A Soft-Clustering Algorithm for Automatic Induction of Semantic Classes

Elias Iosif and Alexandros Potamianos

Dept. of Electronics and Computer Engineering, Technical University of Crete, Chania, Greece
iosife@telecom.tuc.gr, potam@telecom.tuc.gr

Abstract

In this paper, we propose a soft-decision, unsupervised clustering algorithm that generates semantic classes automatically using the probability of class membership for each word, rather than deterministically assigning a word to a semantic class. Semantic classes are induced using an unsupervised, automatic procedure that uses a context-based similarity distance to measure semantic similarity between words. The proposed soft-decision algorithm is compared with various "hard" clustering algorithms, e.g., [1], and it is shown to improve semantic class induction performance in terms of both precision and recall for a travel reservation corpus. It is also shown that additional performance improvement is achieved by combining (auto-induced) semantic with lexical information to derive the semantic similarity distance.

Index Terms: semantic classes, unsupervised clustering

1. Introduction

Many applications dealing with textual information require classification of words into semantic classes including language modeling, spoken dialogue systems, speech understanding and information retrieval [2]. Manual construction of semantic classes is a time consuming task and often requires expert knowledge. In [3], lexico-syntactic patterns are used for hyponyms acquisition. A semi-automatic approach is used by [4] in order to cluster words according to a similarity metric. In [5], an automatic procedure is described that classifies words and concepts into semantic classes, according to the similarity of their lexical environment. More recently, in [1], a combination of multiple metrics is proposed for various application domains.

All of the above iterative approaches assign deterministically a word into a particular induced semantic class. In this paper, we propose an iterative soft-decision, unsupervised clustering algorithm, which instead of deterministically assigning a word to a semantic class computes the probability of class membership in order to generate semantic classes. The proposed soft clustering algorithm is compared to the hard clustering algorithm, used in [4, 5, 1]. Various other "hard" clustering algorithms are also evaluated that use lexical-only, or lexical and (auto-induced) semantic information for deriving class estimates. It is shown that: (i) the proposed soft-clustering algorithm outperforms all hard-clustering algorithms and (ii) best results are obtained when both lexical and semantic information are used for classification.

2. Hard Clustering Algorithm

We follow a fully unsupervised, iterative procedure for automatic induction of semantic classes, consisting of a *class generator* and a *corpus parser* [5]. The *class generator*, explores the immediate context of every word (or concept) calculating the similarity between pairs of words (or concepts) using KL metric, The most semantically similar words (or concepts) are grouped together generating a set of semantic classes. The *corpus parser*, re-parses the corpus and substitutes the members of each induced class with with an artificial class label. These two components are run sequentially and iteratively over the corpus (the process is similar to the one shown in Fig. 2 but with a hard decision in step II).

2.1. Class Generator

Our approach relies on the idea that the similarity of context implies similarity of meaning [6]. We assume that words, which are similar in contextual distribution, have a close semantic relation [4, 5]. A word w is considered with its neighboring words in the left and right contexts within a sequence: $w_1^L \ w \ w_1^R$. In order to calculate the distance of two words, w_x and w_y, we use a relative entropy measure, *Kullback-Leibler* (KL) distance, applied to their conditional probability distributions [4, 5, 7]. For example, the left KL distance is

$$KL^L(w_x, w_y) = \sum_{i=1}^{N} p(w_i^L|w_x) \log \frac{p(w_i^L|w_x)}{p(w_i^L|w_y)} \qquad (1)$$

where $V = (w_1, w_2, ..., w_N)$ is the vocabulary set. The similarity of w_x and w_y is estimated as the sum of the left and right context-dependent symmetric KL distances:

$$\begin{aligned} KL(w_x, w_y) = KL^L(w_x, w_y) + KL^L(w_y, w_x) + \\ + KL^R(w_x, w_y) + KL^R(w_y, w_x) \end{aligned} \qquad (2)$$

If w_x and w_y are lexically equivalent, then $KL(w_x, w_y) = 0$.

Using distance metric KL the system outputs a ranked list of pairs, from semantically similar to semantically dissimilar. In [5], a new class label is created for each pair and the two members are assigned to the new class. However, there is no way to merge more than two words (or concepts) at one step which may lead to a large number of hierarchically nested classes. In [4], multiple pair merges are used. However, the number of pair merges is predefined and remains constant for all system iterations. These simple grouping algorithms were extended in [1] by allowing varying number of pair merges.

For example, assume that the pairs (A,B), (A,C), (B,D) were ranked at the top three list positions. According to the proposed algorithm, the class (A,B,C,D) will be created. To avoid over-generalizations only pairs that are rank ordered close to each other are allowed to participate in this process. The parameter "Search Margin", SM, defines the maximum distance between two pairs (in the semantic distance rank ordered list) that are allowed to be merged in a single class. Consider the following pairs (A,B), (B,C), (E,F), (F,G), (C,D) rank-ordered

from one to five, where A, B, C, D, E, F, G represent candidate words or classes. For $SM = 2$ the classes (A,B,C) and (E,F,G) will be generated, while for $SM = 3$ the classes (A,B,C,D) and (E,F,G) will be generated. By adding the search margin SM constraint it was observed that performance improved [1].

2.2. Corpus Parser

The corpus is re-parsed after the class generation. All instances of each of the induced classes are replaced by a class label. Suppose that the words "Noon" and "LA" are categorized to the classes <time> and <city>, respectively.[1] The sentence fragment "Noon flights to LA" becomes "<time> flights to <city>". After the corpus re-parsing, the algorithm continues to the next system iteration. So, the lexical type of the clustered words is substituted by semantics and it is not present anymore in the corpus during the next iterations.

3. Soft Clustering Algorithm

The previously described hard clustering algorithm suffers from some drawbacks. First, a word is deterministically assigned to only one induced class. This isolates the word from additional candidate semantic classes. Furthermore, if the word categorization is false, then the erroneous induced class is propagated to the next class generation via the next iterations, leading to cumulative errors. Also, as the corpus is re-parsed, lexical information is being eliminated and substituted by the imported auto-induced semantic tags. This it is likely to produce fallacious semantic over-generalizations [5].

We propose a fully unsupervised, iterative *soft clustering algorithm* for automatic induction of semantic classes. The proposed algorithm follows a similar procedure as the hard clustering algorithm but alleviates the aforementioned disadvantages. A word is soft-clustered to more than one induced class according to a probabilistic scheme of membership computation thus reducing the impact of classification errors. In addition, the lexical nature of the corpus is preserved by equally weighting lexical and derived semantic information in the distance computation metric. Thus the soft clustering algorithm combines both lexical and induced semantic information as explained next.

3.1. Soft Class N-gram Language Model

Recall the example of Section 2.2 where the words "Noon" and "LA" are categorized to the classes <time> and <city>, respectively. The key idea of the proposed soft clustering algorithm allows words to belong to more than one induced semantic classes. In Figure 1 each word that belongs to multiple classes is represented by multiple triplets $(w_i, c_j, p(c_j|w_i))$, where w_i is the word itself, c_j is the label of an induced semantic class (concept) and $p(c_j|w_i)$ is the *probability of class membership*, that is the probability of word w_i being member of class c_j, as defined in Section 3.2.2. This "soft" class assignment shown in Fig. 1 is represented as (Noon,<time>,0.45), (Noon,<city>,0.05) and (LA,<time>,0.075), (LA,<city>,0.425). Note that it is not required that all words are assigned to classes; in Section 3.2.2, the multi-class soft-assignment criterion is discussed. Additionally, for all corpus words we retain the lexical form; for each word w_i there is an (additional) triplet $(w_i, w_i, p(w_i|w_i))$ with fixed probability $p(w_i|w_i) \equiv 0.5$, e.g., (Noon,Noon,0.5), (to,to,0.5). By design the probability mass is equally split between the lexical and semantic information, i.e., for each word the sum of class membership probabilities over all classes is equal to 0.5 and equal to the probability of the word retaining its lexical form.[2]

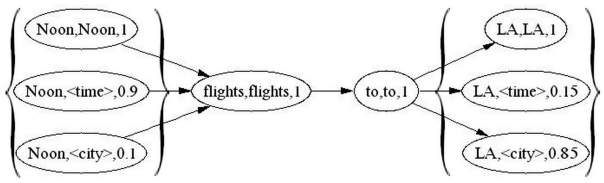

Figure 1: Sentence fragment with multiple semantic representations after the 1^{st} iteration of class induction.

A number of semantic classes are generated at every system iteration. We define the set S^n of induced classes generated up to the n^{th} iteration, the corpus vocabulary set V containing all words, and their union $C^n = S^n \bigcup V$. Using the above definitions we propose an n-gram language model for the class labels and words, elements of set C^n. The maximum likelihood (ML) unigram probability estimate for c_j is

$$\Big(\hat{p}(c_j)\Big)_{ML} = \frac{\sum_{\forall w_i \in V} p(c_j|w_i)}{\sum_{\substack{\forall w_i \in V \\ \forall c_j \in C^n}} p(c_j|w_i)} \tag{3}$$

i.e., the sum of class membership probabilities of every vocabulary word with respect to c_j. The corresponding maximum likelihood estimate for the bigram probability of a sequence c_j, c_{j+1} is

$$\Big(\hat{p}(c_{j+1}|c_j)\Big)_{ML} = \frac{\sum_{\forall w_i \in V} p(c_{j+1}|w_i)p(c_j|w_i)}{\sum_{\forall w_i \in V} p(c_j|w_i)} \tag{4}$$

In the case of unseen bigrams we use the back-off language modeling technique to estimate the bigram probability as follows:

$$\hat{p}(c_{j+1}|c_j = \textit{backoff}(c_j)\hat{p}(c_{j+1}) \tag{5}$$

The proposed soft class language model is built on both lexical and semantic context and differs somewhat from traditional class-based language models.

3.2. Induction of Semantic Classes

The proposed system using the soft clustering algorithm works similarly to the hard clustering system, with the addition of the class membership calculation algorithm. The soft-clustering algorithm consists of three steps *class generator*, *membership calculator* and *corpus parser*, as shown in Fig. 2. An example 1^{st} system iteration is also shown in this figure.

3.2.1. Class Generator

First, each corpus word is transformed to the triplet format. Second, a soft class n-gram language model is built, as defined in Section 3.1. Then the KL distances between words are computed according to Eqs. 1 and 2. Note that the probabilities are computed using the generalized n-gram estimation Eqs. 3, 4

[1] The algorithm has no concept of these classes and the above labels are used only for example reasons. In practice alphanumerics are used for each semantic class as it is created.

[2] Note that, as shown in Fig. 1, words that are not (yet) candidates for any semantic class have a lexical form probability equal to one, e.g., (flights, flights,1).

and 5. Next a set of semantic classes is generated using the pair merging strategy, described in Section 2.1. For each candidate class the class membership probability is computed using the *membership calculation* algorithm outlined next.

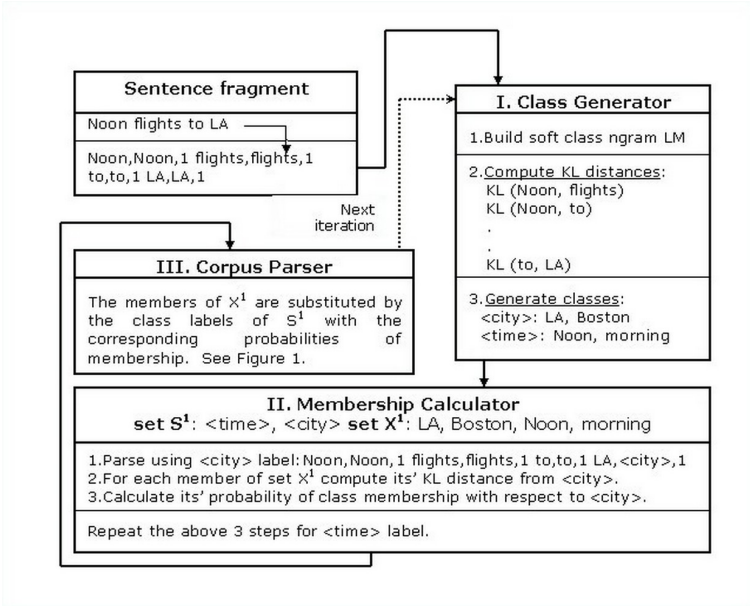

Figure 2: Soft clustering system architecture and example iteration.

3.2.2. Membership Calculator

Given the set of semantic classes S^n generated at the n^{th} system iteration, the probability of class membership between words and each class s_j of S^n is computed. This is not done for the entire corpus vocabulary, but only for the words that were assigned deterministically to the classes of S^n by the *class generator*. In other words, we relax the word-class hard assignment to word-classes soft assignment but otherwise keep the iterative process of word to class assignment as in the hard clustering algorithm. Let the words that are assigned to classes up to iteration n be members of a set $X^n \subset V$. Also, recall that each word member of X^n is retained (assigned to itself) with fixed probability equal to 0.5. The probability of class membership between a word, w_i, and a class (or itself), c_j, is given by the following equations:

$$p(c_j|w_i) \equiv 0.5, \tag{6a}$$

if $c_j = w_i$ and $w_i \in V$, and

$$p(c_j|w_i) \equiv \frac{e^{-KL(s_j,w_i)}}{\sum_{\forall s_j \in S^n} e^{-KL(s_j,w_i)}}, \tag{6b}$$

where $c_j \in S^n$ and $w_i \in X^n \subset V$.

The KL distance between a word w_i and a class $c_j = s_j$ is computed as follows: (i) the corpus is parsed and each word in c_j (excluding w_i) is substituted by the appropriate class label, (ii) a bigram language model is built using Eqs. 3-5, and (iii) the KL distance is calculated using Eq. 2. Then the equations above are applied to compute the probability of class membership.

The motivation behind Eq. 6b is that words that are semantically similar to a class are member candidates for this class. The enumerator of Equation 6b distributes exponentially less membership probability mass to the classes that have greater KL distance from the word w_i . The exponential form of Equation 6b has more drastic separability regarding strong and weak class candidates compared to a linear equation. Eq. 6b is a slightly modified reverse-sigmoid membership function, commonly used in fuzzy logic.

Note that the total probability of class membership for every soft-clustered word $w_i \in X^n$ equals to 1, i.e.,

$$\sum_{\forall c_j \in C^n} p(c_j|w_i) = \overbrace{\sum_{\forall s_j \in S^n} p(c_j = s_j|w_i) + p(c_j = w_i|w_i)}^{0.5+0.5} = 1, \tag{7}$$

where $w_i \in X^n$. The equation implies a linear, fixed combination between lexical and semantic information, which are equally weighted. Every word of X^n is allowed to participate into the generated classes of S^n with membership probabilities summing to 0.5, while it is also lexically retained with a fixed probability equal to 0.5.

3.2.3. Corpus Parser

The *corpus parser* re-parses the corpus and substitutes the words in the middle field of the triplet with the appropriate class labels, assigning the corresponding probabilities of class membership to the third field. For example, given that the word "Noon" was assigned to the classes <time> and <city> with membership probabilities 0.9 and 0.1 respectively, it is parsed as (Noon,<time>,0.9) and (Noon,<city>,0.1), as is shown in Figure 1.

Additionally, every corpus word is lexically retained with fixed probability equal to 0.5, if it was soft clustered (else 1). For example, the word "flights" was not grouped to any induced class by the *class generator*. The *corpus parser* keeps its' lexical form as (flights,flights,1). For the word "Noon" for instance, that was soft-clustered to the classes <time> and <city> the lexical probability is 0.5.

4. Experimental Corpus and Procedure

We experimented with the ATIS corpus which consists of 1,705 transcribed utterances dealing with travel information. The total number of words is 19,197 and the size of vocabulary is 575 words.

We studied the performance of the proposed *soft-clustering* algorithm in terms of precision and recall. We compare the soft-clustering to the *hard clustering* algorithm where a word is assigned deterministically only to a single induced class [1]. Also, we conducted a hard-clustering experiment where the semantic classes are induced in a single iteration, henceforth referred as *lexical*, In the *lexical* experiment, no generated labels are imported to the corpus and only lexical information is exploited for class induction. Finally, we conducted an hard-clustering experiment where *semantic* and *lexical* information is combined using equal and fixed weights of 0.5, henceforth referred as the *hard+lexical* experiment. These additional experiments are included to help us better understand the cause of improvement of the proposed algorithm vs the one in [1]; specifically if the improvement is due to mixing lexical and semantic information, or using soft- instead of hard-clustering (or both).

The three components of the proposed soft-clustering al-

gorithm are run sequentially and iteratively over the corpus, as described in Section 3.2. The following parameters must be defined: (i) the total number of system iterations (SI), (ii) the number of induced semantic classes per iteration (IC), and (iii) the size of Search Margin (SM) defined in Section 2.1. The same iterative procedure and parameters are also followed and defined for the hard-clustering algorithm, described in Section 2. Regarding the lexical experiment, the class generator of Figure 2 is run once ($SI=1$), generating the total required semantic classes for evaluation.

5. Evaluation

For the evaluation procedure of the ATIS corpus, we used a hand-crafted semantic taxonomy, consisting of 38 classes that include a total of 308 members. Every word was assigned only to one hand-crafted class. For experimental purposes, we generated manually characteristic "word chunks", e.g., T W A → T_W_A. Also, all of the 575 words in the vocabulary were used for similarity computation and evaluation. An induced class is assumed to correspond to a handcrafted class, if at least 50% of its members are included ("correct members") in the handcrafted class. Precision and recall are used for evaluation as in [1].

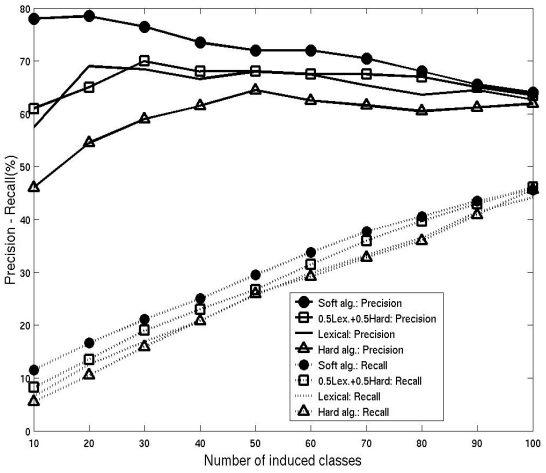

Figure 3: Precision and recall of *soft*, *hard*, *lexical* and *hard+lexical* algorithms on the ATIS corpus.

Figure 3 presents the achieved precision and recall for the *soft* and *hard* clustering algorithms, and also for the *lexical* and *hard+lexical* ones. Precision and recall were computed for 80 induced semantic classes, using SM=10.

The proposed *soft* algorithm generated 80 classes at the 3^{rd} iteration. During the previous two iterations we calculated the *probability of class membership* over 15 induced classes (5 and 10 classes at 1^{st} and 2^{nd} iteration). The *hard* and *hard+lexical* algorithm was run for 3 iterations, generating 5 deterministic classes at 1^{st} iteration, 10 at 2^{nd} and the rest 65 classes at the 3^{rd} iteration. During the *lexical* experiment 80 classes were generated at 1^{st} iteration.

The proposed *soft* algorithm (•) outperforms the other approaches (*hard* (△), *lexical* and their combination *hard+lexical*), especially for the first 40 induced classes, in terms of precision. It is also interesting that the *lexical* algorithm outperforms the *hard* clustering algorithm (△). Regarding recall scores, the *soft* algorithm (•) is shown to achieve consistently higher results than the other approaches. Also the fixed combination *hard+lexical* performs slightly better than the the other two "hard" algorithm indicating that the combination between lexical and semantic information does provide some performance advantage.

6. Conclusions and Future Work

In this paper, a soft-clustering algorithm for auto-inducing semantic classes was proposed that combines lexical and semantic information. It was shown, that the proposed algorithm outperforms state-of-the-art hard-clustering algorithms such as [1]. It was also shown that most of the improvement is due to the introduction of soft-clustering (via a probabilistic class-membership function) and less so to the combination of lexical and semantic information for class induction.

We are currently investigating the effectiveness of the soft-clustering algorithm for various application domains, as well as computational complexity issues (compared with hard-clustering). We are also investigating the optimal combination of various metrics of lexical and semantic information in the semantic similarity distance.

Acknowledgments This work was partially supported by the EU-IST-FP6 MUSCLE Network of Excellence.

7. References

[1] Iosif, E., Tegos, A., Pangos, A., Fosler-Lussier, E., Potamianos, A., "Unsupervised Combination of Metrics for Semantic Class Induction," In: Proc. SLT, 2006.

[2] Fosler-Lussier, E., Kuo, H.-K. J., "Using Semantic Class Information for Rapid Development of Language Models Within ASR Dialogue Systems," IN: Proc. ICASSP, 2001.

[3] Hearst, M., "Automatic Acquisition of Hyponyms from Large Text Corpora," IN: Proc. COLING, 1992.

[4] Siu, K.-C., Meng, H.M., "Semi-Automatic Acquisition of Domain-Specific Semantic Structures," In: Proc. EUROSPEECH, 1999.

[5] Pargellis, A., Fosler-Lussier, E., Lee, C., Potamianos, A., Tsai, A., "Auto-Induced Semantic Classes," Speech Communication. 43, 183-203., 2004.

[6] Herbert R., Goodenough, B.J., "Contextual Correlates of Synonymy," Communications of the ACM, vol. 8, 1965.

[7] Pereira, P., Tishby, N., Lee, L., "Distributional Clustering of English Words," ACL, 1993.