to theoretical linguistics*. Cambridge: Cambridge University Press.

Manandhar, S., Klapaftis, I., Dligach, D., & Pradhan, S. (2010) . Semeval-2010 task 14: Word sense induction & disambiguation. In *Proceedings of the 5th international workshop on semantic evaluation* (pp. 63–68). Association for Computational Linguistics, Los Angeles, CA.

Martin, J., & Jurafsky, D. (2000). *Speech and language processing*. Upper Saddle River, NJ: Pearson.

Miller, G. (1995). Wordnet: A lexical database for english. *Communications of the ACM*, *38*(11), 39–41.

Nakov, P., & Hearst, M. (2003). Category-based pseudowords. In *Companion volume of the proceedings of the human language technology conference of the North American chapter of the association for computational linguistics (HTL-NAACL)* 2003—Short Papers (pp. 70–72). Association for Computational Linguistics, Edmonton, Alberta, Canada.

Navigli, R. (2009). Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, *41*(2), 10.

Navigli, R., Litkowski, K., & Hargraves, O. (2007) . Semeval-2007 task 07: Coarse-grained english all-words task. In *Proceedings of the 4th international workshop on semantic evaluations* (pp. 30–35). Association for Computational Linguistics, Prague.

Otrusina, L., Smrž, P. (2010) . A new approach to pseudoword generation. In *Proceedings of the seventh international Conference on language resources and evaluation (LREC'10)* (pp. 1195–1199). European Language Resources Association, Valletta.

Parker, R., Graff, D., Kong, J., Chen, K., & Maeda, K. (2011) . English Gigaword, 5th edn. Linguistic Data Consortium, Philadelphia, PA. https://catalog.ldc.upenn.edu/LDC2011T07.

Pilehvar, M. T., & Navigli, R. (2013). Paving the way to a large-scale pseudosense-annotated dataset. In *Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: Human language technologies (HTL-NAACL)* (pp. 1100–1109). Association for Computational Linguistics, Atlanta, GA.

Pilehvar, M. T., & Navigli, R. (2014). A large-scale pseudoword-based evaluation framework for state-of-the-art word sense disambiguation. *Computational Linguistics*, *40*(4), 837–881.

Richter, M., Quasthoff, U., Hallsteinsdóttir, E., & Biemann, C. (2006) . Exploiting the Leipzig corpora collection. In *Proceedings of the fifth Slovenian and first international language technologies conference, IS-LTC '06* (pp. 68–73). Slovenian Language Technologies Society, Ljubljana.

Riedl, M. (2016) . Unsupervised methods for learning and using semantics of natural language. Ph.D. thesis, Technische Universität Darmstadt

Ruohonen, K. (2013) . Graph theory. Tampereen teknillinen yliopisto (trans: Tamminen, J., Lee, K.-C., & Piché, R.). http://math.tut.fi/~ruohonen/GT_English.pdf. Originally titled Graafiteoria, lecture notes.

Schütze, H. (1992) . Dimensions of meaning. In *Proceedings of Supercomputing'92* (pp. 787–796). ACM/IEEE, Minneapolis, MN.

Strehl, A., & Ghosh, J. (2002). Cluster ensembles–A knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, *3*, 583–617.

Turney, P., & Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, *37*(1), 141–188.

Véronis, J. (2004). Hyperlex: Lexical cartography for information retrieval. *Computer Speech & Language*, *18*(3), 223–252.

Watts, D., & Strogatz, S. (1998). Collective dynamics of small-world networks. *Nature*, *393*(6684), 440–442.

Widdows, D., & Dorow, B. (2002) . A graph model for unsupervised lexical acquisition. In *Proceedings of the 19th international conference on computational linguistics* (vol. 1, pp. 1–7). Association for Computational Linguistics, Taipei.