Synonym Dictionary Improvement through Markov Clustering and Clustering Stability[⋆]

David Gfeller, Jean-Cédric Chappelier, and Paolo De Los Rios

Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland

Abstract. The aim of the work presented here is to clean up a dictionary of synonyms which appeared to be ambiguous, incomplete and inconsistent. The key idea is to use Markov Clustering and Clustering Stability techniques on the network that represents the synonymy relation contained in the dictionary. Each densely connected cluster is considered to correspond to a specific concept, and ambiguous words are identified by the evaluation of the stability of the clustering under noise. This allows to disambiguate polysemic words, introducing missing senses where required, and merge similar senses of a same word if necessary.

Keywords: Complex Network, Markov Clustering Algorithm, Clustering Stability, Community Structure, Synonymy, Word Sense Disambiguation.

Introduction

One aspect of the complexity of text mining comes from the synonymy and the polysemy of the words. The aim of Word Sense Disambiguation (WSD) is precisely to associate a specific sense to every word within context [Besançon *et al.*, 2001, Schütze, 1998]. The determination of the list of possible senses for a given word is a key aspect in this disambiguation. A possible starting point could be a dictionary of synonyms. It happens however that, due to both inherent human errors and errors coming from the automatic (semi-supervised) construction of the dictionary, the synonymy network can contain several mistakes and turn out to be ambiguous, incomplete or inconsistent.

This paper starts with a brief description of the synonymy network derived from the dictionary we used. Then, the clustering algorithm applied to improve it is presented and the general method to identify ambiguities in the clustering is introduced. Finally, the results obtained are discussed.

1 The Synonymy Network

The synonym network we here consider has been built from a French dictionary of synonyms. The synonymy relation is defined between the words in one of their senses and is considered to be symmetric. The resulting network

[⋆] This work was financially supported by the EU commission by contracts COSIN (FET Open IST 2001-33555) and DELIS (FET Open 001907).

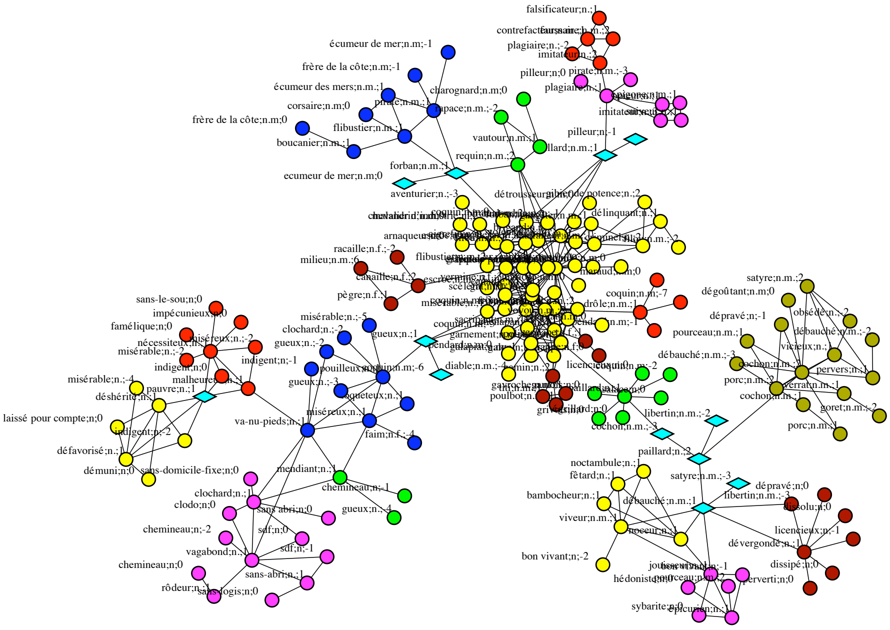

Fig. 1. MCL clustering of the component with 185 elements (different level of grey represent different clusters). Unstable nodes (explained in section 3) are represented with diamonds.

is thus undirected (and unweighted). It is not fully connected but consists of many disconnected components[1], with a power-law size distribution (see fig. 2); the nodes inside a single component representing the same "concept"[2].

The first encountered problem in this dictionary was the existence of much too large synonym components (up to almost 10,000 nodes, see fig. 2). The transitive closure of the synonymy relation connects words which have different senses; e.g. *fêtard* ("merrymaker") and *sans-abri* ("homeless") (see fig. 1). This is due to words which are still ambiguous[3] and relate different "concepts": even if a path exists between two nodes, the slight changes in the senses that could occur at each step along this path may result in a quite different sense between both ends. Moreover these big components clearly show a sub-structure suggesting a partition into smaller clusters.

The second encountered problem was that some words were given too many senses, i.e. senses that actually correspond to the same "concept".

To solve these problems, the Markov clustering algorithm (MCL) [Van Dongen, 2000] was first applied to the synonymy network. The idea is that words with tighter neighborhoods are likely to be less ambiguous than words with

[1] i.e. groups of words which are claimed to be synonyms
[2] We use the word "*concept*" to denote a group of senses that are synonyms.
[3] i.e. the distinction between two of their senses has not been introduced

fuzzier neighborhoods [Sproat and van Santen, 1998], and thus a cluster is likely to correspond to words with a close meaning, and therefore represents one "concept". Then, in order to find ambiguous words, we noise the edges and compare the clusters obtained for several noisy realizations of the network. This provides informations about the network that could not have been extracted with the standard clustering algorithms.

2 Markov Clustering

In this section, MCL, the clustering algorithm we used for splitting the big components into smaller clusters, is briefly described.

Its basis is that "*a random walk on a network that visits a dense cluster will likely not leave it until many of its vertices have been visited*" [Van Dongen, 2000]. The idea is to favor the most probable random walks, increasing the probability of staying in the initial cluster. The algorithm works as follows: (1) consider the adjacency matrix of the network[4] ; (2) normalize each column to one, in order to obtain a stochastic matrix S; (3) square the matrix S; the element $(S^2)_{ij}$ is the probability that a random walk starting at node j ends up at node i after two steps; (4) take the r^{th} power of every element of S^2 (typically $r \approx 1.5 - 2$); this favors the most probable random walks; (5) go back to 2 until convergence.

After several iterations we end up with a matrix stable under MCL. Only a few lines of the matrix have non-zero entries, which give the cluster structure of the network. Note that the parameter r can tune the granularity of the clustering: a small r corresponds to a few big clusters, whereas a big r corresponds to smaller clusters. Comparing the results with different r for some of the components, we chose $r = 1.6$ as a reasonable value.

As an example, the result of MCL on the component of size 185 is displayed in fig. 1. The obtained subdivision into smaller clusters is definitely more meaningful; e.g. *fêtard* is no longer in the same cluster as *sans-abri*.

MCL was applied on the biggest components of the network. We noticed that, as the size of the components becomes smaller than 40, the clustering is often not meaningful anymore, since the components do not show any particular community structure. After clustering the biggest components, the cluster size distribution shown in fig. 2 is obtained. The power-law is still conserved, but the size of the biggest components is much reduced.

3 Unstable Nodes

MCL partitions the network into clusters without overlap, i.e. every node is assigned to a single cluster ("hard-clustering"). However, the resulting

[4] For an undirected and unweighted network, this matrix is symmetric and composed only of zeros and ones.

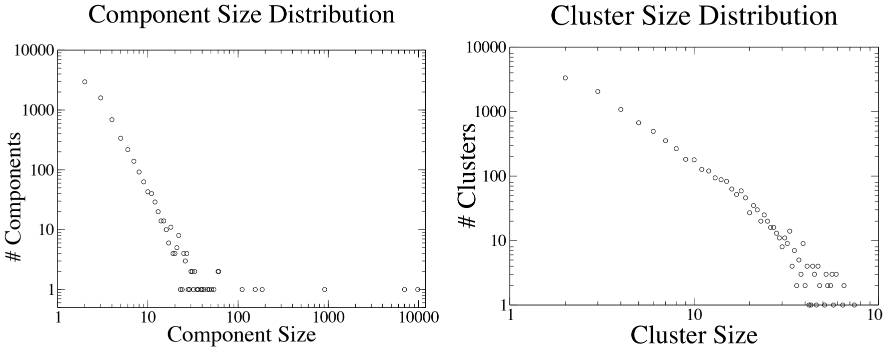

Fig. 2. Left: Distribution of the size of the components in the whole network (log-log scale). Right: Distribution of the size of the clusters after MCL with $r = 1.6$.

clustering is sometimes questionable – both from a topological and a linguistic point of view –, especially for nodes that "lie on the border" between two clusters (see fig. 3).

The problem of finding ambiguities is closely related to the evaluation of the robustness of the clustering. Some attempts, based on particular clustering algorithms, were drawn recently to solve this problem [Wilkinson and Huberman, 2004, Reichardt and Bornholdt, 2004]. We present here a new method based on the introduction of a stochastic noise over the edges of the network, and apply it in the framework of MCL. This consists in adding noise over the non-zero entries of the adjacency matrix[5]. Running MCL with noise several times, some nodes are switching from one cluster to the other (for example node "*reprendre;6*" in fig. 3). This procedure is now detailed.

Let p_{ij} be the probability for the edge between node i and node j of being inside a cluster. After several runs of the clustering algorithm with the noise, a weighted network is obtained where edges with probability 1 are always within a cluster and edges with probability close to 0 connect two different clusters. Edges with a probability smaller than a threshold θ are thus considered as "*external edges*". By removing those edges, one gets a disconnected network[6].

[5] In this study, the noise added over the edges weights (originally equal to 1) is equally distributed in $[-\sigma, \sigma]$, $0 < \sigma < 1$. With σ close to 0, unstable nodes are not detected, while with $\sigma \simeq 1$ the topology of the network changes dramatically. The results were stable for a broad range of values of σ around 0.5. For example in the component displayed in fig. 3, the node "*reprendre;6*" was identified as the only unstable node for $0.35 \leq \sigma \leq 0.8$.

[6] For the choice of the parameter θ, we looked at the distribution of the probabilities p_{ij} over the whole network. As expected, this distribution has a clear maximum in 1, corresponding to edges that are never cut by MCL, preceded by a region corresponding to edges almost never cut. Since for $p_{ij} \leq 0.8$ the distribution is almost flat, we choose $\theta = 0.8$.

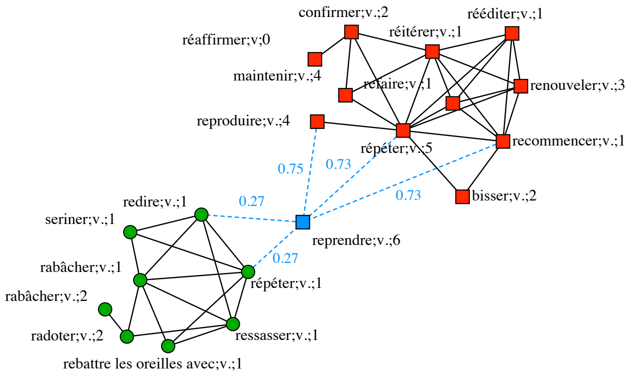

Fig. 3. Small sub-network with one unstable node (“*reprendre;6*”), extracted from a component of 111 nodes. The values over the dashed edges are the probabilities for the edges to be inside a cluster (average over 100 realizations of the clustering with $r = 1.6$, and $\sigma = 0.5$.). Only probabilities smaller than $\theta = 0.8$ are shown. The shape of the nodes indicates the cluster found without noise.

In this section, we use the word “*cluster*” for the clusters obtained without noise, and “*subcomponent*” for the disconnected parts of the network after the removal of the external edges. If the community structure of the network is stable under several repetitions of the clustering with noise, the subcomponents of the disconnected network correspond to the clusters obtained without noise. In the opposite case, a new community structure appears with some similarity with the initial one.

In order to identify which subcomponents correspond to the initial clusters and which are new subcomponents, we introduce the notion of similarity between two sets of nodes. If E_1 (resp. E_2) is the set of clusters (resp. the set of subcomponents), we use the Jaccard index to define the similarity s_{ij} between cluster $C_{1j} \in E_1$ and subcomponent $C_{2i} \in E_2$: $s_{ij} = \frac{|C_{2i} \cap C_{1j}|}{|C_{2i} \cup C_{1j}|}$.

If $C_{1j} = C_{2i}$, $s_{ij} = 1$ and if $C_{1j} \cap C_{2i} = \emptyset$, $s_{ij} = 0$. For every $C_{1j} \in E_1$, we find the component C_{2i} with the maximal similarity and identify it with the cluster C_{1j} (C_{2i} often corresponds to the stable core of the cluster C_{1j}). If there is more than one of such components, none of them is identified with the cluster. In practice, this latter case is extremely rare.

Nodes belonging to subcomponents that have never been identified with any cluster could be defined as *unstable* nodes. However, this definition suffers some drawbacks since it sometimes happens that a big cluster splits into two subcomponents of comparable size. Considering that almost half of the nodes of the cluster are unstable is not realistic and a new cluster should be defined instead. In practice, subcomponents of four nodes or more often correspond to a cluster not detected by the algorithm. We therefore define the unstable nodes as the nodes belonging to subcomponents that have not been identified with a cluster and whose size is smaller than 4.

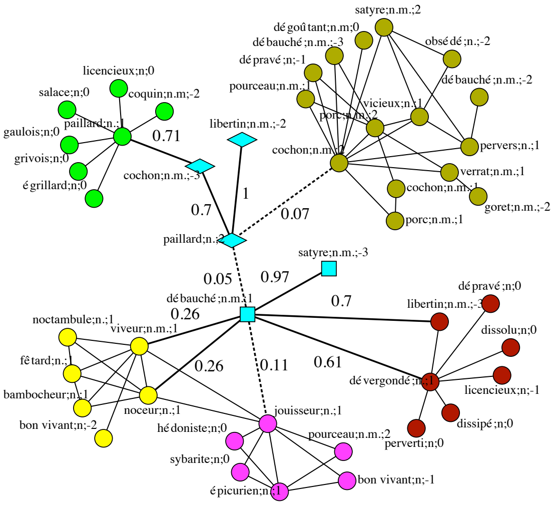

Fig. 4. Zoom over the bottom-right of fig. 1. Five unstable nodes have been found (non-circle nodes). The splitting of them proceeds as follows: removing the edges with $p_{ij} < 1-\theta$ (dashed-line), they are divided into two groups ({*cochon;-3 libertin;-2, paillard;2*} (diamonds) and {*débauché;1, satyre;-3*} (squares). The first group has only one adjacent subcomponent. It is therefore merged to this subcomponent. The second group has two adjacent subcomponents. It is thus duplicated and merged into those two subcomponents.

In the framework of a synonymy network, unstable nodes, which lie on the border of two subcomponents, correspond to polysemic words which have not been clearly identified as such (i.e. one of their senses is not present in the dictionary). We thus decided to split these nodes among their adjacent subcomponents. The adjacent subcomponents are defined as the subcomponents to which the node is connected through at least one edge with a probability higher than a given threshold θ'. Typically we choose $\theta' = 1 - \theta$, where θ was the threshold for defining an edge as external.

If several unstable nodes are connected together, we split them according to the following procedure: first group these nodes keeping only the edges with $p_{ij} > \theta'$; then, for each group, duplicate it and join it to its adjacent subcomponents (see fig. 4).

Finally, the second problem, where the same word appears with different senses in a cluster (e.g. *rabâcher* in fig. 3) has been addressed by simply merging into a single node the nodes that correspond to the same word in a same subcomponent. Indeed, if a node appears twice in a subcomponent, both senses are actually not different, at least not at the level of granularity used. Such a situation occurs 4,642 times in the whole network (the total number of nodes is 50,913). This number is more than four times smaller than before the MCL clustering (21,261 "duplicates").

4 Discussion and Conclusion

The objective evaluation of the work presented here is not easy. How to evaluate whether better results are obtained when no reference to compare to is available[7]? One possible evaluation could be to compare the performances obtained in a targeted WSD experiment using the original and the corrected resource. It is however highly dependent on the targeted application. We thus rather choose to evaluate the method presented here by a subjective (i.e. human centered) validation, achieved by sampling components in the newly obtained network. This evaluation appeared to be very convincing[8].

However, we still wanted to develop objective overall clues from the network to try to objectively grasp the benefits of the method. We, for instance, computed the clustering coefficient of the unstable nodes. The clustering coefficient (C) of a node is the number (N_3) of 3-loops passing through the node, divided by the the maximal possible number of such 3-loops. When the node has degree k, $C = \frac{2\,N_3}{k(k-1)}$. If a node lies in the middle of a densely connected cluster, it quite likely has a high clustering coefficient. If the node lies between clusters, it has a small clustering coefficient[9].

Using MCL, the assumption is made that a community structure is present in the network. Since MCL may also give a partition of random networks without any community structure, it is important to validate this assumption. The introduction of the probability p_{ij} over the edges provides a way to do so. In the case of a random network, the community structure found by the clustering algorithm is expected to be very sensitive to the noise, whereas in the case of a network with a clear community structure, the clusters are quite stable, except for a few unstable nodes. To characterize these two situations, we introduce the *clustering entropy* S_c as a measure of the clustering stability:

$$S_c = -\frac{1}{M} \sum_{(i,j)} \Big(p_{ij} \log_2 p_{ij} + (1 - p_{ij}) \log_2 (1 - p_{ij}) \Big),$$

where M is the total number of edges (i, j) in the network.

Important differences in the clustering entropy between networks with a clear community structure and random networks with no community structure are expected. If the network is totally unstable (i.e. $p_{ij} = \frac{1}{2}$ for all edges), $S_c = 1$, while if the edges are perfectly stable ($p_{ij} = 0$ or 1), $S_c = 0$.

To avoid biasing the results, we compare the components with a randomized version of the same component in which the degree of each node is conserved [Maslov and Sneppen, 2002]. Table 1 shows the comparison for several big components of the network of synonyms. The clustering entropy

[7] had we had one, wouldn't have we developed a method to correct it!

[8] See fig. 1, 2, 4 and 3 for illustrations.

[9] For example, twelve unstable nodes were found in the component displayed in fig. 1. The average clustering coefficient of these nodes is 0.08, while the average clustering coefficient over the whole component is 0.42.

Component Size	S_c(original)	S_c(random)
912	0.25±0.01	0.55±0.01
185	0.19±0.01	0.62±0.02
155	0.27±0.01	0.55±0.03
111	0.21±0.01	0.69±0.02
61	0.20±0.01	0.68±0.04
60	0.19±0.01	0.76±0.04
54	0.21±0.01	0.60±0.07
51	0.21±0.01	0.69±0.05
average	0.21	0.64

Table 1. Comparison for several components. S_c(original) is the clustering entropy of the original components. S_c(random) is the average clustering entropy for 50 randomized versions of the component. We used $r = 1.6$ and $\sigma = 0.5$.

of the randomly rewired components is at least twice bigger than the entropy of the original components. This experimentally shows that the clusters obtained with MCL are not an artifact of the method, but correspond to a real community structure in the network.

Applying the MCL clustering algorithm, the network of synonyms splits into sensible clusters, significantly improving, at least subjectively, the quality of the dictionary. Most of the clusters can be interpreted as groups of synonyms and, at a coarse-grained level of representation, correspond to a general concept of the language. The method introduced to identify nodes which lie between clusters and to check the robustness of the clustering appeared to be fruitful in the splitting of polysemic nodes. We emphasize that this method does not depend of a particular clustering algorithm and can be applied on any complex network.

References

[Besançon *et al.*, 2001]R. Besançon, J.-C. Chappelier, M. Rajman, and A. Rozenknop. Improving text representations through probabilistic integration of synonymy relations. In *Proc. of ASMDA'2001*, pages 200–205, 2001.

[Maslov and Sneppen, 2002]S. Maslov and K. Sneppen. Specificity and stability in topology of protein networks. *Science*, 296:910–913, 2002.

[Reichardt and Bornholdt, 2004]J. Reichardt and S. Bornholdt. Detecting fuzzy community structures in complex networks with a potts model. *Phys. Rev. Lett.*, 93(218701), 2004.

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

[Sproat and van Santen, 1998]R. Sproat and J. van Santen. Automatic ambiguity detection. In *Proc. of ICSLP'98*, 1998.

[Van Dongen, 2000]S. Van Dongen. *Graph clustering by flow simulation.* PhD thesis, University of Utrecht, 2000.

[Wilkinson and Huberman, 2004]D.M. Wilkinson and B.A Huberman. A method for finding communities of related genes. *PNAS*, 101:5241–5248, 2004.