Unsupervised Word Sense Induction from Multiple Semantic Spaces with Locality Sensitive Hashing

Claire Mouton - CEA LIST - Exalead
CEA LIST - Claire.Mouton@cea.fr
Exalead - 10 place de la Madeleine - 75008 Paris
Claire.Mouton@exalead.com

Guillaume Pitel - Epiphyte
10 rue de Vouillé - 75015 Paris
Guillaume.Pitel@epiphyte.eu

Gaël de Chalendar - CEA LIST
18, route du Panorama
BP6, FONTENAY AUX ROSES
F-92265 France
Gael.de-Chalendar@cea.fr

Anne Vilnat - LIMSI/CNRS
bât. 508 - Université Paris XI
BP 133 - 91403 Orsay Cedex France
Anne.Vilnat@limsi.fr

Abstract

Word Sense Disambiguation is the task dedicated to the problem of finding out the sense of a word in context, from all of its many possible senses. Solving this problem requires to know the set of possible senses for a given word, which can be acquired from human knowledge, or from automatic discovery, called Word Sense Induction. In this article, we adapt two existing meta-methods of Word Sense Induction for the automatic construction of a disambiguation lexicon. Our adaptation is based on multiple semantic spaces (also called Word Space Models) produced from a syntactic analysis of a very large number of web pages. These adaptations and the results presented in this article differ from the original methods in that they use a combination of several high dimensional spaces instead of one single representation. Each of these competing semantic spaces takes part in a clustering phase in which they vote on sense induction.

Keywords

semantic space, word space model, dimensionality reduction, locality sensitive hashing, word sense induction, words clustering, multi-represented data

1 Introduction

A single word may potentially convey many different senses but despite this potential for ambiguity, misinterpretations occur relatively rarely in actual human linguistic interactions, including written texts. As a consequence, it is possible to draw the hypothesis that a wordsense should be inferable from the context it appears in (letting aside the question of the context size). Word Sense Disambiguation is the task dedicated to this problem of finding out the sense of a word in context. For a more detailed study, [1] and [12] have covered an exhausting overview of the task. However, solving this problem requires first to know the set of possible senses for a given word. The mere possibility of doing this exhaustively is a much debated issue, because even among humans, it is very difficult to agree on a set of senses for any given word. However if we consider it possible to obtain a list of the senses, even if it is incomplete, then it can be acquired from human knowledge (as one can find in any good dictionary), or from automatic discovery, a task which is generally called Word Sense Induction.

This task (and consequently Word Sense Disambiguation itself) is often tackled using methods that are able to group similar words together, in order to perform a clustering over the neighbors of the target, and thus to discover one 'sense' for each cluster found. This approach puts a large part of the burden into the similarity measure. While some methods use graph-based measures (thus making use of manually crafted structured resources), many unsupervised approaches are based on the Word Space Model paradigm, sometimes also called Semantic Vector Space paradigm. This paradigm relies on the hypothesis that a word-sense depends on the contexts it appears in [8]. For instance, all words that designate a *mammal* will tend to occur with verbs like *eat*, *run* or *breathe*. Because of this, if we represent each word by a normalized vector of the cooccurrence counts of this word with every other words in a given corpus, two words that share a lot of semantic features will tend to have a small angular distance.

Semantic spaces are interesting because they are built completely automatically from a corpus. They do not require error prone and costly manual work and can be quickly updated to reflect the emergence of new words in a constantly evolving domain . Semantic spaces can be built using various context types such as documents, paragraphs or occuring words, or again lemmas appearing near the source word (cooccurents). In [16], dimensions of Vector Space Model represent the terms occurrence frequencies in each document where each column represent a document. In [11], dimensions correspond to the most frequent terms of the vocabulary and the values are the number of cooccurrences, in fixed-size windows, between the line term and the column term (or possibly the mutual information). In [13] and [7], each dimension corresponds to a context obtained through a given syntactic

International Conference RANLP 2009 - Borovets, Bulgaria, pages 287–291

relation and the value is their cooccurrence frequency.

With these semantic spaces where each word is represented by a vector from which is derived a measure of similarity with other words, similar words can be grouped together Senses of polysemic words can be induced from various term selections. In [10], clusters are built in one step from the full vocabulary, each word may appear in several clusters. These clusters represent synonym classes and senses of words belonging to several clusters are thus discriminated. [17] approach consists in clustering all the occurence contexts in a corpus of each word for which senses are being learned. Finally, some systems like [4], [20] or [6] cluster cooccurrents of the words they want to distinguish the senses and [14] cluster the nearest neighbors in the semantic space.

Our work largely follows [14] as we also group nearest neighbors but differs from it as we dispose of a panel of different semantic spaces (built on different syntactic relations) with different specificities that we combine, instead of using a unique semantic space.

Section 2 of this paper details the conception of the semantic spaces we use. We present in Section 3 the method of dimensionality reduction we use. Section 4 describes the clustering methods developed to take into account differences between spaces. Finally, Section 5 gives perspectives on the results and some directions towards future works.

2 Description of the semantic space

Semantic spaces we use for word sense induction are issued from the work of [7] . The semantic spaces are built using the results of a large scale syntactic analysis. When we extracted them for our work, the French corpus on which this semantic space is built was made of two millions urls of French language pages on which a syntactic analysis (including lemmatization) was processed using LIMA, the CEA LIST natural language processing system [2].

This parser is a dependency analyser. Thus it extracts binary relations between tokens, like subject_verb, object_verb, noun_complement, etc. These relations are oriented: for example in the phrase *'advance in NLP'*, *NLP* appears in the context of *advance* for the noun_complement relation but *advance* appears in the context of *NLP* for the noun_complement reverse relation.

The dictionary used to build the matrices (i.e. the semantic space) is made of the 68,000 most frequent words of the French language. To each binary relation is associated a distinct matrix that registers the cooccurrence frequencies of these 68,000 words through the given relation. In order to be able to take into account the information given by all words, even the rare ones, we work with mutual information matrices computed from the frequencies matrices. Mutual information is computed according to the following formula, where P_i is the probability of occurrence of the term described by line i in any context of the given relation, P_j is the probability of occurence of the context defined by column j, and $P_{i,j}$ is the probability of cooccurrence of the term i with the context j in the given relation.

$$MI = log(\frac{P_{i,j}}{P_i * P_j}) \quad (1)$$

The same process is applied on reverse relations and on cooccurrences in fixed-size windows (5, 10, 20). Finally, we obtain 71 sparse square matrices of 68,000 dimensions. We will see later that these matrices contain different and complementary kind of information.

3 Approximative KNN

Given the size of these matrices, it is highly desirable to perform a dimensionality reduction before using them for nearest neighbor search and clustering. A linear Principal Component Analysis method such as Latent Semantic Analysis [9] would have been an alternative if not for the original dimensionality of our data and the quadratic ($O(n^3)$) complexity of the underlying Singular Vector Decomposition. On the other hand, Random Indexing [19] while more scalable, was not a real option because of its reported low quality. We chose to use Locality Sensitive Hashing which is supposed to be more scalable than LSA and whose quality was still to be tested on complex tasks. [3] has defined a family of LSH functions for which the hashed signatures keep their angular similarity for any input vectors pair. [15] has shown that this hashing is particularly well adapted to set up a fast nearest neighbors search method.

3.1 Dimension reduction: Locality sensitive hashing

The goal of a hashing is to obtain a footprint smaller than the original signature. Here, we want a function h giving a smaller footprint and respecting the property that if two vectors v_1 and v_2 from the initial space are similar, then the two hashed vectors $h(v_1)$ and $h(v_2)$ are also similar.The hashing functions family proposed by [3] allows to approximate the cosinus measure whose efficiency has already been shown for proximities of the elements in word spaces. We outline the method below.

We draw randomly according to a Gaussian distribution d unit vectors $\overrightarrow{r}$. This drawing provides an equidistributed breakdown on the unitary hypersphere.

Let a family of functions be defined by:

$$h_{\overrightarrow{r}}(\overrightarrow{u}) = \begin{cases} 0 \; if \; \overrightarrow{r}.\overrightarrow{u} \geq 0 \\ 1 \; if \; \overrightarrow{r}.\overrightarrow{u} < 0 \end{cases} \quad (2)$$

Let two vectors $\overrightarrow{u}$ and $\overrightarrow{v}$, the probability to draw a random vector defining a hyper plane that separates them is:

$$Pr[h_{\overrightarrow{r}}(\overrightarrow{u}) \neq h_{\overrightarrow{r}}(\overrightarrow{v})] = \theta(\overrightarrow{u}, \overrightarrow{v})/\pi \quad (3)$$

On a number of randomly drawn vectors this probability can be measured. Indeed, the probability that a randomly drawn hyperplane has separated the original two vectors u and v is the probability that the hyperplane has given a different bit for the two hash results for u and v. The formula 4 gives this probability:

$$Pr[h_{\overrightarrow{r}}(\overrightarrow{u}) \neq h_{\overrightarrow{r}}(\overrightarrow{v})] = hamming\_distance(\overrightarrow{u}, \overrightarrow{v})/d \quad (4)$$

By combining 3 and 4 the following approximation is obtained:

$$\theta(\overrightarrow{u}, \overrightarrow{v}) \approx hamming\_distance(\overrightarrow{u}, \overrightarrow{v})/d * \pi \quad (5)$$

3.2 Fast approximate nearest neighbors search

A fast approximate nearest neighbor search in a space with a Hamming distance was proposed by [3] and picked up by [15]. The method consists in pulling random p permutations of d elements. For each permutation, (a) the signatures are permuted bit by bit, (b) all the elements are sorted following lexicographic order, and (c) the B elements closest to the n source elements which cosine approximation is lower than a given threshold are kept.

We build the list of nearest neighbours words we want to cluster, using the main trends of that algorithm.

3.3 Results

We have been able to compute the list of k nearest neighbors of polysemous words for each of the syntactic spaces and we see for example in table 1 the results with $k = 10$ for the word *vol*[1] in various spaces.

We notice that the nearest neighbors obtained for the word *vol* are quite different depending on which space is used. Some spaces gather nearest neighbors oriented towards a precise meaning, while others return mixed meanings. uses can be separated into: *media type*, *computers aspect* and *content and uses*, the distinction between spaces remains unclear except for the *apposition* relation. We observed that these distinctions largely depend on the polysemous word processed: the discriminating spaces, when they exist, are not always the same. Therefore we can assume that each space contains different information which is worth being taken into account.

Our goal is to build sets of nearest neighbors representing different uses. Since automatic induction exploits the contexts of words occurrences in a corpus, various usages of even monosemous words can appear and be discriminated in the same way as different senses of polysemous words would. We thus prefer the term *use* to the term *sense*. Depending on the words, spaces discriminate quite heterogeneously their various *uses*. We propose in the next section a clustering method taking into account the specificity of each space while allowing the inter-space consolidation.

4 Word sense induction by multi-represented words clustering

In our approach, we want to cluster the nearest neighbors of a word into several clusters so that each of them represents a sense. We wish to be able to distinguish different clusters as in the manually built example of

[1] which means *fly* or *steal*

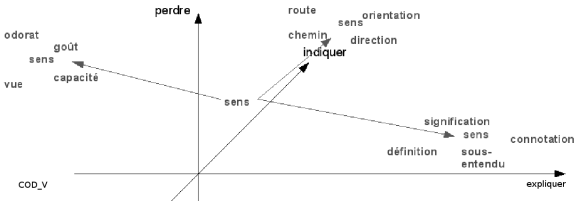

Fig. 1: *Senses discrimination in the object_verb base*

Figure 1. This figure shows a 3-dimensional projection where object_verb contexts should allow the discrimination of three meanings of the word *sens*: the *sens* as the ability to perceive, the *sens* giving an orientation indication, and the *sens* giving a semantic indication.

Traditional clustering methods are used on a single vector space, as illustrated in the ideal example above. But we have seen previously that the various spaces emphasized different closenesses. It can be seen for example in table 1) that the object_verb relation highlights the semantic proximity on *flying* meaning of the word *vol* while the *apposition* space does not highlight it at all, but stresses the proximities of *theft* meaning. We thus assume that the spaces peculiarities will help to better distinguish clusters of meaning if each of them is separately taken into account rather than considering a global space built by concatenation of all the matrices. To do this, we are guided by Shared Nearest Neighbors clustering algorithms adapted and used by [5] and [6] and by the Hyperlex algorithm developed by [20]. We propose two "clustering by vote" methods.

For each of the parts of speech for which we want to produce senses, *i.e. noun, verb, adjective*, we retain the relevant spaces . For instance when dealing with a verb, we do not consider the complement_noun space involving only nouns and we use the reverse object_verb space and not the object_verb one. We also consider only spaces involving relations between plain words (noun, verb, adverb, adjective). For example, we leave out the relationship between a determiner and its noun.

After experimentation, we choose to cluster the words that appear at least twice in the 30 nearest neighbors of the selected spaces. Let E be this set. The following sections present the two methods.

4.1 Method based on the shared nearest neighbors algorithm

This method processed in three steps. First, a number of seeds (not fixed in advance) is extracted. The remaining elements are then assigned to these clusters. Finally clusters whose seeds are too close are joined and elements belonging to clusters considered as too small are reassigned.

The algorithm we propose is based on the [5] SNN algorithm seed selection method for which there is no need to choose *a priori* the number of seeds. However, experiments give us better results with a direct neighbors graph than with the shared nearest neighbors graph. We then keep the first one. One possible explanation for this surprising result compared

vol	complement_noun	vol, avion, voyage, retour, course, achat, opération, première, journée, premier
	object_verb	vol, voyage, retour, création, attaque, changement, mise, mariage, acte, fin
	apposition	vol, meurtre, racket, fraude, chantage, assassinat, homicide, rapine, violence, crime
	apposition.reverse	vol, meurtre, viol, prostitution, rapine, adultère, proxénétisme, idolâtrie, agression, luxure

Table 1: *10 nearest neighbors of words vol*

to the results presented in [5] is that we use a small part of the original space elements. There is therefore too few data to use this second level of information. We also retain the idea of defining the seeds using the *strong links*. The name Shared Nearest Neighbors being no longer relevant, we now refer to this algorithm as MultiNN.

In each space (object_verb, subject_verb, ...):

1. we build the nearest neighbors graph in which each element of E has a corresponding node and every distance between two of the elements is represented by an edge weighted by their approximate cosine distance.

2. we define a strong link threshold (eg. 0.2 of the maximum distance in the space) and discard edges whose values are below this threshold;

3. for each node in the graph (elements of E) we compute the sum of its remaining links.

4. if this sum is larger than a certain threshold (eg. 0.3 of the maximum in the space), we say that this is a *local seed* for this space.

For each element of E, we know the number of spaces in which it is a *local seed*. If this number is bigger than a threshold (eg 0.8 of the number of collaborating spaces), then this element is said to be a *global seed*.

We build clusters around each global seed in this way:

- In each space, we remove the elements of E which have been called *global seed*.
- In each space and for each element of E, we store a vote for its nearest global seed (highest approximate cosine).
- We sum the votes on all spaces and assign each element to its most popular seed (possibly several ones in case of equality).
- In each space, if two seeds have a total value exceeding a certain threshold, the space votes for the merging of the two clusters involved.
- If a sufficient number of spaces votes for this merging, the two clusters are merged into one.
- The elements of the clusters considered too small compared to the number of elements to be clustered are reassigned one by one to large clusters.

4.2 Method based on the HyperLex algorithm

We adapt a second method to our multi-representation of words. This is the Hyperlex algorithm presented by [20]. It was originally applied to a list of co-occurrences but in our case we apply it to a list of nearest neighbors.

The the first part of the original algorithm proceeds as follows.
Let E be the set of words to cluster.

- Take the most frequent element (hub) in E. If the number of its connected nodes whose weight is below a certain threshold φ is greater than a given number k, this hub becomes the seed of a component.
- Remove this hub and if it has been determined to be a seed, remove also its connected nodes from E.
- Repeat until E is empty.

In the scope of our multiple representations, we define the distance from any element of E to the source as the average distance over the spaces and use the decreasing distances as the order in which to test whether a node is a seed. We assume that a word close to the source will be a good sense representation. We do not need to test the high frequency term in first place because our graph is a nearest neighbours graph in which a high frequency of a term does not infer more outgoing edges. Instead of being discarded, items related to a seed are candidates to become elements of the seed component. The different spaces vote to validate this attribution. In this first study we do not implement the second part of the algorithm (spanning tree calculation) as we assume that the process of voting will filter out the words which are not part of a component and let them be attached to a forthcoming seed.

4.3 Results

The results were evaluated manually in this first step by comparing our results with those obtained by the original algorithms for a predefined list of words. For instance, table 2 presents the results of the four methods (the two original and our two modified versions) for the word *barrage*.

The comparison of our clusters with those of the original algorithms is difficult because we did not try to group the same type of terms (cooccurrents vs. syntactic nearest neighbors). We can notice that the distinguished senses are not always the same. We find firstly in 3.1, 4.1 and 4.4, the usage of *barrage* as *hydraulic dam* which was discriminated for its use as an *industrial system* and in 3.2 and 4.2 as a *building on a river*. The senses 3.3 and 4.3 of *barrage* correspond to the meanings *roadblock, police barrage or factory strike* that we cannot well distinguish. The dictionary *Petit Larousse* used during the ROMANSEVAL campaign [18] does not itself distinguish them and simply considers them as *obstacles*. This is the same physical object, but the usage is different. Finally, we could not find the sportive meaning of *play-off match*, whose use is present in very few locutions such as *match de barrage*. The fact that hardly no nearest neighbour can express this meaning and the sparseness of its occurrences can both explain the difficulty we have to extract it. Nevertheless, for the word *vol*, our algorithm correctly extracted the *theft offence* and *flight* meanings, which was not the case for the HyperLex algorithm.

As Word Sense Induction systems group different initial terms and learn on different corpus bearing potentially different semantic content, an automatic evaluation of our

Source word	barrage
HyperLex [20]	1.1 : eau, construction, ouvrage, rivière, projet, retenue, crue
	1.2 : routier, véhicule, camion, membre, conducteur, policier, groupement
	1.3 : frontière, Algérie, militaire, efficacité, armée, Suisse, poste
	1.4 : match, vainqueur, victoire, rencontre, qualification, tir, football
SNN [6]	2.1 : manifestant, forces_de_l'ordre, préfecture, agriculteur, protester, incendier, calme, pierre
	2.2 : conducteur, routier, véhicule, poids_lourd, camion, permis, trafic, bloquer, voiture, autoroute
	2.3 : fleuve, lac, rivière, bassin, mètre_cube, crue, amont, pollution, affluent, saumon, poisson
	2.4 : blessé, casque_bleu, soldat, milicien, tir, milice, convoi, évacuer, croate, milicien, combattant
MultiHyperLex	3.1 : infrastructure, défense, établissement, installation, aménagement
	3.2 : rivière, digue, pont, canal, lac
	3.3 : train, station, usine, bâtiment, route, véhicule
MultiNN	4.1 : bâtiment, usine, véhicule, aménagement, bassin, chantier, infrastructure
	4.2 : pont, barrière, digue, écluse
	4.3 : route, rivière, train, défense
	4.4 : station, canal, centrale, établissement, infrastructure, installation

Table 2: *Comparison of clusters built for the word* barrage

results by comparing the resulting clusters with those produced by other methods (whatever the mapping method used) can prove to be inaccurate. A more reliable evaluation would be to perform a more applicative task based on those clusters (like Word Sense Disambiguation or Information Retrieval) and to evaluate the results of the latter. An automatic evaluation of this kind is still to be performed in order to judge which of the two proposed approaches give the better results and to assess the cluster discrimination quality.

5 Discussion and future work

Although we do not have to choose the number of clusters to obtain, allowing a different number of meanings for each word, fixing the values of the parameters involved in the two algorithms remains a real problem in the sense that we currently have no way to learn the optimal parameters. We are thus forced to choose them by experimentation.

An advantage of these methods can be put forward: in addition to distinguishing clusters of meaning, we assume that the use of nearest neighbors as a set of elements to cluster (not using cooccurrents) will allow the use of those clustered neighbors as learning data for a classifier performing word sense disambiguation. This hypothesis will be tested soon.

Furthermore, the finalization of this work requires to quickly carry out various studies. To highlight the contributions of the method presented here, we want to build clusters from the same set of closest neighbors, using various spaces configuration: first the window cooccurrences space only, second each of the syntactic spaces alone, and third the concatenation of all matrices, which should differ highly from our combination.

Finally, we plan to automatically build clusters of those sense-clusters to define WordNet-like *synsets*. We will use them to disambiguate and index a collection of documents and study the contribution of this disambiguation to information retrieval.

References

[1] E. Agirre and P. Edmonds, editors. *Word Sense Disambiguation: Algorithms and Applications (Text, Speech and Language Technology)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[2] R. Besançon and G. de Chalendar. L'analyseur syntaxique de lima dans la campagne d'valuation easy. In M. Jardino, editor, *Actes de TALN 2005 (Traitement automatique des langues naturelles)*, Dourdan, June 2005. ATALA, LIMSI.

[3] M. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, 2002.

[4] B. Dorow and D. Widdows. Discovering corpus-specific word senses. In *EACL 2003*, 2003.

[5] L. Ertz, M. Steinbach, and V. Kumar. Finding topics in collections of documents: A shared nearest neighbor approach. In *Text Mine 01, Workshop of the 1st SIAM International Conference on Data Mining*, 2001.

[6] O. Ferret. Discovering word senses from a network of lexical cooccurrences. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 1326, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

[7] G. Grefenstette. Conquering language : Using nlp on a massive scale to build high dimensional language models from the web. In *Proc. of the 8th CICLing Conference*, pages 35–49, Mexico, 2007.

[8] Z. Harris. Distributional structure. In J. J. Katz, editor, *The Philosophy of Linguistics*, pages 26–47. Oxford University Press, New York, 1985.

[9] T. K. Landauer and S. Dumais. A solution to plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psychlogical Review*, 104(2):211–240, 1997.

[10] D. Lin. Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL98*, pages 768–774, 1998.

[11] K. Lund and C. Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, and Computers*, 28:203–208, 1996.

[12] R. Navigli. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2):1–69, 2009.

[13] S. Padó and M. Lapata. Dependency-based construction of semantic space models. *Comput. Linguist.*, 33(2):161–199, 2007.

[14] P. Pantel and D. Lin. Discovering word senses from text. In *Proc. of ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2002*, Edmonton, Canada, 2002.

[15] D. Ravichandran, P. Pantel, and E. Hovy. Randomized algorithms and nlp: Using locality sensitive hash functions for high speed noun clustering. In *Proceedings of ACL*, Ann Arbour(MI), 2005.

[16] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. 18(11):613–620, 1975.

[17] H. Schütze. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123, 1998.

[18] F. Segond. Framework and results for french. *Computers and the Humanities, Special Issue on SENSEVAL*, 34(1-2), 2000.

[19] L. Sellberg and A. Jönsson. Using random indexing to improve singular value decomposition for latent semantic analysis. In *Proceedings of the 6th Conference on Language Resources and Evaluation*, Marrakech, Morocco, 2008.

[20] J. Véronis. Hyperlex: lexical cartography for information retrieval. *Computer Speech and Language*, 18:223–252, 2004.