FUZZY SEMANTIC CLUSTERING OF WORDS AND COLLOCATIONS

Master Thesis
September 15, 2014 - March 13, 2015
SONY EuTEC - Stuttgart, Germany
École Polytechnique Fédérale de Lausanne
Communication Systems

Author:
Aubry Cholleton, aubry.cholleton@epfl.ch

Supervisors:

Dr. Fritz Hohl
SONY, European Technology Center (EuTEC)
Speech and Sound Group (SSG)

Dr. Jean-Cédric Chappelier
École Polytechnique Fédérale de Lausanne
Artificial Intelligence Laboratory (LIA)

March 2015

Abstract

Many of the state-of-the-art Natural Language Processing components such as Named Entity Recognition or Semantic Role Labeling are based on supervised machine learning techniques. The performance of these techniques is often highly influenced by the quality of a set of features which can well represent the characteristics of signs. Semantics is an inherent property of signs in our language, and it can give us a lot of information about a piece of text. However, extracting, representing and using it as a feature in an NLP component is not a trivial task.
First, single words do not always carry enough relevant semantic information, especially in *collocations* such as "white house", where two words have to be considered as a single semantic entity. To address this issue, we propose an implementation and evaluation of a new collocation extraction system which combines a syntactic rule-based filter together with some test-statistics using various association measures.
Secondly, representing semantic information is difficult, since complex hierarchical and fuzzy relations might occur (polysemy, homonymy, hypernymy, holonymy ...). Traditional approaches like hard clustering, in which each word or collocation is mapped to a single semantic cluster, are therefore not ideal. In order to model complex semantic relations in a better way, we use the Brown hierarchical clustering algorithm and a fuzzy semantic word clustering technique based on Latent Dirichlet Allocation introduced in [Chr11], that we extend to cluster collocations. We also introduce a method to name clusters using heuristics and semantic relation graphs. Finally, we use our semantic clusters to build features in order to improve two existing NLP components, namely a Named Entity Recognizer based on Conditional Random Fields, and a Semantic Role Labeler based on Maximum Entropy Models. Using these features, nearly 3% gain in F1-score can be observed on a Named Entity Recognition system, and a 0.7% improvement in F1-score for predicate and semantic arguments classification is achieved. Similarly, we show that fuzzy clustering of collocations in addition to single words does improve the overall performance under certain conditions.

Contents

List of Figures

List of Tables

Abbreviations

CoNLL Conference on Natural Language Learning. 47, 60

CRF Conditional Random Field. 3, 9, 43, 50, 59, 60, 68

EM Expectation Maximization. 27

FCM Fuzzy c-mean. 27

GMM Gaussian Mixture Model. 27

HMM Hidden Markov Model. 27, 43

LDA Latent Dirichlet Allocation. 5, 27, 29–31, 33–35, 37, 40, 43, 52, 56, 61, 65, 67, 68

LM-BFGS Limited-Memory Broyden-Fletcher-Goldfarb-Shanno. 44, 60

MCMC Markov chain Monte Carlo. 30

MEM Maximum Entropy Model. 3, 59

MEMM Maximum Entropy Markov Model. 43

NE Named Entity. 43, 46, 47, 53, 54

NER Named Entity Recognizer. 5, 51, 68

NER Named Entity Recognition. 43, 44, 46, 56, 61, 62, 67, 68

NLP Natural Language Processing. 5, 35, 36, 68

PLSA Probabilistic Latent Semantic Analysis. 12, 27, 29

POS Part-of-Speech. 18, 43, 44

SRL Semantic Role Labeler. 5

SRL Semantic Role Labeling. 61, 62, 68

SVM Support Vector Machine. 3

1 Introduction

Semantics, from the ancient Greek σημαντικός, *significant*, is the "science of the signification" of words and expressions [Bré04]. Semantics is a complex topic which involves the understanding of properties and relations behind the meaning of entities. We define an entity as a semantical unit which cannot be broken into smaller units. In most of the cases, such an entity will only consist of a single word (unigram), but can also be composed of several words ("New York", "red phone"), in this case, it is called a *collocation*. A more formal definition of *collocations* is given in Chapter 2. Among the semantic properties and relations of entities, several will be of main interest in this thesis.

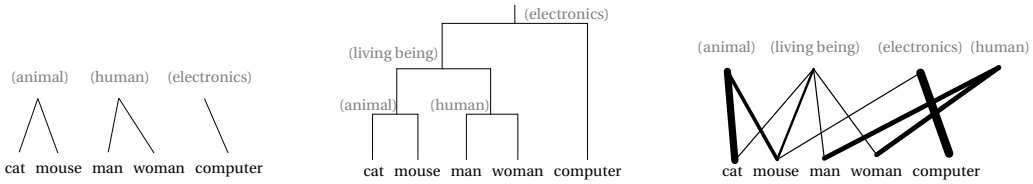
Polysemy This property expresses the fact that an entity can have different related meanings in the same semantic domain, which also share the same etymologi. For example "wood" may describe the material from which trees are made, as well as an area with a lot of trees.

True Homonymy Sometimes, a given entity has two different meanings which do not share any common root or concepts, in this case we call them *homonyms*. For example the word "bass" can designate both a music instrument (from latin *bassus*, "low") or a fish (from Proto-Germanic *bars*, "sharp"). These two meanings are obviously unrelated, which is what differentiates this notion from polysemy.

Hyponymy / Hypernymy Some hierarchical relations also exist between entities in the same semantic field, for example "cat" and "mouse" are said to be two *hyponyms* of "animal". And "animal" is said to be the *hypernym* of "cat" and "mouse".

Meronymy / Holonymy Another type of hierarchical semantic relationship which denotes that an entity is a constituent of another entity. We say for example that "Bavaria" is a *meronym* of "Germany", and "Germany" is the *holonym* of "Bavaria".

Synonymy Two entities are synonyms if their meaning is identical or at least very similar. (e.g. "cluster" and "group")



(a) Hard Clustering (b) Hierarchical Clustering (c) Fuzzy Clustering

Figure 1.1 – Illustration of different clustering techniques. For clarity the clusters in the figure are annotated with a semantic concept in gray.

Recently a lot of research has been done on how to extract and represent the semantics of words and expressions, most of which relies on the distributional hypothesis [Har54], summarized in [Fir57] by the fact that "the complete meaning of a word is always contextual, and no study of meaning apart from context can be taken seriously.". In other words, we can derive the meaning of entities by analyzing the entities which occur within a window of limited length around it.

1.1 Clustering

One of the most simple ways of representing semantic information is clustering. Clustering corresponds to the action of automatically grouping similar elements together (in our case, entities with similar semantic properties). Although assigning a single cluster (representing one meaning) to an entity can be sufficient to model some semantic aspects, such as simple synonymy, it obviously systematically fails to reflect more advanced properties like polysemy or hyponymy. For this reason some more advanced distributional clustering techniques have been introduced, in order to add a hierarchy to the clusters [BDM+92], or to assign several clusters to a single entity in a fuzzy way [BEF84, Chr11]. One of the advantages of clustering is that this task is usually done in an unsupervised way; making use of the distributional hypothesis, it does not require any other information than the data itself (in our case the text) to perform semantic clustering.

Figure 1.1 illustrates an example output of several clustering methods. Graph (a) corresponds to a standard clustering approach where only one cluster is assigned to an entity. We can see that the "mouse" word is here not related to the "electronics" cluster, therefore ignoring the "computer mouse" meaning (polysemy). In the second graph, which illustrates hierarchical clustering, it is possible to see that the clusters are forming a tree, and a relation is now visible between all the words corresponding to living beings (hyponymy, hypernymy), which is not there in the previous method. However the relation between "mouse" and the electronics cluster is still missing. Finally, graph (c) which illustrates fuzzy clustering can model most

of these relations, notice that each word is connected to several clusters, and that the link between the words and the clusters can have different strength. More details about clustering techniques will be discussed in Chapter 3.

1.2 NLP Components

1.2.1 Named Entity Recognition

Named Entity Recognition is one of the most important tasks in Natural Language Processing. A Named Entity Recognizer is a component which can automatically identify and classify Named Entities in an input text. Named entities can be for example names of *persons* (e.g. "Noam Chomsky"), *locations* (e.g. "Stuttgart", "Germany"), *organizations* (e.g. "U.N.", "European Union"). As an example, consider the sentence "Noam Chomsky is going to Geneva to visit U.N..". A Named Entity Recognizer is expected to produce the following output "**Noam Chomsky**$_{\text{person}}$ is going to **Geneva**$_{\text{location}}$ to visit **U.N.**$_{\text{organization}}$.". Identifying named entities is a key preprocessing step to several NLP tasks, such as machine translation, information retrieval, word sense disambiguation, etc.

1.2.2 Semantic Role Labeling

Semantic Role Labeling corresponds to the task of automatically identifying and classifying predicates and their semantical arguments and modifiers in a sentence. For example, if we consider the sentence "He will not give you the answer.", the output of a semantic role labeling system would be the following :

[He$_{\text{Arg0}}$,$_{\text{Arg0}}$] [will$_{\text{ArgModifier-mod}}$] [not$_{\text{ArgModifier-neg}}$] [give$_{\text{PRED:give.v.1}}$] [you$_{\text{Arg1}}$,$_{\text{Arg0}}$] the [answer$_{\text{Arg2}}$,$_{\text{PRED:answer.n.1}}$].

Where the red subscripts correspond to the verb predicate "give" and the blue ones apply to the noun predicate "answer". The labels of the arguments comes from the entry of the predicate sense in a *propbank*. A *propbank* is a resource which identifies for every senses of each predicate, the semantic nature of its arguments. An example entry ("give.n.1") is shown in Table 1.1. When a corpus is annotated with information from a propbank, it is said to be a propbank corpus.

1.2.3 Supervised learning

Unlike the clustering task, which can be performed in an unsupervised way, without other information than the input text itself, these components often use some supervised machine learning techniques like Conditional Random Fields (CRFs) [LMP01, FGM05], Maximum Entropy Models (MEMs) [LCL$^+$05] or Support Vector Machines (SVMs) [CV95, HPW$^+$04],

V	Verb
Arg0	giver
Arg1	given to
Arg2	thing given
Arg3	attribute
ArgModifier-mod	modal
ArgModifier-neg	negation

Table 1.1 – PropBank entry for “give.n.1”

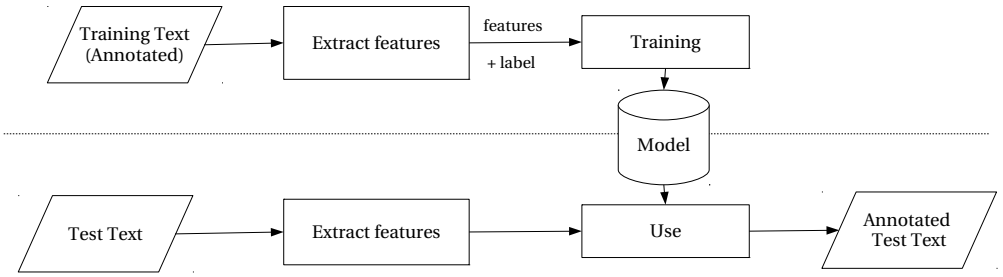


Figure 1.2 – Structure of a typical Machine Learning System

which are trained on instances annotated with a given label (Named Entity tag, Semantic role label ...). The structure of such a supervised machine learning system is shown in Figure 1.2. During the training phase (top half of the graph) the system learns a model from some annotated instances. Each training instance consists of a feature vector and a label. In our case this vector contains information about the current entity, together with information about the context around this entity. There are many ways to build such a feature vector, the most common one being a combination of the *one-hot* binary representation of the token or lemma, of the part-of-speech tag, of some syntactic properties (such as capitalization) for the current word and the neighboring words, together with some bi-gram combinations of these features. An example of such a vector can be seen in Table 1.2. Once the model is trained, it can be used to classify unannotated (unlabeled) text data. Chapter 4 and 5 will provide the details about two such systems.

It is known that extending the feature vector with some semantic clustering information learned in an unsupervised way can improve the performance of such Natural Language Processing components at a lesser cost [Chr11, TRB10, KBK15].

Role	Feature vector						Label
Feature	Token	Part-of-Speech Tag	Capitalization	Previous token	POS tag of the previous word	...	Named Entity Type
Value	house	NN	No_upper	the	DT	...	O
Vector encoding	00000001000...000	0001000	0	00010000000...000	0100000	...	0000100

Table 1.2 – Example of a feature vector and its one-hot encoding

1.3 Overview

In order to extract and use semantic information from entities in Natural Language Processing (NLP) components, we will perform three main steps. It is first necessary to identify collocations, in order to know whether we have to consider a single word or a group of word as a semantical unit. The second task is to extract and represent the semantic information of these words and collocations using a clustering approach. And the third task is to find some good parameters and generate good features from these representations in order to improve the performance (in terms of precision and recall) of two Natural Language Processing components such as Named Entity Recognizer (NER) and Semantic Role Labeler (SRL).

The first part of this thesis will focus on the collocation extraction task, after describing the difficulties of this task and presenting the state of the art techniques, we will describe our implementation of a new collocation extraction system which combines a rule-based filter together with a statistical hypothesis testing approach based on the best association measures from the literature (Pointwise Mutual Information, Mutual Dependency [Pec10], ...). We will perform an evaluation of our system and the methods we used.

The second part of the thesis will concern the semantic clustering task, and its applications on two Natural Language Processing components (Named Entity Recognition, Semantic Role Labeling) of an existing NLP toolkit internally used in Sony Speech and Sound Group. After describing the Brown hierarchical clustering algorithm, and a fuzzy clustering technique based on LDA [Chr11] in Chapter 3, we will investigate how to extend this method from unigrams to collocations. Finally we will integrate these techniques in a Named Entity Recognizer (based on Linear-chain Conditional Random Fields) and a Predicate Structure Analyzer (based on Maximum Entropy Models) by building features from the obtained clusters, and evaluate to what extent these new features can improve the performance of such systems and at what cost (in Chapters 4 and 5).

Figure 1.3 is illustrating the global structure of our system. Each component will be discussed in more detail through this thesis.

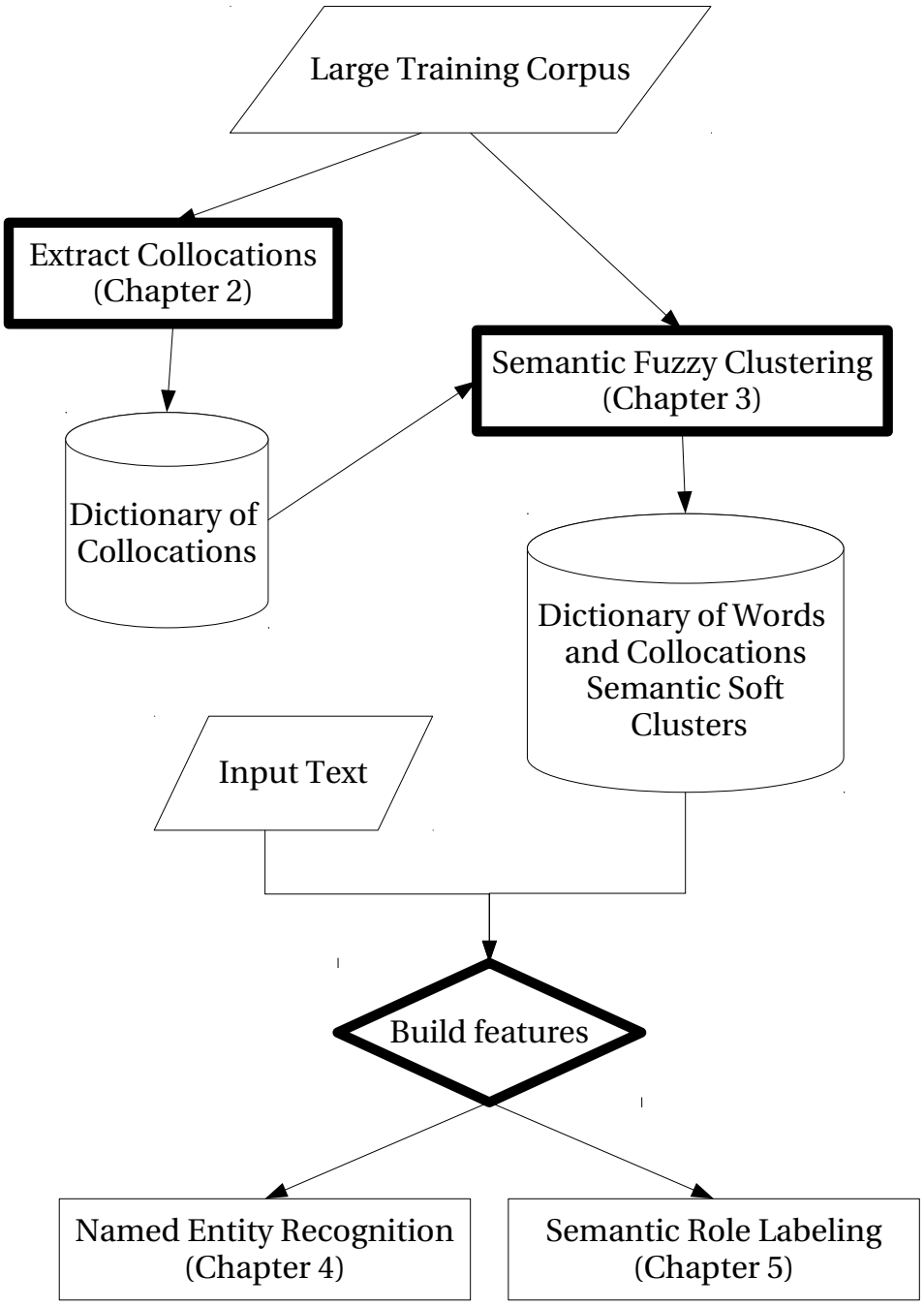


Figure 1.3 – Overview of our system

2 Collocation Extraction

In every language, some words tend to occur next to each other "very often and more frequently than would happen by chance" [HT11]. This might be due to the fact that they correspond to an *idiomatic expression* (e.g. "raining cats and dogs"), a *named entity* (e.g. "Noam Chomsky", "New York", "The Catcher in the Rye"), a *nominal or adjectival compound* (e.g. "high school", "red phone"), a *verb-particle construction* (e.g. "turn on", "go out"), a *light verb construction* (e.g. "give a hug", "make a mistake") or some other special syntactic or semantic structures. Our goal here is not to make an exhaustive list of these cases, as a deeper investigation in this direction has been done by [SBB+02].

Although a lot of various definitions could be accepted for the term *collocation*, "considering every group of words with a high co-occurrence frequency" is a bit too wide for our applications. Since our goal is to eventually cluster these collocations semantically, we would like to recognize collocations only if they form a semantical unit. A *collocation* will thus be defined as "a sequence of two or more consecutive words, that has characteristics of a syntactic and semantic unit whose exact and unambiguous meaning cannot be derived from the meaning or connotation of its components" [Cho88], which is more appropriate for our needs and seems to be the most commonly used in the literature [Pec10, SJ01, BZL11, MS99]. As you might have noticed, some categories listed in the first paragraph do not fulfill these conditions because they are compositional; especially *light verb constructions*, some kind of *named entities* like books titles, or some compositional idioms. Note that the terms *Multi-Word Unit* [SJ01] and *(connected) Multi-Word Expression* are sometimes used as a synonym of *(non-compositional) collocation*.

2.1 Properties of collocations

Several noticeable properties of collocations have been inferred from this definition [MS99].

Limited compositionality This definition particularly emphasizes the *non-compositional* semantic nature of collocations, which is considered as the most important characteristic of collocations in the literature. The meaning of the parts of a collocations cannot be composed to deduce the meaning of the whole collocation. As an example, in "high school", the meaning of the whole collocation cannot be fully derived from the word "high", which is here not qualifying the altitude, nor from the word "school" which could designate any kind of school. Nevertheless, with the bi-gram "red car", one can easily get the full meaning of it by composing the individual meaning of "red" and "car".

Limited modifiability The concept of *Non-modifiability* (or *Non-decomposability* [BZL11]) of collocation can also be discussed [MS99]. It states that the syntactic structure of a collocation is static; it is most of the time not possible to change the order of the words or to add a word to the expression. As an example, "cut and paste" is a common technical collocation but "paste and cut" does not make any sense.

Limited substitutability It is not possible to replace one of the words composing a collocation by a synonym or by a similar word. Consider the collocation "short circuit", substituting "short" by "small" would completely change the meaning of the whole collocation. A "short circuit" being, according to http://wiktionary.org, "a usually unintentional connection of low resistance or impedance in a circuit such that excessive and often damaging current flows in it.", whereas a "small circuit" is just a circuit which has a limited size.

These properties are not only theoretical concepts, but they could actually be used as a way to automatically extract collocations from a corpus.

2.2 Collocation Extraction

2.2.1 Motivation

According to Anastasiou [AH09], several topics are important when considering collocations. In this chapter we will mainly discuss the *identification* topic, which they define as the automatic identification of collocations in a text. We will investigate which techniques are commonly used to perform this task, and propose an implementation of a collocation extraction system together with a detailed evaluation of the results.

Identifying collocations is the first step towards various new applications, and can significantly improve the performances of existing applications in various fields, such as machine translation, parsing, information extraction and information retrieval [Pec05].

In our case, we would like to extract collocations in order to improve the performances of a semantical clustering algorithm. Consider the sentence "Named Entity Recognition is an interesting Natural Language Processing task.". Let's assume that we are using a simple distributional semantic clustering algorithm which is using contextual features. Also assume that we set the context window to 1 and that we filter stop-words before running the algorithm. Without taking collocations into account, the semantical clustering of the word "natural" would be wrongly influenced by the word "language" and "interesting". Instead it would be much more relevant to cluster "Natural Language Processing" as a single entity using the words "interesting" and "task".

2.2.2 Identification

Rule-based approach

According to our definition, collocations are also *syntactic units*, which suggests that we could actually try to recognize them by looking for n-grams matching some appropriate syntactic rules. For example ADJ NOUN or NOUN NOUN are some typical collocation forms [WSN10, Ser11]. If this technique alone provides some poor performances, and requires the output of a part of speech tagger, it can significantly improve the performance of a system when some simple syntactic patterns are used as a preprocessing filtering step. We use this technique as an optional filtering step in our collocation extraction system.

Some more advanced rule-based systems make use of a dependency-parser [Lin99, Lin], which allows a better capture of the relation between related non-contiguous words.

Supervised learning methods

Machine Learning techniques can be successfully used to identify collocations; for instance a CRF classifier has been trained on the Wiki50 corpus annotated with collocations using the feature set used in Szarvas et al [SFK06, VNB11]. Their goal was not only to identify collocations but also to classify them by type. We did not follow this approach since we did not have a large enough corpus annotated with collocations to train such a system.

Unsupervised statistical methods

The very first definition of a collocation we stated, i.e. "some words which tend to occur next to each other more often than it could happen by chance" (page 7), suggests that we could use some *hypothesis testing* techniques to identify collocations. Let's consider a corpus as a sequence of N words, where each word is modeled by a random variable w_i. In this case the *null hypothesis* would be that the words in the n-gram occurs together by chance, i.e. that they

are statistically independent, which by definition means :

$$P(w_1 w_2 ... w_n) = \prod_{i=1}^{n} P(w_k)$$

And our *alternative hypothesis* would be that the words forming the n-gram are not statistically independent, i.e. $P(w_1 w_2 ... w_n) \neq \prod_{i=1}^{n} P(w_k)$

We estimate $P(w_1 w_2 ... w_n)$ and $P(w_i)$ by maximum likelihood given the corpus. Therefore

$$P(w_i) \approx \frac{f_{w_i}}{N}$$

and

$$P(w_1 w_2 ... w_n) \approx \frac{f_{w_1 w_2 ... w_n}}{N}$$

Where f_{w_i} is the number of time the term w_i occurs in the corpus. We can also estimate the expected frequency under the *null hypothesis* which is

$$\xi_{w_1 w_2 ... w_n} := \frac{\prod_{i=1}^{n} f_{w_i}}{N}$$

In the literature, collocations extraction techniques are usually applied and evaluated on bi-grams only. We will discuss later the reason for the need to consider longer n-grams, but we would like to be able to handle n-grams of any length. However, some of the methods we will use would be complex to generalize for n larger than 2. For this reason will use the approximation introduced by Schone [SJ01] which redefines any n-gram $w_1 w_2 ... w_n$ as a bi-gram $W_1 W_2$ with $W_1 = w_1 w_2 ... w_i$ and $W_2 = w_{i+1} w_{i+2} ... w_n$ such that $i = \arg\max_i P(W_1) P(W_2)$. Any n-gram can then be statistically treated as a bi-gram (consisting of only two random variables) $W_1 W_2$. We can now redefine

$$P(w_1 w_2 ... w_n) \approx P(W_1 W_2) \approx \frac{f_{W_1 W_2}}{N}$$

and

$$\xi(w_1 w_2 ... w_n) \approx \xi(W_1 W_2) \approx \frac{f_{W_1} f_{W_2}}{N}$$

Various association measures can be used to measure the collocation association between two words; Pecina performed an impressive evaluation of 82 different association measures on his Czech dataset [Pec05]. We chose to focus on the most promising ones and the most common ones that we would like to integrate to our system, namely *t-score,z-score, chi-squared test, dice score, symmetric conditional probability, point-wise mutual information, mutual dependency* and *unigram sub-tuples*.

T-score [CH90]

$$\frac{f_{W_1W_2} - \xi_{W_1W_2}}{\sqrt{f_{W_1W_2}(1-(f_{W_1W_2}/N))}} \approx \frac{f_{W_1W_2} - \xi_{W_1W_2}}{\sqrt{f_{W_1W_2}}} = \frac{f_{W_1W_2} - \frac{f_{W_1}f_{W_2}}{N}}{\sqrt{f_{W_1W_2}}}$$

Z-score [BR73]

$$\frac{f_{W_1W_2} - \xi_{W_1W_2}}{\sqrt{\xi_{W_1W_2}(1-(\xi_{W_1W_2}/N))}} \approx \frac{f_{W_1W_2} - \xi_{W_1W_2}}{\sqrt{\xi_{W_1W_2}}} = \frac{f_{W_1W_2} - \frac{f_{W_1}f_{W_2}}{N}}{\sqrt{\frac{f_{W_1}f_{W_2}}{N}}}$$

Chi-squared test [MS99]

$$\sum_{i\in\{W_1,\overline{W_1}\}}\sum_{j\in\{W_2,\overline{W_2}\}}\frac{(f_{ij}-\xi_{ij})^2}{\xi_{ij}} = \sum_{i\in\{W_1,\overline{W_1}\}}\sum_{j\in\{W_2,\overline{W_2}\}}\frac{(f_{ij}-\frac{f_if_j}{N})^2}{\frac{f_if_j}{N}}$$

where $\overline{W_i}$ denotes any entity different than W_i.

Dice Score [Dic45]

$$\frac{2f_{W_1W_2}}{f_{W1}+f_{W_2}}$$

Unigram Sub-tuples [BJ01]

$$\log\frac{f_{W_1W_2}f_{W_1\overline{W_2}}}{f_{\overline{W_1}W_2}f_{\overline{W_1}\overline{W_2}}} - 3.29\sqrt{\frac{1}{f_{W_1W_2}}+\frac{1}{f_{W_1\overline{W_2}}}+\frac{1}{f_{\overline{W_1}W_2}}+\frac{1}{f_{\overline{W_1}\overline{W_2}}}}$$

Symmetric Conditional Probability [dSL99]

$$\frac{P^2_{W_1W_2}}{P_{W_1}P_{W_2}}$$

Point-wise Mutual Information [CH90]

$$\log_2\left(\frac{P_{W_1W_2}}{P_{W_1}P_{W_2}}\right)$$

Mutual Dependency [TFK02]

$$\log_2\left(\frac{P^2_{W_1W_2}}{P_{W_1}P_{W_2}}\right)$$

Log frequency biased Mutual Dependency [TFK02]

$$\log_2\left(\frac{P^2_{W_1W_2}}{P_{W_1}P_{W_2}}\right)+\log_2 P_{W_1W_2}$$

Some authors tried to strictly enforce one or several of the properties of collocations. For example Schone and Jurafsky propose to enforce non-compositionality by applying Probabilistic Latent Semantic Analysis (PLSA) at a word level [SJ01] and to compute the correlation between the semantic vectors corresponding to each parts of a collocation. A weak correlation would be a good sign of non-compositionality, but the gain in term of performances is not worth the added complexity. Similarly, one could try to re-score a collocation by substituting one of the word by a synonym [Pea02], a much lower score would indicate a low substitutability. For our use-case, we found that a strict enforcement of the non-compositionality and non-substitutability properties of collocations is not required.

2.3 Implementation

2.3.1 Existing systems

Several collocation extraction systems are already available on-line, [RDAV12] describes some of them. We will have a brief overview of their features and characteristics of the most interesting ones.

UCS toolkit (http://www.collocations.de/software.html) This toolkit, written in Perl and R, consist of a collection of libraries and scripts which provides a very broad list of association measures to perform statistical analysis. However it supports only bi-grams, and requires a list of bi-grams together with their joint and marginal frequencies as input (an external system to compute n-gram frequencies is required).

Text NSP (http://search.cpan.org/dist/Text-NSP/) Text NSP offers the possibility to compute n-gram frequencies and a few association measures, it can supports n-grams lengths up to 4. This toolkit written in perl does not provide any evaluation tools.

mwetoolkit (http://mwetoolkit.sourceforge.net) Multiword Expression toolkit provides a full-stack MWE identification system written in Python, from n-gram and word counting in large corpora to association measures and some simple evaluation tools. It also requires preprocessing the corpus with a POS tagger. However it works only from a Unix

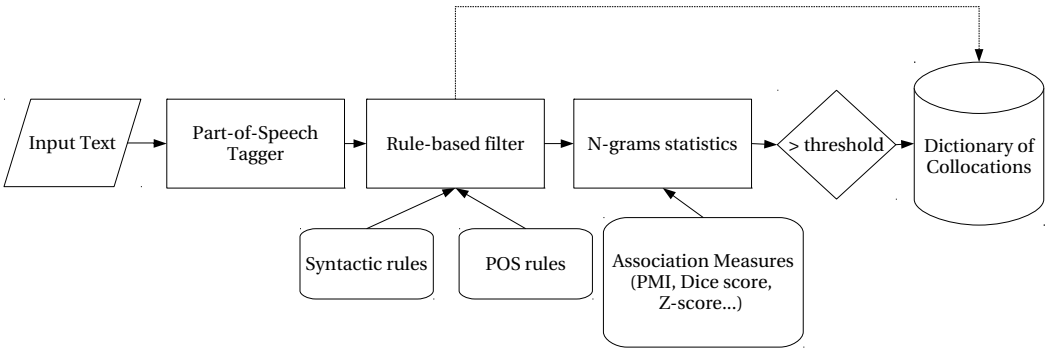


Figure 2.1 – Structure of the collocation extraction system

command line interface (no API) and it is published under the GNU GPL license v2.0.

2.3.2 System architecture

Our system is implemented in Java and can be used through a Command Line Interface or an Application Programming Interface. It can extract collocations from input text using a combination of syntactic rules defined as regular expressions on Part-of-Speech tags and lemmata as well as some n-gram statistics using association measures which can be configured (*t-score, z-score, chi squared, dice score, symmetric conditional probability, pointwise mutual information, mutual dependency, log frequency biased mutual dependency, unigram subtuples* or *raw frequency*). As input, a raw string of characters, IOB (including CoNLL) format, or already preprocessed (compressed) NGram counts tables (GOOGLE 1T corpus) can be used.

Various options are supported including the possibility to ignore n-grams with a frequency lower than a given threshold, n-grams longer than a given value or n-grams which do not occur in at least a given number of documents (in case the input consists of several documents). It is also possible to filter stop-words as a preprocessing step and to choose one or several collocation association measures.

Some evaluation tools are also part of the system. Performance evaluation (precision/recall, true/false positives/negatives) can be performed against a given gold standard. Wordnet collocations and Wiki50 annotations are the two built-in gold standards but a custom one can be provided in IOB format, or as a raw list of collocations. It is also possible to compare several gold standards. A Python script allows to easily run evaluation tasks and outputs results as graphs.

2.3.3 Complexity Analysis

Our system relies on a hash table structure which associates every n-gram in the corpus to its frequency. We suppose that the hash table can fit in the memory. If we assume that we consider n-grams up to length $k \ll N$, and that there is no hash collision, this hash table can be built in $\mathcal{O}(kN) = \mathcal{O}(N)$ steps with a similar memory complexity using a simple iterator over the input text.

The second step is to compute the score for every n-gram. For each entry of the hash table, we split the corresponding n-gram into two smaller n-grams, to get the marginal frequencies as described in the previous section. To compute the score, we then have to look up these two n-grams to get their frequencies, together with the frequency of the whole n-gram. Since the lookup operation in a hash table has a time complexity of $\mathcal{O}(1)$, computing the score for N n-grams will take $\mathcal{O}(N)$ steps.

Finally we have to return the top-k collocations, which takes $\mathcal{O}(N \log k)$ steps (to sort the scores) and still $\mathcal{O}(N)$ memory space. Therefore the overall space and time complexity of the system is $\mathcal{O}(N)$.

2.4 Intrinsic Evaluation and Results

Evaluating a collocation extraction system is not a trivial task. Indeed, the definition of a collocation itself is not well defined and as a consequence there is no widely accepted gold standard for this task. Annotated corpora are very rare because of this same reason and are also very difficult to build (since the definition of a collocation can be very subjective), which has led many scientists to use collocations contained in WordNet or some other dictionaries as a gold standard.

What are good evaluation metrics for collocation extraction? Is WordNet really a gold-standard ? How to evaluate our collocation extraction system? Which association measure performs best? How does the output of our system differ from one input to another? These are some questions we will answer in this section.

2.4.1 Evaluation metrics

We will use *recall* and *precision* as evaluation metrics. *Recall* indicates which proportion of the gold standard collocations has been retrieved from the input by our system. *Precision* indicates which proportion of the collocation retrieved from the input by our system are actually corresponding to a gold standard collocation.

It is considered that *true positives (tp)* corresponds to the retrieved collocations which are actually part of the gold standard, that *false positives (fp)* are the collocations retrieved by our system which are not part of the gold standard and that *false negatives (fn)* are the collocations

which appears in the gold standard but which are not retrieved by our system.

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

2.4.2 Gold standard choice

Wiki50

[VNB11] introduces the Wiki50 Corpus, which seems to be the only relatively large corpus in English annotated with collocations. The corpus consists of 50 Wikipedia articles, 114 570 tokens, in 4350 sentences. Collocations are annotated by type, but since our goal is only to extract collocations which correspond to our definition stated at the beginning of this chapter, we are only interested in *nominal compounds*, *adjectival compounds* and *named entities*. This corresponds to a total of 6565 unique collocations.

Wordnet

Wordnet is a large lexical database available on-line. It contains a lot of English nouns, adjectives, and verbs, grouped into sets of synonyms. It also contains some semantic information and relations about the word it contains. In our case, we are mostly interested in the headwords and especially in the ones containing multiple words, since from our definition of collocation, it is highly probable that collocations will be listed in a dictionary resource such as Wordnet. Since Wordnet contains more than 50000 multi-word entries, some authors claim that WordNet collocations form a real gold standard for the collocation extraction task [TFK02, SJ01].

To evaluate whether using the multi-word headwords of WordNet is a safe gold standard to evaluate corpus specific collocations, we have computed precision and recall of the top-N% most frequent annotated collocations in Wiki50 against collocations which occur in WordNet headwords.

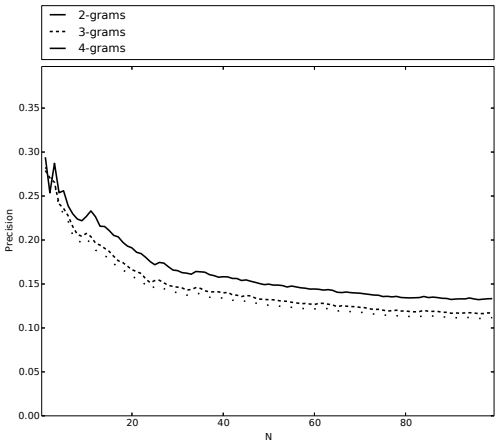


Figure 2.2 – Precision of the top-N% Wiki50 collocations against Wordnet

On Figure 2.2, we can see that about 13.32% of the bi-grams annotated as collocation in Wiki50 actually appear in WordNet. On the other hand the maximum recall is 0.82%, indicating that our Wiki50 Corpus does not cover a very broad range of topics.

We decided to investigate a bit more to determine what kind of collocations WordNet contains. If we consider only Wiki50 collocations that appear in at least 2 documents (2 Wikipedia articles), the precision jumps to 47.94%, which suggests that Wordnet contains mainly generic collocations that are not topic specific.

2.4.3 Evaluation of our system

We first evaluated our system on the Wiki50 corpus. The corpus text was provided as input, and we used the annotated collocations as gold standard to compute precision and recall for various association measures.

Wiki50 bigrams

Since we would like to use collocation extraction as a preprocessing step to improve our semantical clustering algorithms, precision is more important than recall, since a bad precision could potentially be harmful for the semantical clustering performance. Figure 2.3a shows the performance of our system using various association measures without any syntactic filtering. We used a baseline which assigns a random score to each n-gram, as we can see, log-Frequency Biased Mutual Dependency provides the best precision results for low recalls. The frequency bias was introduced by Thanopoulos to favor frequent n-grams among the most dependent ones [TFK02]. We used this association measure after inspecting the false positive outputs produced by the *Pointwise mutual dependency score*, which contains a lot of bi-grams composed from low-frequency words. Still the precision seems a bit low, but does not

actually reflect the true performance of the system, since our gold standard does not exactly fit our definition. We tried to perform an analysis of the false positives to understand what kind of bi-grams were wrongly extracted as collocation in Table 2.1. Most of the false positives are actual collocations which only appear as a part of a longer collocation (e.g. “san fransisco theater”) in the gold data, and which cannot be considered as true positives. Extracting them should not be considered as a mistake, but we did not find any way of preventing this, since dropping every subset of a collocation during evaluation would introduce a very high bias.

During the evaluation, collocations containing numbers were ignored. Pure numbers and dates cannot be considered as collocations and are thus not annotated in Wiki50, but we believe that extracting them will not affect the output of the semantical clustering.

Table 2.1 – Analysis of 30 random false positives in Wiki50 using the Pointwise Mutual Information association measure

False positives	**Cause**

`berkeley underground`
`barapulla nallah`
`juan luis`
`san francisco`
	Nested Collocations

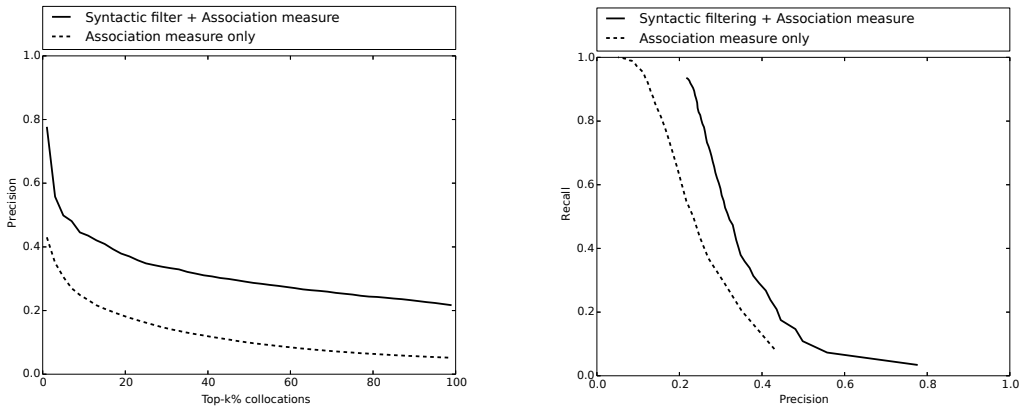
`brief demos`
`whole body`
`improved graphics`
`abandoned dorm`
`consistently associated`
	Common Expression, Compositional Bi-grams

`15,000 raised`
`90 billion`
`1985 quarterfinals`
`1982-1983 indoor`
	Dates and Numbers

`full-scale lmf` `hon. galahad`	Abbreviations
`crime watch.`	Parsing errors

Syntactic filtering

To increase the precision of our system, we added a rule-based filter, which allows to define some regular expressions on n-gram tokens and corresponding Part-of-Speech (POS) tags. After several experiments, and following the existing literature [Ser11], we finally chose to keep only the bi-grams of the form *Noun-Noun*, *Adjective-Noun*, *Adverb-Noun*, *Adverb-Adjective*, or the ones containing a foreign word. Similarly, we exclude all bi-grams containing "and", "the", "of", "a", "an". Figure 2.4 shows an improvement of almost 20% in precision due to the filtering. The recall is negatively affected, but precision was the weak point of our component and is the priority for our task.



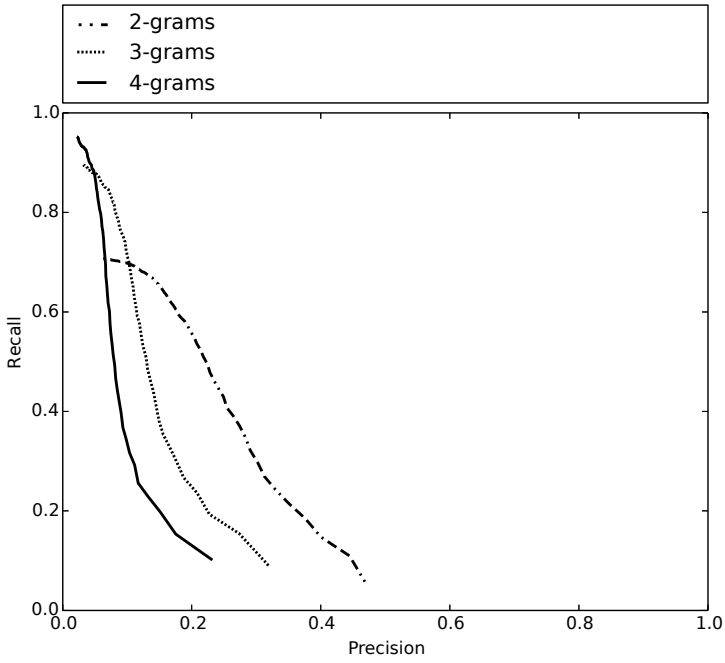
(a) Precision

(b) Precision vs. Recall

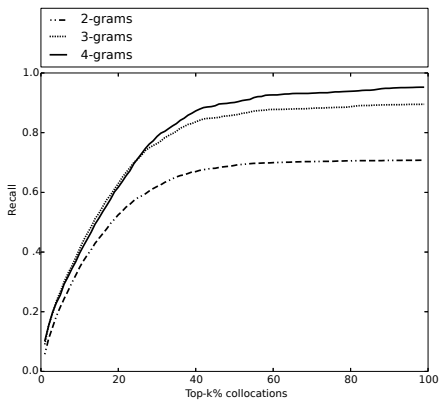
Figure 2.4 – Precision of collocation extraction with association measure and syntactic filter

Beyond bi-grams

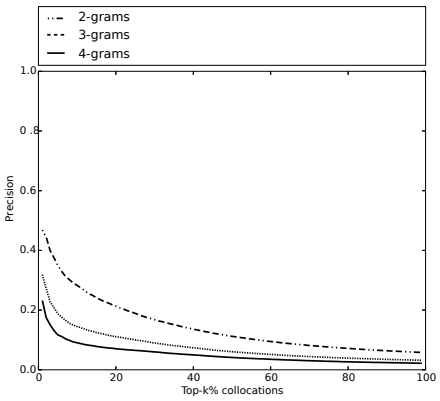
In order to determine whether it is worth considering n-grams with $n > 2$, we performed the evaluation on the Wiki50 corpus using the log-frequency biased mutual dependency association measure, considering n-grams of length $n \in \{2,3,4\}$. Figure 2.5a clearly shows that dealing with 3-grams and 4-grams introduces a high loss in precision, which we would like to avoid for our applications. On the other hand we can see that more than 70% of the collocations in the corpus are bi-grams, and more than 90% are either bi-grams or tri-grams.



(a) Precision vs. Recall



(b) Recall



(c) Precision

Figure 2.5 – Precision and Recall using log frequency biased mutual dependency measure for various n values in Wiki50

Google 1T

We also evaluated how the system behaves on a much bigger corpus, consisting of raw textual data from the web; Google 1T (1 trillion tokens). This corpus is not annotated so we performed the evaluation of the results using WordNet.

After our filtering, the input consists of 806 831 375 571 unigrams and 360 585 814 746 bigrams. To limit the computation time, and because the corpus is containing a lot of noisy data

(automatically extracted from the web, such as page menu tags, random strings, etc.) we only consider the 1.607.534 bi-grams occurring more than 800 times in the corpus. We computed the recall value of the top Google 1T collocations against the set of collocations which appear as Wordnet headwords (Figure 2.6).

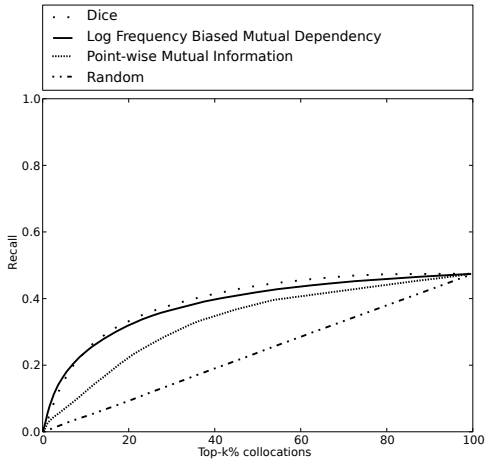


Figure 2.6 – Recall of the top-N% Google 1T collocations against Wordnet

In this case it is not relevant to evaluate precision since WordNet is obviously not an exhaustive list of all existing collocations. We can see that our system is still able to retrieve 40% of WordNet collocations in the top-30% returned results. The different association measures have the same behavior as they had on the Wiki50 input, *Dice score* and *Log-frequency biased pointwise mutual dependency* are performing best.

2.4.4 Note on disambiguation

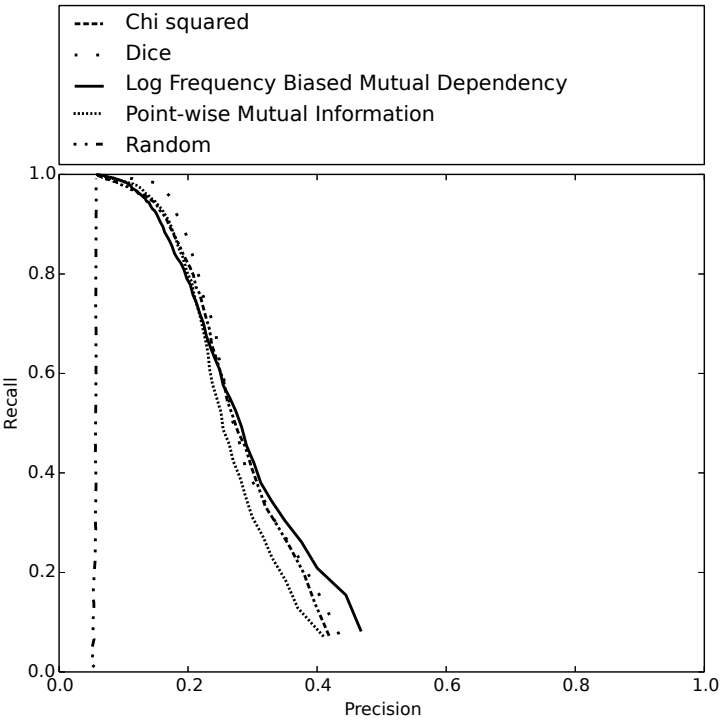
A given collocation such as “white house” could of course designate the residence of the President of the U.S., but could also be used in a particular context to describe any house of white color. Distinguishing a non-compositional from a compositional instance of a given group of words is a non-trivial task. This corresponds to the the *disambiguation* topic introduced by [AH09]. Since our task is mainly to automatically build a collocation dictionary, we assume that on a given input, compositional occurrences of a collocation are negligible, and that the performance of the system will be mainly characterized by the choice of the association measure threshold. Such a threshold has to be carefully chosen for a given application and a given input. We will discuss this point in Chapter 4.

2.4.5 Conclusion and possible improvements

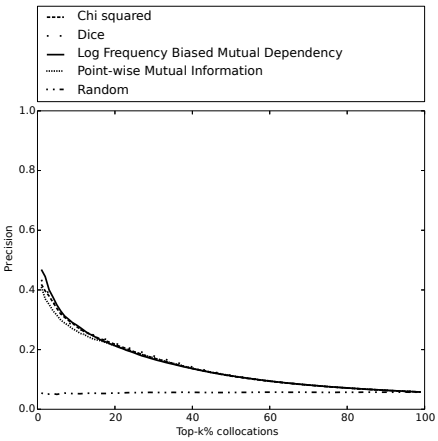
We implemented a fully usable collocation extraction system, combining two of the main techniques commonly used in the field (rule-based filter, and test statistics using association

measures), and performed an extensive evaluation on the Wiki50 Corpus.

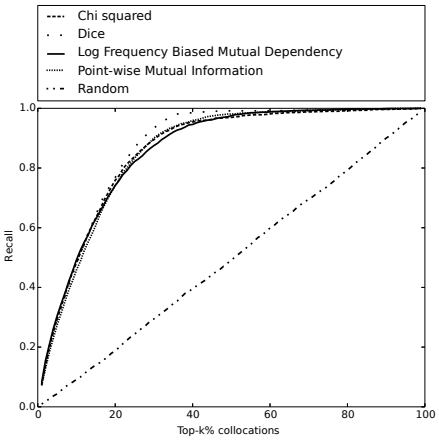
We believe that the improvement in this field will probably imply the annotation of a large corpus, in order to be able to use some supervised machine learning techniques, which are particularly good when no rules or obvious statistical model is good enough to model the complexity of a concept such as collocations.



(a) Precision vs. Recall



(b) Precision



(c) Recall

Figure 2.3 – Precision and Recall for different association measures in Wiki50

3 Semantic Clustering

Clustering corresponds to the task of automatically grouping similar elements together. Figure 3.1 is a simple illustration of this idea. A category, represented here by a color, was assigned to each data point, such that close points belong to the same category, using the popular k-means algorithm [M+67] (black crosses correspond to the centers of the clusters).

Algorithm 1 K-means algorithm

```
1 Choose the number of clusters
2 Randomly assign a cluster to each point
3 While the cluster allocation has not converged :
4   Compute the centroid of each cluster
5   Assign each point to the cluster with the closest centroid
```

Semantic clustering thus corresponds to the task of grouping words which share some similar semantic properties into the same categories. In our case, we are interested in unsupervised clustering algorithms, which require only the raw text as an input (without annotations). However, the task of semantic clustering introduces two main additional constraints; first semantic is very a complex concept and the semantic of a word cannot be reduced to a single category. For this reason standard clustering techniques such as k-means are not well suited when it comes to semantic clustering. Secondly the task of finding a good input representation for a word, which can carry the useful information required for the semantic clustering is not trivial.

In this chapter, we will describe two semantic word clustering methods which can address these issues. Both of the approaches rely on the assumption that words which are semantically similar will tend to appear in similar contexts (i.e. the distributional hypothesis).

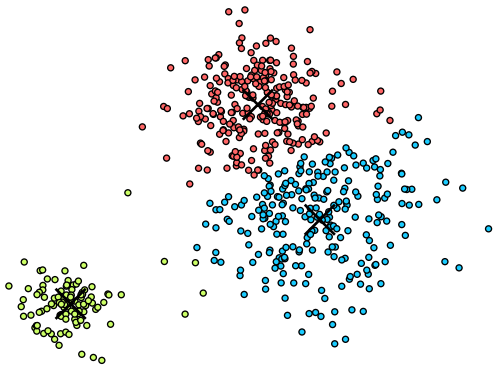


Figure 3.1 – K-means clustering of a set of data points, using 3 clusters

3.1 Hierarchical clustering

3.1.1 Brown Clustering

Brown Clustering, also called IBM clustering, is a popular clustering technique introduced in [BDM+92]. Let's assume that each word $w_1, w_2, ...w_N$ in a corpus V belongs to a given cluster. Where $\pi : V \rightarrow 1, 2, ...C$ is a partition of the vocabulary in C clusters (which maps each word to its cluster). The Brown clustering algorithm can be derived from a generative model. In this model, similar to a feed-forward hidden Markov model [BP66], it is assumed that each word has a given emission probability given one cluster, and that each cluster has a given probability of occurring, given the previous cluster. The generative process of a document is then straightforward; for each word a cluster is picked according to the previous cluster and its transition probability, and a word is then chosen according to the emission of words for this cluster.

We can express the likelihood of such a model as follow :

$$P(w_1, w_2...w_N) = \prod_{i=1}^{N} P(w_i|\pi(w_i))P(\pi(w_i)|\pi(w_{i-1}))$$

Where $P(w_i|\pi(w_i))$ is said to be the probability of emission of w_i given a cluster $\pi(w_i)$, and $P(\pi(w_i)|\pi(w_{i-1}))$ is said to be the probability of transition from cluster $\pi(w_i)$ to clus-

ter $\pi(w_{i-1})$. The log-likelihood of such a model can be expressed as

$$P(w_1, w_2 ... w_N) = \sum_{i=1}^{N} \log(P(w_i|\pi(w_i))P(\pi(w_i)|\pi(w_{i-1})))$$

Which can be rewritten as ([BDM+92], [Lia05]) :

$$\sum_{c=1}^{C}\sum_{c'=1}^{C} p(c,c')\log\frac{p(c,c')}{p(c)p(c')} + \sum_{w} P(w)\log P(w)$$

$$= MI(C,C') - H(W)$$

$MI(C,C')$ being the mutual information between adjacent clusters. Since the entropy of the words $H(W)$ does not depend on the clusters, maximizing the likelihood of this model corresponds to maximizing the mutual information between adjacent clusters, which naturally leads to the greedy algorithm 2.

Algorithm 2 Brown clustering algorithm

```
1 Initialize by creating one cluster per word
2 For the N−C merge steps:
3   Merge the two adjacent clusters c and c′ which maximizes the mutual
      information between clusters.
```

This approach consists of successively merging clusters from bottom to top, for this reason it is called an agglomerative bottom-up algorithm. The obtained clustering is hierarchical, it consists of a tree, where each node can be seen as a cluster which contains all the children clusters, and where the leaves are words. Figure 3.2 illustrates a possible subtree for the Brown clustering. At some iteration in the algorithm , “cat” and “mouse” are merged, two iterations later, the cluster containing “eat, drink” and the one containing “man, cat, mouse” are merged. By cutting the resulting tree at different depth we can choose the number and the broadness of the clusters (the top and bottom red cuts on Figure 3.2 correspond respectively to clusters 0, 11, 10 and 01, 00, 11, 101, 100).

The cost of this naive algorithm is $\mathcal{O}(N^5)$ but some more efficient approaches are implemented in practice, and achieve $\mathcal{O}(Nw^2 + T)$, where N is the number of words to cluster, and w is an initial window size [Lia05].

3.2 Fuzzy clustering

Unlike hard clustering techniques, which assigns one or several (hierarchical) clusters to each element in a boolean way (in Figure 3.1, each data point belong to a single one of the 3 clusters, each of them corresponding to a given color, with cluster’s centers marked with a cross), fuzzy clustering or soft clustering allows to assign several clusters to one element

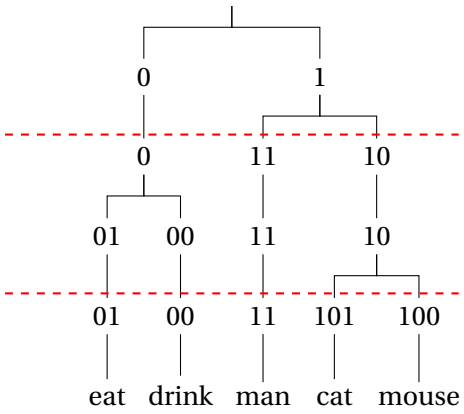


Figure 3.2 – Example Brown clustering subtree

together with memberships values to express the strength of the relation between each cluster and the element. Figure 3.6 shows the output of a fuzzy clustering algorithm, each area on the pie chart illustrates how much the word is linked to a given cluster. In the case of semantic word clustering, this obviously allows to better capture the polysemy of a word. For example the word “bass” can have several meanings, including a musical instrument or the name of a fish. Using a standard or hierarchical semantic clustering approach, we would assign only one meaning, typically the most common one, and completely throw away the other meanings, even if they could contain some very useful information for some tasks (Named Entity Recognition, Semantic Role Labeling, etc.).

Fuzzy clustering can be achieved in multiple ways ([NN06]). One of the most popular fuzzy clustering algorithm is the fuzzy c-means ([BEF84]), also called fuzzy k-mean clustering ([NN06]) which is a fuzzy variant of the k-mean clustering algorithm. Given f a fuzzifier value, algorithm 3 describes how to find the membership values $m_k(w)$, for every cluster k and every word w.

Algorithm 3 Fuzzy c-mean clustering

1 Choose K, the number of clusters
2 Randomly assign a cluster to each word
3 While the cluster allocation has not converged :
4 For each cluster $k < K$:
5 Compute the centroid $c_k = \frac{\sum_w m_k(w)^f w}{\sum_w m_k(w)^f}$.
6 For each word :
7 Recompute the membership values according to $m_k(w) = \frac{1}{\sum_j (\frac{distance(c_k,x)}{distance(c_j,x)})^{\frac{2}{f-1}}}$.

Since we would like to perform semantic fuzzy clustering on words, applying fuzzy c-mean to

our problem would require to project our words and collocations into a vector space and to find an appropriate distance measure which can model the semantic distance between words or collocations. A lot of research has already been done on how to build such a representation (e.g. tf-idf, pointwise mutual information [CH90], or some more advanced distributed word embeddings using neural networks [MCCD13, MSC+13], matrix factorization [LG14], principal component analysis [SIJ06] ...) and some successfully combined it with a fuzzy clustering technique in various domains [LW09]. We tried to apply the Apache Commons implementation of Fuzzy c-mean (FCM) on the *word2vec* vector model [MCCD13, MSC+13] trained on the first 20M words from Wikipedia, but the $\mathcal{O}(NC^2)$ time complexity of each iteration of FCM [WC11] was prohibitive given our input size N (> 10M words) and the number of clusters C needed (a few hundreds).

Another approach to the problem, similar to what is done in the case of Brown Clustering, would be to consider a probabilistic graphical model, and to introduce some latent variables to model the clusters. One could think of several such models as Gaussian Mixture Model (GMM), Hidden Markov Model (HMM) [BP66], PLSA [Hof99] or LDA [BNJ03]. It would then be possible to perform statistical inference of the parameters of the model on a large training set using the Expectation Maximization (EM) algorithm (which can here also be seen as a soft-variant of the k-means algorithm), or Gibbs sampling. The membership values of the words in clusters would thus correspond to the latent variables distributions conditioned on observed words. This is the approach we selected, more specifically using a variant of LDA, due to some previous encouraging results in the literature [Chr11].

3.2.1 Latent Dirichlet Allocation

Latent Dirichlet Allocation is one of the most popular topic modeling techniques. Its original purpose is to model the topics in a textual documents in an unsupervised way. For example consider the document in Figure 3.3 extracted from a corpus containing the first paragraph of Wikipedia articles, inferring the parameters of a LDA model on this corpus allows to estimate a topic distribution for the document (Figure 3.4), and for every document of the corpus, as well as a distribution of words for each topic (Table 3.1).

Sony Corporation, commonly referred to as Sony, is a Japanese multinational conglomerate corporation headquartered in Konan Minato, Tokyo, Japan. Its diversified business is primarily focused on the electronics (TV, gaming consoles, refrigerators), game, entertainment and financial services sectors. The company is one of the leading manufacturers of electronic products for the consumer and professional markets. Sony is ranked 105th on the 2014 list of Fortune Global 500. Sony Corporation is the electronics business unit and the parent company of the Sony Group, which is engaged in business through its four operating segments – electronics (including video games, network services and medical business), motion pictures, music and financial services. These make Sony one of the most comprehensive entertainment companies in the world.

Figure 3.3 – Beginning of the Wikipedia article for “Sony”, the colors corresponds to the topics in Figure 3.4 and Table 3.1

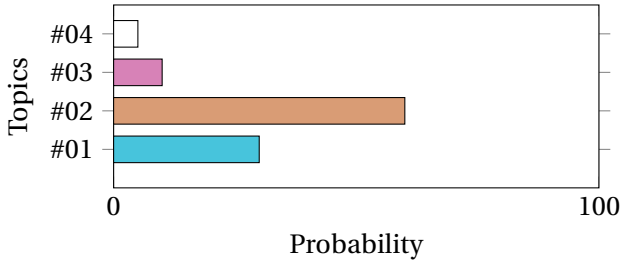


Figure 3.4 – Illustration of a part of the topic distribution for the document in Figure 3.3

Topic #01	Topic #02	Topic #03	...
electronic : 0.23	business : 0.43	japan : 0.21	
computer : 0.19	sony : 0.22	tokyo : 0.12	
entertainment : 0.18	samsung : 0.6	island : 0.5	...
game : 0.15	corporation : 0.2	minato : 0.1	
...	..	...	

Table 3.1 – Word distributions corresponding to some topics

More formally it consists of a probabilistic generative graphical model which assumes that each document contains a mixture of topics, which are sampled from a multinomial distribution (specific to each document). Each topic is associated to a multinomial word distribution. When a corpus is generated, it is assumed that each word of a document is chosen by first picking a topic according to the topic distribution for this document (Figure 3.4 illustrate a part of the distribution corresponding to the document in Figure 3.3), and then choosing a

word from the word distribution for this topic (examples of such distributions are showed in Table 3.4). Both topics and word distributions are smoothed by picking the multinomial distributions' parameters from a Dirichlet distribution. These priors are one of the main contribution of LDA over PLSA, and makes it better to handle unseen documents. Note that this model does not introduce any syntactic constraints on words, and represents documents as simple bags of words. Algorithm 4 describes this process in a more formal way and Figure 3.5 represents the corresponding plate diagram (given a corpus C consisting of D documents of length N, a set T topics, and a vocabulary V).

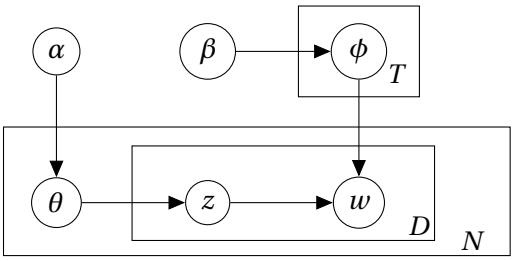


Figure 3.5 – Latent Dirichlet allocation as a plate diagram.

Algorithm 4 Latent Dirichlet Allocation

Choose the hyper−parameters $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, ...\alpha_{|T|}\}$ and $\boldsymbol{\beta} = \{\beta_1, \beta_2, ..., \beta_{|V|}\}$
For every document $d \in \{1, ..., D\}$:
  Choose $\theta_d \sim$ Dirichlet($\boldsymbol{\alpha}$)
For every topic $t \in \{1, ..., T\}$:
  Choose $\phi_t \sim$ Dirichlet($\boldsymbol{\beta}$)
For each document $d \in \{1, ..., D\}$:
  For each word position $i \in \{1, ..., N\}$:
    Pick a topic $z_{d,i} \sim$ Multinomial(θ_d)
    Pick a word $w_{d,i} \sim$ Multinomial($\alpha_{z_{d,i}}$)

We are interested in finding the distributions of parameters $\mathbf{Z}$, $\boldsymbol{\theta}$, $\boldsymbol{\phi}$ given the words $\mathbf{W}$ in the documents of a training corpus, and the hyperparameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.

However, we have that

$$P(\mathbf{Z}, \boldsymbol{\theta}, \boldsymbol{\phi} | \mathbf{W}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{P(\mathbf{Z}, \boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{W} | \boldsymbol{\alpha}, \boldsymbol{\beta})}{P(\mathbf{W} | \boldsymbol{\alpha}, \boldsymbol{\beta})} = \frac{P(\mathbf{Z}, \boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{W} | \boldsymbol{\alpha}, \boldsymbol{\beta})}{\iint \sum_{\mathbf{Z}} P(\mathbf{Z}, \boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{W} | \boldsymbol{\alpha}, \boldsymbol{\beta}) d\boldsymbol{\theta} d\boldsymbol{\phi}}$$

and the denominator of this expression is intractable due to the exponential number of terms it contains. An alternative popular solution is to marginalize the parameters of the multinomial distributions $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ and to sample the distribution of latent topics $P(\mathbf{Z}|\mathbf{W}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ using a

Markov chain Monte Carlo (MCMC) method called Gibbs Sampling [Gri02]. The idea behind this technique is that we can affect a random topic $z_{d,i}$ to each word in the corpus, and then iteratively re-sample each $z_{d,i}$ according to the conditional distribution $P(z_{d,i}|\mathbf{Z}_{-(d,i)}, \boldsymbol{\alpha}, \boldsymbol{\beta})$, where $\mathbf{Z}_{-(d,i)}$ denotes the topic assignment of all words in the corpus but $w_{d,i}$. Eventually, after iterating a large number of times and under the assumptions of our model, the samples will converge to the ones from the true $P(\mathbf{Z}|\mathbf{W}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ distribution. $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ can then be easily empirically estimated from the resulting word-topic affectations.

At every step, $z_{d,i}$ is sampled according to the following distribution [Gri02, Cha11] :

$$\begin{aligned} P(z_{d,i} = z|\mathbf{Z}_{-(d,i)}, \mathbf{W}, \boldsymbol{\alpha}, \boldsymbol{\beta}) &\propto P(w_{d,i}|z_{d,i} = z, \mathbf{Z}_{-(d,i)}, \mathbf{W}_{-(d,i)}, \boldsymbol{\beta}) \cdot P(z_{d,i} = z|\mathbf{Z}_{-(d,i)}, \boldsymbol{\alpha}) \\ &= \frac{n^{w_{d,i},z}_{-(d,i)} + \beta_{w_{d,i}}}{\sum_{w \in V}(n^{w,z}_{-(d,i)} + \beta_w)} \cdot \frac{m^{d,z}_{-(d,i)} + \alpha_z}{\sum_{z \in T}(m^{d,z}_{-(d,i)} + \alpha_z)} \end{aligned}$$

where $n^{w_{d,i},z}_{-(d,i)}$ is the number of times where topic z was assigned to the term of $w_{d,i}$, excluding the topic attribution for $w_{d,i}$. Similarly $m^{d,z}_{-(d,i)}$ is the number of time topic z was assigned in document d, excluding the attribution for $w_{d,i}$.

This process is repeated several times over all the words of all the documents. $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ can then be easily estimated :

$$\theta_{d,z} = \frac{m^{d,z} + \alpha_z}{\sum_{z' \in T} m^{d,z'} + \alpha_{z'}}$$

$$\phi_{z,w} = \frac{n^{w,z} + \beta_w}{\sum_{w' \in V} n^{w',z} + \beta_{w'}}$$

With $m^{d,z}$ the number of times topic d is used in document d, and $n^{w,z}$ the number of times word w is assigned to topic z in C.

These estimations are averaged over different samplings, ignoring the first sampling iterations, which have not converged yet. We used LDA and Gibbs sampling [Gri02] implementation of MALLET library [McC02].

3.2.2 LDA for fuzzy word clustering

Grzegorz Chrupała proposed a simple method in order to perform fuzzy clustering at a word level using LDA [Chr11]. We will use this is approach to cluster words, and we will give an overview of this method in this section. The assumption here is once again that the semantics of a word or a collocation mainly depends on its context (i.e. the other words and collocations located within a given window around it), which is also known as the distributional hypothesis [Har54]. This would allow us to use Latent Dirichlet Allocation as a way to perform fuzzy clustering at a word level.

In the standard LDA approach, a set of bag-of-words documents is used as input for the training. In our case, we will build these documents from the word types of our input text. For each word type in the input text, we will create a new *context document*, and add the words located within a window of size 2 around each of the occurrences of this word type in the input text to this *context document*. The set of all *context documents* constitutes the input for our topic modeling system. Therefore, each cluster corresponds to one topic in the LDA model and the membership value of one word type for this cluster is the probability $P(z|d)$ that cluster/topic z occurs given the word/context document d.

Let's consider an example. In the 50 first million words from the English Wikipedia corpus, the word "bass" occurs 12408 times. Let's look at some of its occurrences :

> ... tuba, timpani, snare drum, bass drum, triangle, wood block, cymbals, low and high ...

> ... philosophical system, philosopher Robert H. Bass has argued that her central ethical ...

> ... he is seen struggling to get a double bass onto a train (Strangers on a Train) or walking ...

> ... film poster designers such as Bill Gold and Saul Bass - and kept them busy with ...

> ... rhythm track with a drummer, guitarist and bass player. All the other arrangements ...

> ... Freshwater fish such as bream, shad, bass, and sucker are common. Along the Gulf ...

Using a context window of size 3, and ignoring stop-words, we can generate the following bag-of-words document :

> timpani, snare, drum, drum, triangle, wood, system, philosopher, Robert, argued, central, ethical, struggling, get, double, train, Strangers, Train, Bill, Gold, Saul, kept, busy, countless, track, drummer, guitarist, player, arrangements, vocals, fish, bream, shad, sucker, common, along

This will constitute the equivalent of one document in the standard LDA approach. We can build the same context document for every word type in the input, and get the distribution

of topics $P(z|d)$ for each word types using a collapsed Gibbs Sampler. Here is the obtained fuzzy clustering for our "bass" example, using a window of length 2, 100 clusters and the first 50 millions words from the English Wikipedia.

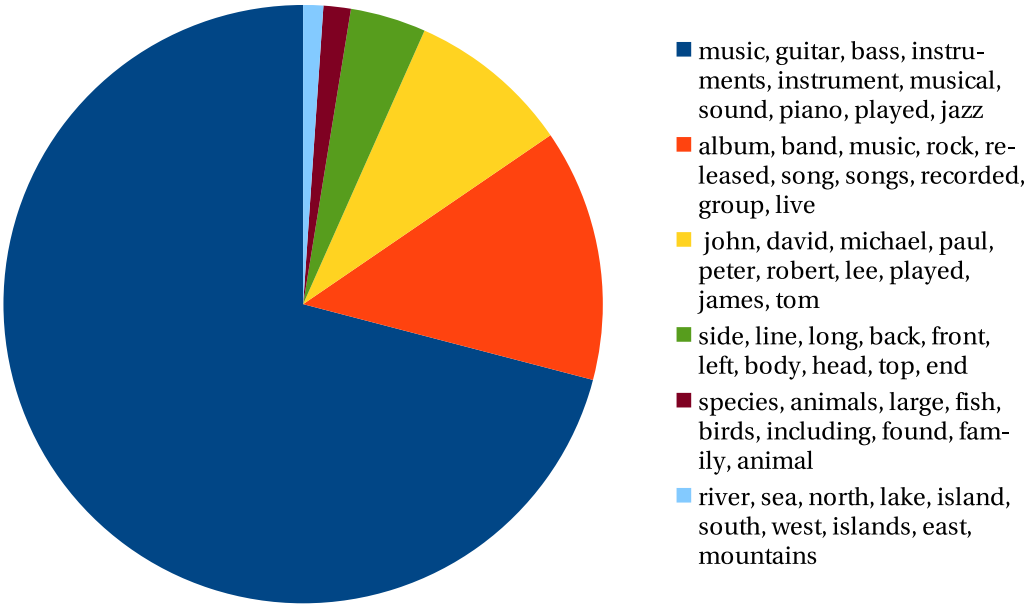
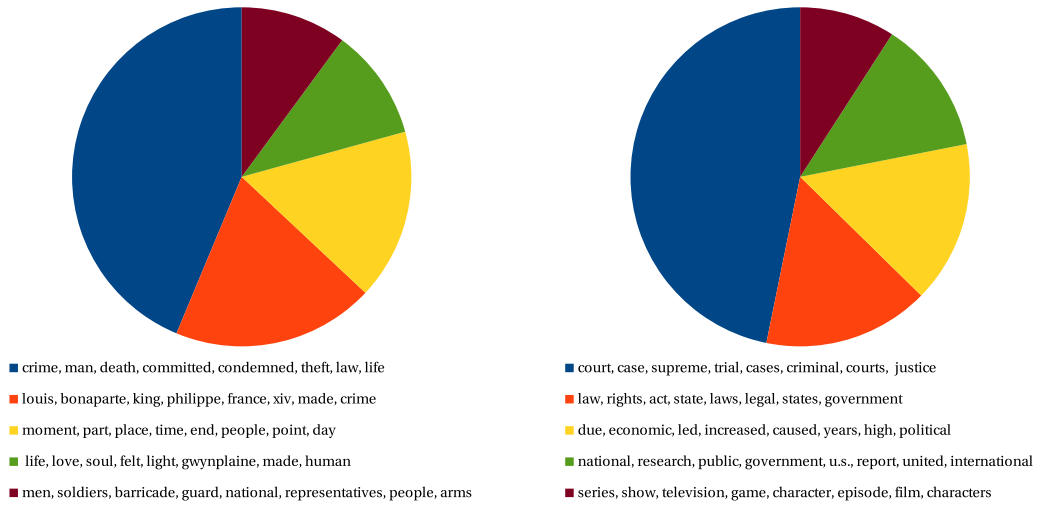


Figure 3.6 – Fuzzy clustering of "bass", top-6 clustered are represented according to their membership values

We can clearly see in Figure 3.6 that the 3 main different senses of the word "bass" in our example occurrences are correctly modeled by the clusters; the first and second clusters correspond to the "bass" modifier applied to a music instrument (i.e. *bass,* $noun_1$ entry of the Oxford Online dictionary [Ste10], "Denoting the member of a family of instruments that is the lowest in pitch" for the first cluster, and "The low-frequency output of a radio or audio system, corresponding to the bass in music." for the second cluster). The third cluster corresponds to the proper noun "Bass". Finally the fifth and sixth clusters correspond to the bass fish meaning (*bass,* $noun_2$ "The common European freshwater perch").

Note that the clustering is highly dependent on the input text. As an illustration consider the clustering of the word "crime" using 3 million words from English the translation of Victor Hugo novels against the same task using the first 50 million words from Wikipedia. Figure 3.7 shows that both clustering outputs indeed share some common characteristics (the government seems to be present in one cluster in both cases), but we can see that most of the clusters are highly influenced by the nature of the input (love and war are some of the fundamental topics in Hugo's novels, whereas economy and laws are the kind of topics described in Wikipedia).



(a) Victor Hugo novels

(b) Wikipedia

Figure 3.7 – Comparison of the clustering of the word “crime” in Victor Hugo novels, and in Wikipedia

3.2.3 Collocation clustering with LDA

Our goal is not only to cluster unigrams, but to also take n-grams with $n > 1$ into account. Nevertheless, we are not interested in any n-grams, since most of them would not bring any useful semantic information to our clustering output. Only collocations, as defined in the previous chapter, could add semantic information, since the actual meaning of them cannot be derived from the unigrams it contains.

We extend the method used for words to collocations, by building a context bag-of-words containing the words which occur within a window of length 2 around each occurrence of the collocation. During our first experiments, these documents were simply added to the training documents corresponding to unigrams, and the Gibbs sampling procedure was performed on the whole set. However, we quickly realized that including the context documents corresponding to collocations in addition to the ones for unigrams in the initial training of the model would bias the results, since it would bias the word distribution in context documents (words around collocations will appear much more often than they actually occur) which will affect the counts n and m in the Gibbs sampling procedure (Section 3.2.1) [Gri02]. Therefore we stick to the approach of Chrupala [Chr11] and consider only unigrams during the training of our model.

The task of estimating the clusters for a new collocation (which corresponds to an unseen context document, as described in the previous section), is in fact the same as inferring the topics of a new document with LDA. We perform this task by initializing topics to be the most probable for each token in the context document of the collocation, and by running

100 iterations of a Gibbs sampler (with a burn-in period of 10 iterations, and saving samples every 10 iterations) to get the true topic distribution according to the previously trained model, using the *TopicInferencer* class from MALLET [McC02]. This way the initial word clustering model can easily be reused at any time to cluster additional collocations.

3.3 Implementation

The system we implemented consists of several components; a Brown clustering tool (using [Lia12]), an LDA based clustering tool for words and collocations (using [McC02]) together with some tools to manage the output of this system, as well as some taggers to annotate files using the output of these systems. All these components can manage several input formats (IOB, one sentence per line, etc.). Automatic testing and parameter choice was performed using Python scripts.

3.3.1 Brown clustering

Brown Clustering was performed on the first 50 millions words from the Polyglot tokenized English Wikipedia Corpus [ARPS13]. The C++ implementation of Percy Lang [Lia12] was used. Because of the hierarchical nature of the Brown Clustering technique, it is interesting to consider different prefix lengths, which will provide different abstraction classes. After several experiments, we chose to focus on class prefixes of length 4, 8 and 12 for 100, 200, 300, 500, 750 and 1000 word classes.

The tool is implemented in Java, and includes a tagger to annotate IOB files with the prefixes of the classes corresponding to the words.

3.3.2 LDA based clustering

As for Brown Clustering, we performed the LDA based clustering on a text consisting of the first 50 millions words from the Polyglot tokenized English Wikipedia Corpus [ARPS13], from which we removed the stop-words. The MALLET library was used for both the training (*ParallelTopicModel*) and the prediction (*TopicInferencer*) tasks [McC02]. Hyper-parameter choice and optimization is performed by MALLET internal procedures. Since the LDA clustering is not deterministic, we run 5 different trainings for 100, 200, 300, 500, 750 and 1000 clusters each (35 in total). For most of the following results, we will consider the average ($\overline{x} = \frac{1}{5}\sum_{i=1}^{5} x_i$) over the 5 models, and estimate the corresponding standard deviation ($s_N = \sqrt{\frac{1}{5}\sum_{i=1}^{5}(x_i - \overline{x})^2}$) We used a window size of 2 words on the left and right context, since we noticed that increasing this number is not improving the results but makes the Gibbs sampling procedure much slower (due to the additional amount of data). The number of Gibbs sampling iterations was set to 1000 for the training phase.

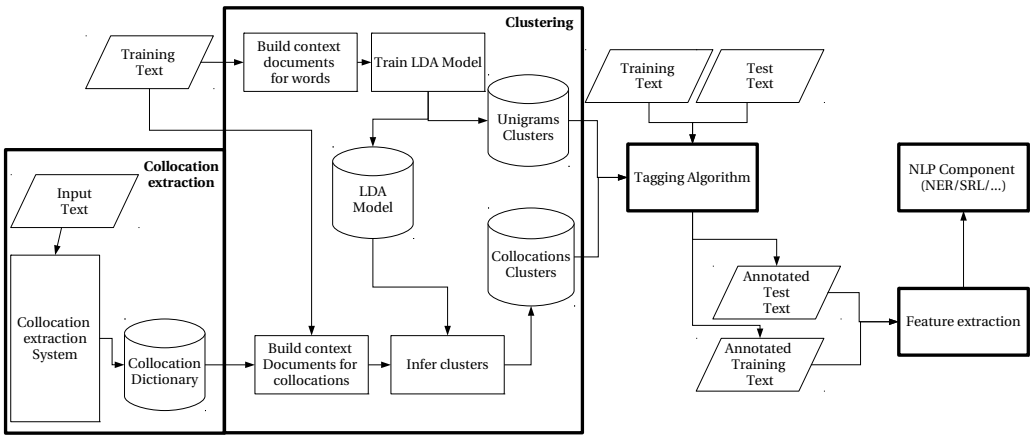


Figure 3.8 – Fuzzy Clustering Architecture

Figure 3.8 is showing the architecture of the fuzzy clustering component and its usage together with a NLP component. The first step, illustrated at the bottom left of the Figure 3.8 and at the top of Figure 1.3, is to build a dictionary of collocations, using our component described in Chapter 2. Secondly, it is necessary to build the fuzzy clusters, by first training LDA on the context of the words in a large training set. Once we have this LDA model, we find the collocations in the training set using the previously built collocation dictionary, and we infer their fuzzy clusters using the technique described in the previous section. We now have a large dictionary of clusters for words and collocations that we will use to annotate the training and test files used by several NLP components.

We implemented two tagging algorithms. First, a non-overlapping one which gives priority to longer matches in the dictionary. In this way, we can annotate each word with its clusters, or the ones of the longest collocation it is part of. Consider the actual output from the tagging algorithm shown in Figure 3.9, each word was annotated with the index of the top-3 clusters from the LDA fuzzy clustering, but notice how “high” and “school” are assigned to the same clusters, this is because “high school” was part of the collocation dictionary, for this reason the clusters of “high school” (197 161 124) were preferred over the respective clusters of high “130 197 155” and school “197 161 263”.

```
a -1 -1 -1
referee 90 9 198
was -1 -1 -1
expelled 83 140 146
from -1 -1 -1
high 197 161 124
school 197 161 124
banned 15 203 212
Thursday 98 141 169
from -1 -1 -1
competition 244 274 34
for -1 -1 -1
one -1 -1 -1
year 98 195 152
```

Figure 3.9 – Cluster tagging example (an actual training or test file will contain other annotations)

Secondly, we also implemented an alternative tagging algorithm, which assigns the word clusters to every words and, in addition, the collocation clusters when the word belongs to a collocation. The corresponding output can be seen in Figure 3.10.

```
a -1 -1 -1 -1 -1 -1
referee 90 9 198 -1 -1 -1
was -1 -1 -1 -1 -1 -1
expelled 83 140 146 -1 -1 -1
from -1 -1 -1 -1 -1 -1
high 130 197 155 197 161 124
school 197 161 263 197 161 124
banned 15 203 212 -1 -1 -1
Thursday 98 141 169 -1 -1 -1
from -1 -1 -1 -1 -1 -1
competition 244 274 34 -1 -1 -1
for -1 -1 -1 -1 -1 -1
one -1 -1 -1 -1 -1 -1
year 98 195 152 -1 -1 -1
```

Figure 3.10 – Alternative cluster tagging example (an actual training or test file will contain other annotations)

The last step is to build features from these clusters which could improve several NLP components.

The tool is implemented in Java, and supports various options, including the number of clusters, the number of iterations of the Gibbs sampler, the number of threads to use, the window size, the stop-words dictionary path, the format and file of the input text, and the format of the output (binary model file, text file containing the clusters for each word (membership values) in plain text). During the first phase, context documents are built using a big HashMap for every token, filtering stop-words and unwanted characters. Secondly, MALLET ParallelTopicModel is trained on these data (by creating an InstanceList from the HashMap), and the cluster membership values are eventually retrieved (getTopicProbabilities).

The collocation clustering module takes as input an existing clustering model (binary or text file), a collocation dictionary (text file containing one collocation per line), the format and path of the input text, the length of n-grams to consider and the output format. The MALLET TopicInferencer is used for this task. The new context documents corresponding to collocations are built in the same way and using the same Pipe than during the word clustering phase. New membership values are retrieved using the getSampledDistribution method.

Finally the annotating tool takes as input the outputs of the clustering algorithms for words and collocations, an IOB text file to annotate (together with the desired position of the new columns), and the desired tagging method.

3.3.3 Performance Comparison

The computation time necessary to cluster unigrams from the 50 million words Wikipedia corpus [ARPS13] is shown in Figure 3.11. Both the Brown clustering implementation [Lia12] and the LDA clustering implementation [McC02] are parallelized, and we ran them using 16 threads (on as many CPU cores) and a sufficient amount of RAM to store all the required data in memory (>100GB). The absolute values on Figure 3.11 are not significant since other tasks were running on the server when we ran the experiments, however, we can clearly see that the LDA clustering time grows linearly with the number of clusters, while for Brown clustering it grows exponentially with the number of clusters. As we will discuss in the following chapters, we notice that to achieve the best respective performance of the two methods (i.e. 200 clusters for LDA, 750 clusters for Brown), the LDA approach is taking significantly less time.

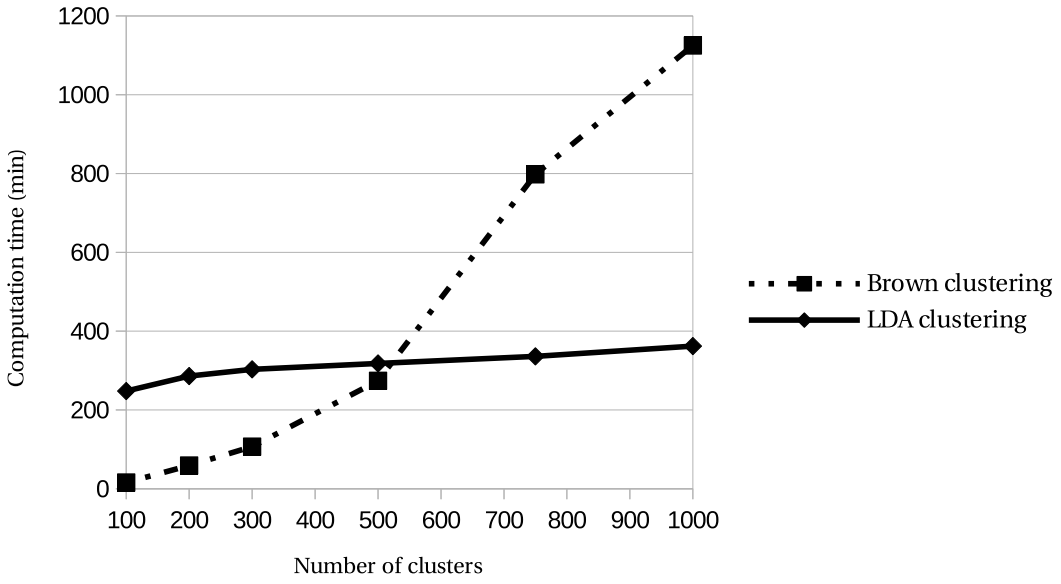


Figure 3.11 – Brown and LDA clustering computation time

3.4 On naming clusters

Connecting the automatically generated clusters to human-made hierarchies or classes is an interesting problem. More specifically, we discussed earlier the narrow relation between clusters and semantic meaning; it should therefore be possible to map each cluster to its corresponding sense. It is not a trivial problem since most clusters do not contain any word which is able to describe it accurately. Take as an example the cluster containing the top-words "red,green,blue,purple,pink". It is obvious that such a cluster should be named "colors", but this information cannot be extracted from the content of the cluster. One could think of a supervised machine learning technique, trained on a large set of manually annotated clusters, but such resources are not available as far as we know. Nevertheless, extensive work has already been done on building resources containing high quality dictionaries annotated with semantic relations, the most famous one being Wordnet, that we introduced in the previous chapter.

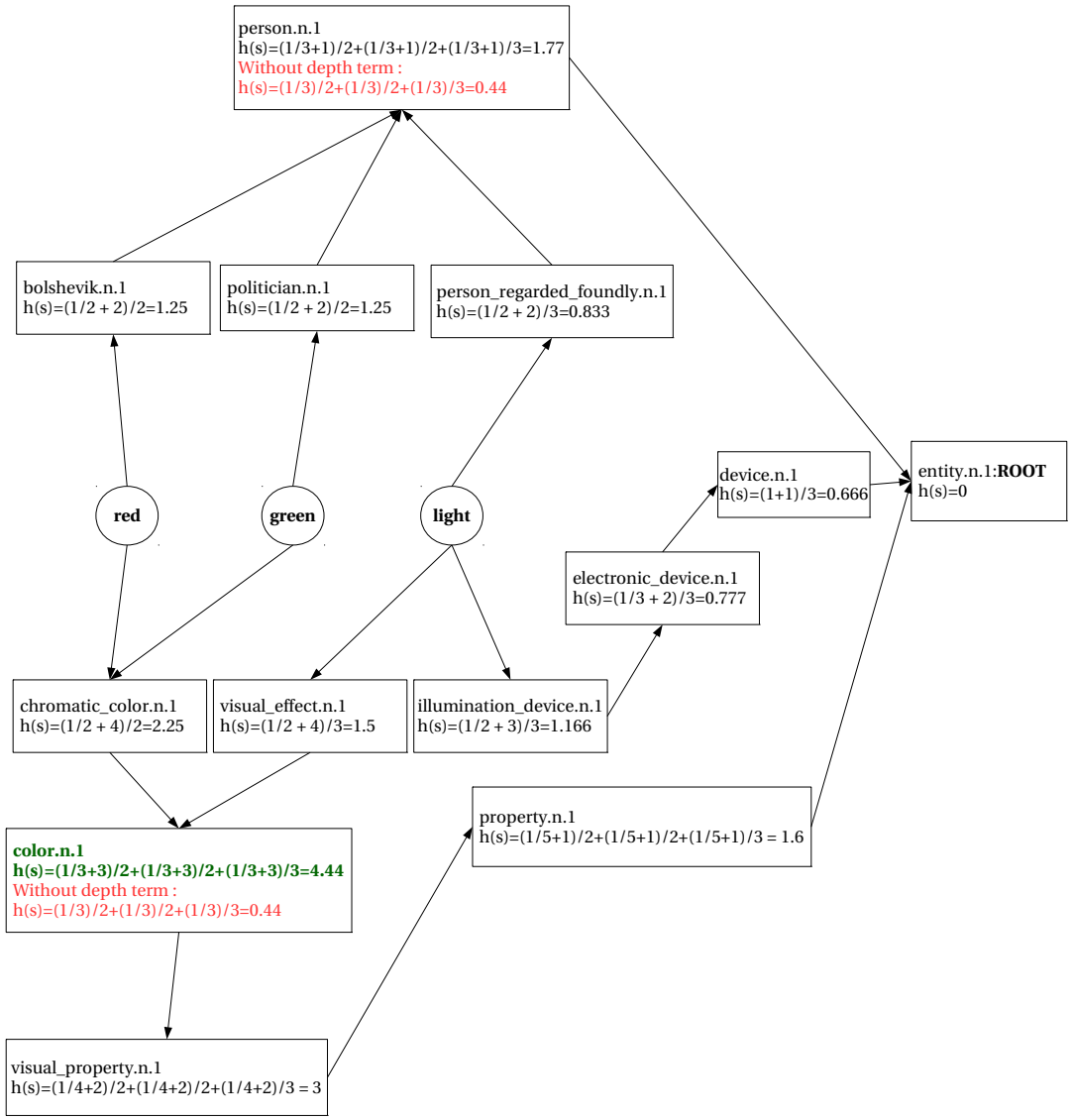


Figure 3.12 – Cluster naming procedure illustrated. For the sake of simplicity, only a subgraph of the actual Wordnet hypernym graph is considered. In red, the score is computed without the depth term, in order to emphasize its role.

Wordnet includes several semantic relations on synsets (sets of synonyms), but here we are mainly interested in hierarchical semantic relations. We believe that a good name for a cluster would ideally correspond to the name of the lowest common hypernym synset of the top-words in this cluster. This is for example the case for the “january,february,march,april” cluster, where each word shares the “georgian_calendar_month.n.1” synset as lowest common

hypernym. Nevertheless, such an approach has several issues in practice. First the clustering output is not always perfect, and it sometimes happens that one of the top-words is out of context. Also all the existing relations are not guaranteed to be represented in Wordnet, and finally, one word can have several unrelated syntactic sets in wordnet (ex : "green" golf/color), and some disambiguation based on the other words in the cluster would be needed. For these reasons, simply looking for the lowest common hypernym of all the synsets of the top-words will in practice always lead to a very generic and meaningless synset, way too high in the hierarchy.

Our solution consists in evaluating a heuristic function h for every synset node in the graph consisting of the union of the transitive closure graphs of hypernyms synsets of the most common words in the cluster. The node with the maximum score h is chosen as a name for the cluster.

$$h(s) = \sum_{w \in \mathbf{W}} \frac{\gamma_w}{||\mathbf{synsets}(w)||} \sum_{s' \in \mathbf{synsets}(w)} f(s, s')$$

and

$$f(s, s') = \begin{cases} \alpha * path\_similarity(s, s') + depth(s)^{\beta} & \text{if } distance(s, s') \leq n \\ 0 & \text{if } distance(s, s') > n \end{cases}$$

where $path\_similarity(s, s') = \frac{1}{shortest\_path\_distance(s,s')+1}$, $depth(s)$ is the longest distance from s to the global root of the graph of hypernyms (entity.n.1), and γ_w is a weight corresponding to the importance of the word w in the cluster. Parameters α, β, and n have to be chosen carefully according to the topology of the hypernym graph. The $\alpha * path\_similarity(s, s')$ term is here to ensure that the chosen synsets share a strong similarity with most of the most common words in the cluster. The second term $depth(s)^{\beta}$ is used to avoid too generic names; the deeper a synset lies (i.e. the more distant from the root), the higher will be his score. The score is normalized in such a way that each word contributes identically to the total score, independently from the number of synsets it has.

Graph 3.12 shows the evaluation of the heuristic in a subgraph from wordnet hypernym relations for the words "red", "green", "light", with parameters $\alpha = 1$, $\beta = 1$ and $\boldsymbol{\gamma} = \mathbf{1}$. We can see that the best synset is "color.n.1", which obviously fit the cluster very well. The red scores are computed without the depth term, in this case the synset "person.n.1" would obtain the same score as "color.n.1"; here the depth term is to give preference to more specific synsets which lie deeper in the graph.

We experimentally chose $\alpha = 10$, $\beta = 2$ and $\boldsymbol{\gamma} = \mathbf{1}$ to obtain the results shown in Table 3.2, using actual clusters from the LDA based method. These parameter values seem to fit the topology of hypernym graphs in Wordnet correctly. Most of the synsets are relevant, but cluster 9 is an illustration of what cannot be handled by our method; when using a large number of clusters (here, 1000) on the Wikipedia dataset, it sometimes happens that one cluster corresponds to a historical event, in this case, hypernym relationships do not help to choose a good name,

since the words might not share any sematical properties. Similarly, cluster 10 does not get a relevant synset since Wordnet lacks the kind of relationship which links different proper names together.

	Top-5 words of the cluster	**Suggested Synset**
1	economy, industry, trade, sector, market	commerce.n.01
2	sky, sun, blue, mountain, cloud	atmosphere.n.05
3	january, february, march, april, may	gregorian_calendar_month.n.01
4	aircraft, air, fighter, force, flight	craft.n.02
5	day, year, calendar, time, month	time_period.n.01
6	red, green, blue, purple, yellow	chromatic_color.n.01
7	ottoman, empire, turkish, sultan, rule	turki.n.01
8	shoes,pants,shirt,cap,gloves	clothing.n.01
9	attacks, september, 2001, terrorist, godzilla	radical.n.03
10	bob,tom,joe,alice,lewis	german.n.01

Table 3.2 – Example clusters and generated synset-based name

4 Application to Named Entity Recognition

In this chapter we will use the semantic clustering information of words and collocations described in the previous chapters to build features which can improve the performance of a Named Entity Recognition (NER) component. After explaining the underlying logic of the classifier and presenting a baseline, we will evaluate the effect of new features based on Brown clustering and LDA based clustering of words. Finally, we will describe several methods to include features based on collocation clusters.

4.1 Linear-chain Conditional Random Fields

Conditional Random Fields are a discriminative undirected graphical model introduced in [LMP01]. It consists of a log-linear model which has several interesting properties which makes it popular for tagging problems compared to its closest alternatives (HMMs which cannot integrate complex features since such a generative graphical model becomes intractable with too many dependencies, and Maximum Entropy Markov Models (MEMMs) which suffer from the label bias issue [LMP01]). We will not explain this model in detail, but will only provide the intuition behind it. More explanations can be found in [SM11].

Consider the NER task introduced in the introduction, in this case a linear-chain CRF is well suited. It consists of a special case of CRF where the graph has a chain structure (i.e. one label is mapped to each observed word, and each label depends on the previous one). Linear chain CRFs can be seen as a classic multinomial log-linear classifier (Maxent) over an entire sequence. As any log-linear model, features are a main concept in CRFs; in order to find Named Entities (NEs), we are not only interested in the word token itself, but several other properties of it and its neighborhood could be of interest (capitalization, POS tags, etc.).

Let $\mathbf{X}$ and $\mathbf{Y}$ be two random variables. Given a sequence $\mathbf{y} = y_1 y_2 ... y_n$ of labels (i.e. named entity tags in a sentence), and a vector $\mathbf{x} = x_1 x_2 ... x_n$ which corresponds to the sequence of words, we define m functions $f_1, f_2 ... f_m$ called feature functions. Each of them maps the parameters $y_i, y_{i-1}, \mathbf{x}, i$ to a real number, which in our case will be only 0 or 1.

We could have for example that $f_1(y_i, y_{i-1}, \mathbf{x}, i) = 1$ if x_{i-1} starts with a capital letter, $f_1(y_i, y_{i-1}, \mathbf{x}, i) = 0$ otherwise, which would correspond to the capitalization feature described below. If the value of a feature is not binary, we generate one binary feature function for each possible value, in order to obtain a one-hot representation of the desired feature (e.g. in the case of POS tags, we will have $f_i(y_i, y_{i-1}, \mathbf{x}, i) = 1$ if x_i is a *noun* (we consider the POS tag as part of the x word itself), 0 otherwise, $f_{i+1}(y_i, y_{i-1}, \mathbf{x}, i) = 1$ if x_i is an *adjective* (we consider the POS tag as part of the x word itself), 0 otherwise, etc.).

For a given word, all the features form a feature vector $\mathbf{f}$ of dimension m (the last feature of the vector is set to 1 in order to incorporate a bias term implicitly). Given a weight vector $\boldsymbol{\lambda}$ of dimension m, we can compute a score for a sequence (i.e. a sentence), by summing the dot product of $\mathbf{f}$ and $\boldsymbol{\lambda}$ for all the n words in the sentence. By exponentiating and normalizing over all possible label sequences (applying the softmax function) in order to obtain a proper probability value, we get :

$$P(\mathbf{y}|\mathbf{x};\boldsymbol{\lambda}) = \frac{\exp(\sum_{i=1}^{n}\sum_{j=1}^{m}\lambda_j f_j(y_i, y_{i-1}, \mathbf{x}, i))}{\sum_{\mathbf{y}}\exp(\sum_{i=1}^{n}\sum_{j=1}^{m}\lambda_j f_j(y_i, y_{i-1}, \mathbf{x}, i))}$$

A L2 regularization term (not described here) is often used to smoothen the model. Our goal is therefore to find the weights which maximize the likelihood of this model, which is done using a variant of the gradient ascent algorithm (Limited-Memory Broyden-Fletcher-Goldfarb-Shanno (LM-BFGS) [LN89]) on our training set. The NER system we are using is based on the CRF++ implementation [Kud07] of Conditional Random Fields.

4.2 NER Component

4.2.1 Baseline

The baseline is the unchanged, original method implemented in our component. The set of features it is using is the combination of the ones shown in Table 4.1 (simple unigram features), Table 4.2 (unigram feature combinations) and Table 4.3 (bi-gram features).

Feature number	Feature Name	Example Value
1	Token	title
2	Token of the previous word n-1	Cup
3	Token of word n+2	a
4	Part-of-Speech tag	NN
5	POS tag of word n-2	JJ
6	POS tag of previous word n-1	NNP
7	POS tag of next word n+1	IN
8	Capitalization	a
9	Capitalization of word n-2	A
10	Capitalization of word n-1	A
11	Capitalization of word n+1	a
12	Capitalization of word n+2	a
13	Case	No_upper
14	Contains number	Has_no_N
15	Word/Integer	Word
16	Word/Integer of word n+1	Word

Table 4.1 – Single NER baseline features

Number	Feature Name	Example Value
17	POS tag of word n+1 / POS tag of word n+2	IN / DT
18	POS tag / Capitalization	NN / a
19	Token / POS tag	title / NN
20	Token / Capitalization	title / a
21	Token / Contains number	title / Has_no_N
22	Token / Case	title / No_upper
23	Case of word n / Case of word n-1	No_upper / No_upper
24	POS tag of word n / POS tag of word n+1	NN / IN
25	POS tag of word n-1 / POS tag of word n / POS tag of word n+1	NNP / NN /IN
26	POS tag of word / Capitalization / Contains number	NN / a / Has_no_N
27	Token n-1 / POS tag of word n-1 / Capitalization of word n	Cup / NNP / a

Table 4.2 – NER baseline features combinations

Feature number	Feature Name	Example Value
1	Token	Cup/title
2	Token of the previous word n-1	Asian/Cup
4	Part-of-Speech tag	NNP/NN
5	POS tag of word n-2	PRP$/JJ
6	POS tag of previous word n-1	JJ/NNP
7	POS tag of next word n+1	NN/IN
8	Capitalization	A/a
9	Capitalization of word n-2	a/A
10	Capitalization of word n-1	A/A
11	Capitalization of word n+1	a/a
12	Capitalization of word n+2	a/a
13	Case	No_upper/No_upper
14	Contains number	Has_no_N/Has_no_N
15	Word/Integer	Word/Word
16	Word/Integer of word n+1	Word/Word
17	Token / Previous Token	title / Cup

Table 4.3 – NER baseline bigram features

4.2.2 Evaluation Metrics

We redefined precision and recall on page 15. We will also use them to evaluate this NER component. In this context we use the following definitions :

True positives (tp)
The number of annotated NE correctly classified as NE.

False positives (fp)
The number of misclassified NE (i.e. assigned to the wrong class, or assigned to a NE class when it is actually not labeled as a NE)

True negatives (tn)
The number of unclassified NEs which are actually not annotated NE.

False negatives (fn)
The number unclassified or misclassified NEs which are actual NEs.

Therefore, precision is the proportion of correctly classified named entities (i.e. which correspond to the annotation) among all classified NEs. Recall is the proportion of correctly classified NEs among all the actual annotated NEs. In addition to these measures, we will use the F1-score which combines both precision and recall according to the following formula:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

as well as the accuracy :

$$\text{accuracy} = \frac{\text{tp} + \text{tn}}{\text{tp} + \text{fp} + \text{fn} + \text{tn}}$$

We evaluate the system on the Conference on Natural Language Learning (CoNLL) 2003 NER Corpus, which contains 4 different NEs classes, *LOC* for locations, *PER* for persons, *ORG* for organization, and *MISC* for miscellaneous entities. The performance of the baseline using our CRF system and the features described in the previous section is shown in Table 4.4.

NE Type	**Precision**	**Recall**	**F1-Score**	**Accuracy**
Overall	84.39	83.07	83.73	96.67
LOC	87.97	89.03	88.50	
MISC	77.58	72.93	75.18	
ORG	81.95	75.98	78.85	
PER	85.71	88.62	87.14	

Table 4.4 – NER baseline performances

4.3 Results

We improved the system by taking all features used in the baseline, and adding some new ones based on our semantic clustering. Feature selection was performed in two steps, first by finding an optimal unigram feature set, and then by selecting additional bi-gram features. In both cases we started with a large set of cluster features, and performed feature ablation testing by recursively pruning some features according to their impact on the F1-score. Given the high number of free parameters in the overall system (number of clusters, context window size, collocation dictionary used, etc.), and the consequent training time (a few hours), several assumptions had to be made to limit the number of feature sets to evaluate. Especially we did not consider features involving words within a window larger than 1 around the current word, and we considered that only the top-3 best semantic clusters of a word could bring useful information to the system.

4.3.1 Brown Clustering

Using the brown clustering technique described in Chapter 3, many feature combinations were tested. Some of the best results were achieved with the ones shown in Table 4.5 and Table 4.6. These results are presented in Figure 4.1 and in more detail in Table 4.7 and 4.8. We can observe a 2.7% F-score improvement over the baseline, as well as 2.75 and 2.66 additional points for

recall and precision using 750 clusters and the feature set B (Table 4.6). It also appears that a high number of clusters (e.g. 750) is clearly improving the performance. Nevertheless, we noticed that once the minimal set of features from Table 4.5 is included, additional features did not bring any significant improvement, since the difference can only be visible with a low number of clusters. Although our two systems and features are different, our results here are similar to the ones obtained by Grzegorz Chrupala when it comes to the improvement induced by Brown clusters.

Feature number	Feature Name
1	cluster prefix of length 4
2	cluster prefix of length 8
3	cluster prefix of length 12
4	cluster prefix of length 8 for the previous word
5	cluster prefix of length 8 for the next word

Table 4.5 – NER brown features A

Feature number	Feature Name
1	cluster prefix of length 4
2	cluster prefix of length 8
3	cluster prefix of length 12
4	cluster prefix of length 12 for the previous word
5	cluster prefix of length 12 for the next word
6	Token / cluster prefix of length 12
7	Token n-1 / cluster prefix of length 12 of word n-1
8	Token n+1 / cluster prefix of length 12 of word n+1
9	Token / cluster prefix of length 12, 8
10	Token n+1 / cluster prefix of length 12, 8 of word n+1
11	Token n-1 / cluster prefix of length 12, 8 of word n-1
12	Token / cluster prefix of length 12, 8 , 4

Table 4.6 – NER brown features B

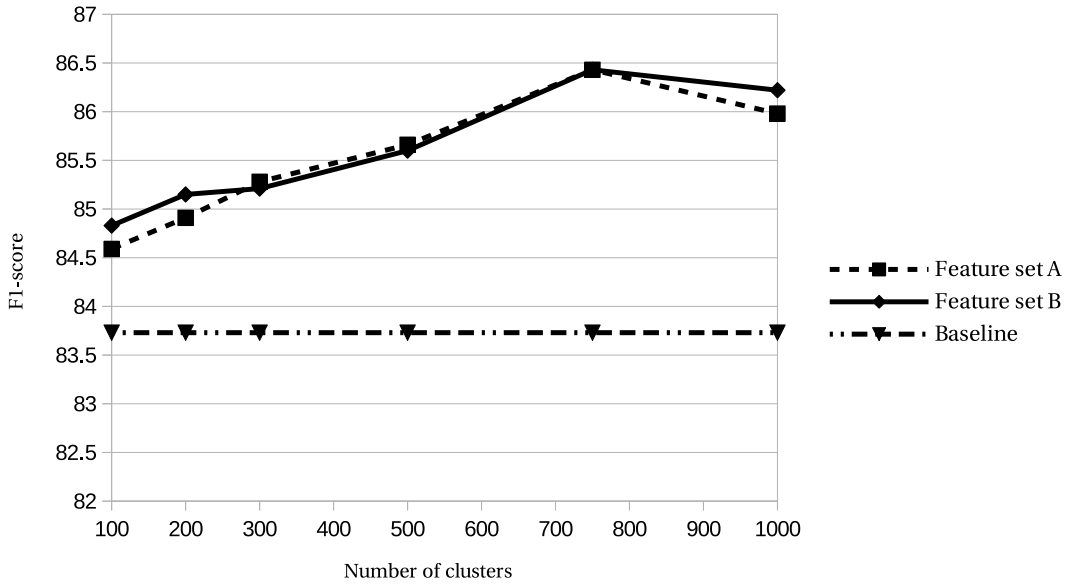


Figure 4.1 – F1-score results for NER with Brown Clustering features

NE Type	Precision	Recall	F1-Score	Accuracy
Overall	87.08	85.78	86.43	97.06
LOC	89.75	89.81	89.78	
MISC	79.11	76.07	75.56	
ORG	82.95	79.95	81.42	
PER	91.72	91.84	91.78	

Table 4.7 – NER brown clustering detailed results using feature set A and 750 clusters

NE Type	Precision	Recall	F1-Score	Accuracy
Overall	87.05	85.82	86.43	97.07
LOC	89.95	90.17	90.06	
MISC	78.47	75.78	77.10	
ORG	83.07	79.77	81.39	
PER	91.56	91.90	91.73	

Table 4.8 – NER brown clustering detailed results using feature set B and 750 clusters

We did not use collocation extraction as a preprocessing step when we use the Brown clustering technique, since Brown clustering already tends to assign the same clusters to n-grams which have a high point-wise mutual information (see Chapter 3).

4.3.2 Fuzzy Clustering

Unigrams

We started by building features from the LDA clusters of unigrams only, using the technique described in Chapter 3. Many experiments were performed, and we will present only the most relevant ones here. We had to deal with the constraint that the CRF implementation (CRF++ [Kud07]) cannot handle continuous features. So instead of using continuous membership values (i.e. probabilities of topics given a document) as a feature, we find the indices of the top-k most probable topics and use them to build discrete features. Among the many features combinations tested, we report here results for the ones shown in Table 4.9 and 4.10 (used in addition to the features of the baseline).

Figure 4.7 is showing the F1-score results for both feature sets (for various numbers of clusters), averaged over 5 LDA training procedures (since it is non deterministic) as well as a combination of the best LDA results among these 5 runs. The standard deviation for the results shown in the graph can be found in Table 4.11. Our best result is achieved using Feature set B and 300 clusters, as shown in Figure 4.12.

Feature number	Feature Name
1	Index of the top cluster
2	Index of the second best cluster
3	Index of the third best cluster
4	Index of the best cluster of the next word
5	Index of the best cluster of the previous word

Table 4.9 – NER clustering feature set A

Feature number	Feature Name
1	Index of the top cluster
2	Index of the second best cluster
3	Index of the third best cluster
4	Index of the best cluster of the next word
5	Index of the best cluster of the previous word
6	Token / Index of the best cluster
7	Token n-1 / Index of the best cluster of word n-1
8	Token n+1 / Index of the best cluster of word n+1
9	Token / Index of the top-1 and -2 clusters
10	Token n+1 / Index of the top-1 and -2 clusters of word n+1
11	Token n-1 / Index of the top-1 and -2 clusters of word n-1
12	Token / Index of the top-1 and -2, -3 clusters

Table 4.10 – NER clustering feature set B

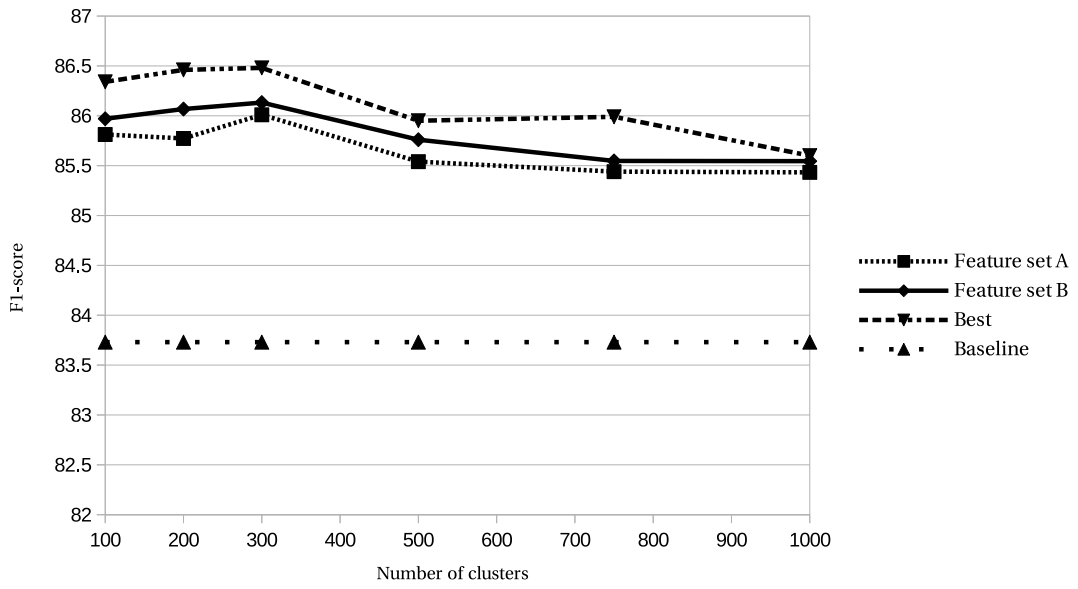


Figure 4.2 – F1-score results for NER with LDA clustering features

4.3.3 Fuzzy Collocation clustering

We investigated whether clustering collocations as single entities would improve the overall results of the NER.

Collocation extraction was performed using our system described in Chapter 2. Choosing

Number of clusters	Feature set A	Feature set B
100	0.355	0.264
200	0.362	0.372
300	0.429	0.390
500	0.096	0.164
750	0.305	0.200
1000	0.079	0.067

Table 4.11 – Standard deviation of the NER F1-Score with LDA clusters

NE Type	**Precision**	**Recall**	**F1-Score**	**Accuracy**
Overall	87.01	85.96	86.48	97.22
LOC	89.32	90.71	90.01	
MISC	81.05	74.93	77.87	
ORG	82.85	81.16	82.00	
PER	91.18	90.79	90.98	

Table 4.12 – NER best LDA clusters performance, for 300 clusters, using feature set B

the parameters when building the collocation dictionary was a critical task, since it can be influenced by the nature of the input data, the syntactic filters, the desired length of collocations (bi-grams, tri-grams, etc.), the association measure used to identify collocations, and the threshold which defines which percentage of the best n-grams (according to the association measure) to consider as a collocation.

We considered two different input corpora, a small one consisting of about 5 million words from Wikipedia, and a larger one consisting of about 10 million words from Wikipedia (the two corpora do not overlap). The collocations were extracted using the log-frequency biased pointwise mutual dependency measure, which performed best in our collocation extraction system evaluation, using no syntactic filters.

For both input sets we successively kept the best 0.5%, 4% and 10% collocations extracted with a highest score when building the collocation dictionaries. In both cases, using only 0.5% of the collocations did not significantly affect the results. We will therefore present here the results corresponding to the top-4% collocations (about 300 000 and 600 000 collocations for the small and large sets) and an even larger set consisting of the top 10% collocations on the 10 million word input set, resulting in a dictionary containing 1.6 million collocations. For all these experiments, we considered bi-gram collocations alone, and bi-gram collocations together with tri-grams ones, giving the priority to tri-gram collocations during the tagging procedure. Figure 4.3 and 4.4 give an idea of the corresponding number of collocations in the training and test sets for these parameters. For example using the top-4% collocations from the 10M words Wikipedia corpus, we find about 1400 bi-gram collocations in the test set, for about 45000 words in total, which means that 6% of the words are tagged as part of a bi-gram

collocation in this case. Finally, the following experiments are not averaged (single run) and run only using the 200 and 300 clusters models, since these values lead to the best results.

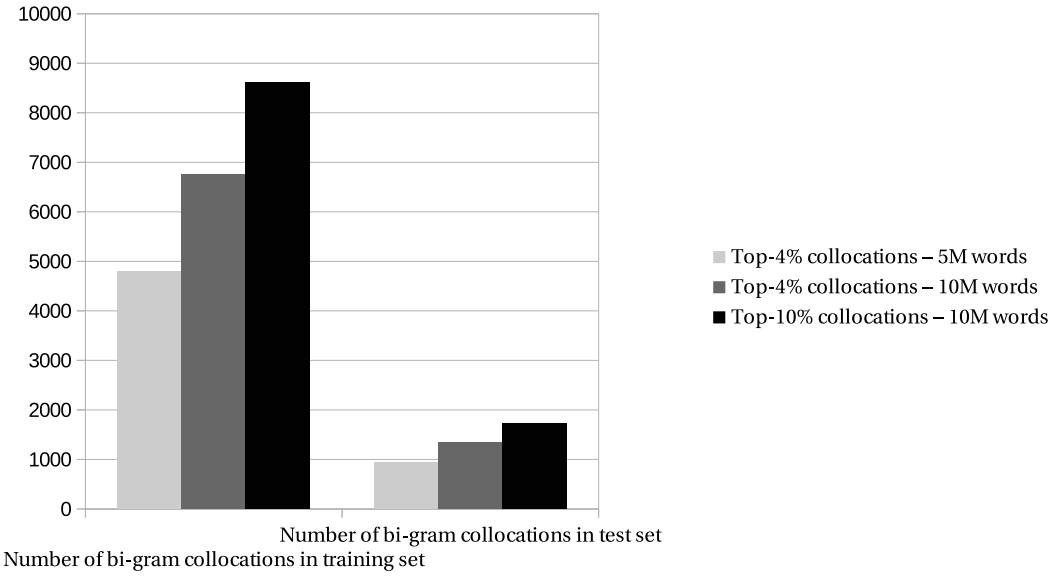


Figure 4.3 – Number of collocations found in the training set containing about 200 000 tokens in total and the test set containing about 45000 tokens in total

Longest collocation tagging

For our first experiment, we followed the same procedure as the one we described for unigrams in the previous section, but we used a non-overlapping tagging algorithm which gives the priority to longer matches to annotate collocations with their proper cluster rather than with the ones of the unigrams it contains (see Figure 3.9 in Chapter 3). We evaluated it using the same feature sets as the ones we used for the unigrams.

As we can see in Figure 4.5 which shows the global F1-score of the system for our various experiments, almost none of these attempts were successful. The baseline in this graph corresponds to the best results using LDA-clusters. The only case where a slight improvement is observed corresponds to the use of a large number of bi-grams, with only 200 clusters and the feature set A presented in Table 4.9. From a general point of view, we observed that collocations are only susceptible to bring a slight improvement when using some very basic features in the classifier. All the other experiments resulted in a decrease of global F1-score, almost equally distributed in the various NEs categories. Only *miscellaneous* named entities seemed to benefit from the addition of collocations (1.13% F1-score improvement at most, using the 4% best bi-gram collocations and 300 clusters), and the *persons* category seemed to suffer the most from the addition of collocations. This could be explained by the fact that the classifier might be better at learning the compositionality of Named Entities itself from the unigram clusters. The notion of compositionality might also be depending on the semantics,

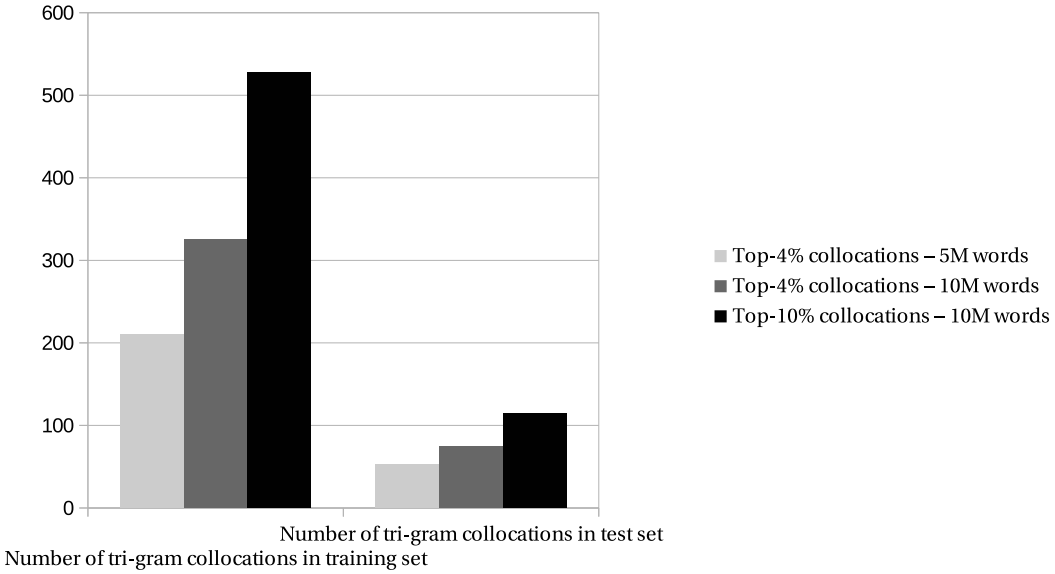


Figure 4.4 – Number of collocations found in the training set containing about 200 000 tokens in total and the test set containing about 45000 tokens in total

therefore, if this feature is useful at all, a log-linear classifier can develop finer and softer decisions than our test-statistics approach, which treats all the collocations in the same way.

Word clustering + Collocation clustering

We made a second type of experiments in which we use a different approach to include the collocations' clusters. We keep the word clusters and their corresponding features, but, for each word, we add the clusters corresponding to the collocation it belongs to (if any), using the second tagger described in Chapter 3, Figure 3.10. Using these new collocation clusters, we use the features shown in Table 4.14 in addition to the ones from Table 4.10 and the ones from the baseline (Table 4.1, Table 4.2, Table 4.3).

The results corresponding to this new setup are exposed in Figure 4.6. In this graph we used the best LDA clustering results as a baseline. The large input set (10 million words from wikipedia) was used to extract the collocations in every cases. This time we get a clear improvement of the F1-score using both 200 and 300 clusters. Using 200 clusters, the F1-score is 0.33% higher using the best 10% collocations, and using 300 clusters, the best results are achieved using the 4% best collocations (0.13% increase of the F1-score). Table 4.13 details the results for 300 clusters, and the 10% best collocations from the 10M words Wikipedia corpus, which leads to the best F1-score. This table has to be opposed to Table 4.12 which shows the results for the same parameters, without using collocations. A small improvement is observed for all categories of NE, except the *miscellaneous*, which seems to be negatively affected by our new features.

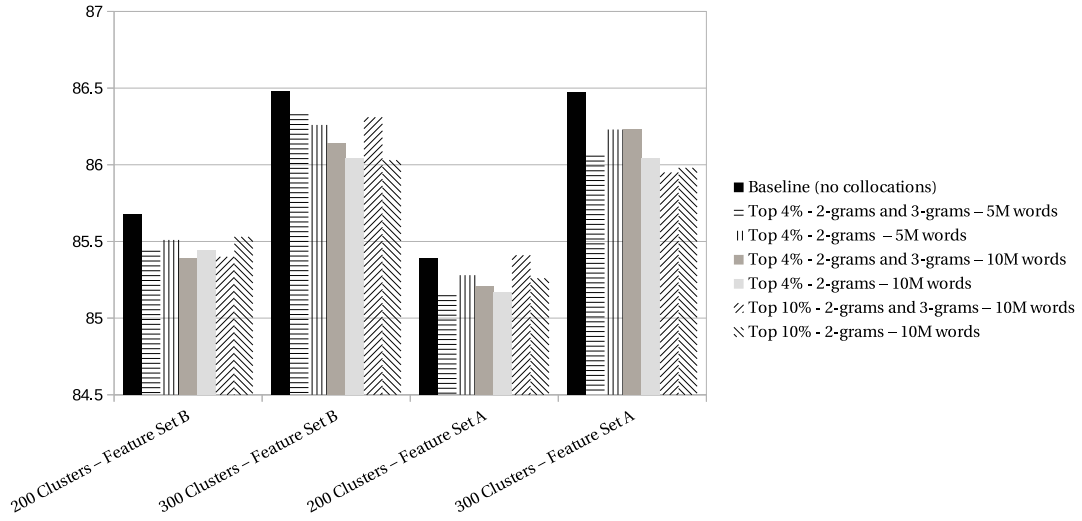


Figure 4.5 – Influence of collocation clustering on F1-score

NE Type	Precision	Recall	F1-Score	Accuracy
Overall	87.12	86.10	86.61	97.22
LOC	89.90	90.71	90.30	
MISC	80.21	74.50	77.25	
ORG	83.29	81.34	82.30	
PER	90.83	91.28	91.05	

Table 4.13 – NER best performance with LDA clusters for words and collocations, using 300 clusters, and the 10% best collocations from the 10M words Wikipedia corpus

Feature number	Feature Name
1	Index of best collocation cluster
2	Index of the second best collocation cluster
3	Index of the third best collocation cluster
4	Index of the best collocation cluster of the next word
5	Index of the second best collocation cluster of the next word
6	Index of the third best collocation cluster of the next word
7	Index of the best collocation cluster of the previous word
8	Index of the second best collocation cluster of the previous word
9	Index of the third best collocation cluster of the previous word
10	Index of the best collocation cluster of the word n-2
11	Index of the second best collocation cluster of the word n-2
12	Index of the best collocation cluster of the word n+2
13	Index of the second best collocation cluster of the word n+2
14	Token / Index of the best collocation cluster
15	Index of the best / second best collocation cluster
16	Index of the best / second best collocation cluster for the next word
17	Index of the best / second best collocation cluster for the previous word
18	Index of the best / second best / third best collocation cluster

Table 4.14 – NER additional collocation features

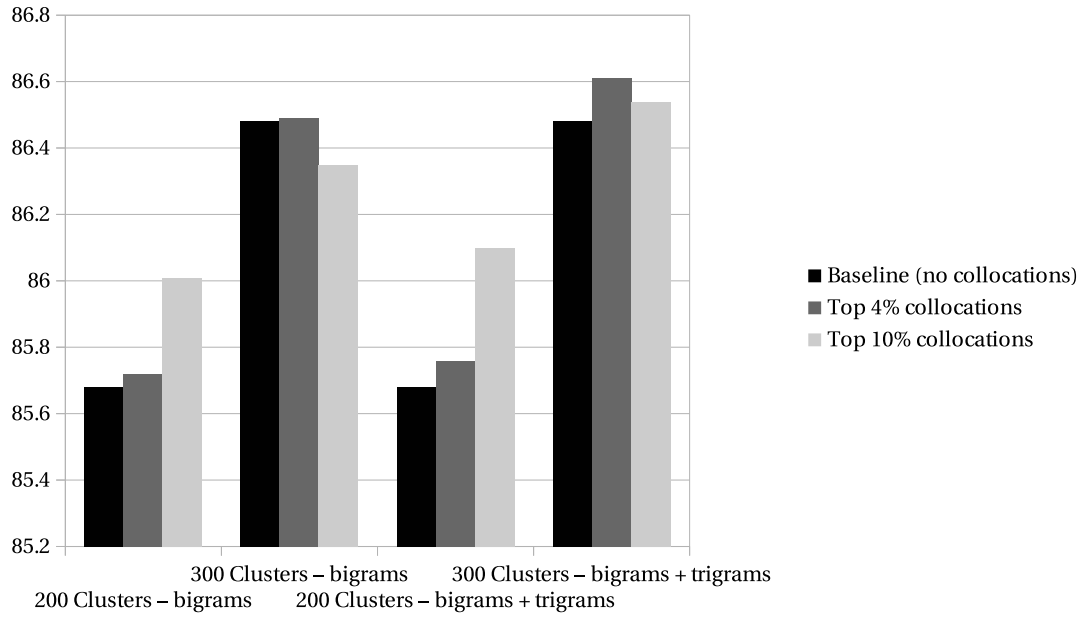


Figure 4.6 – F1-score using collocation clustering and additional features

4.3.4 Comparison

F1-score

The results are summarized in Figure 4.7. The overall best score we achieved on this task is 86.61% F1-score, which is slightly higher than the Stanford NER and higher than 14 out of the 16 systems in the initial CoNLL 2003 shared task competition for the same task. Nevertheless, some existing systems reached a better performance, by including other kind of latent semantic information, such as [KBK15] which combines LDA clusters with information from five other different semantic spaces.

Computational cost

Overall, we notice that in order to achieve the best respective performance of the two methods (i.e. 300 clusters for LDA, 750 clusters for Brown), the LDA approach is taking significantly less time according to Figure 3.11. Indeed, clustering using 300 LDA clusters took 303 minutes, whereas Brown clustering for 750 clusters took 798 minutes for our setup.

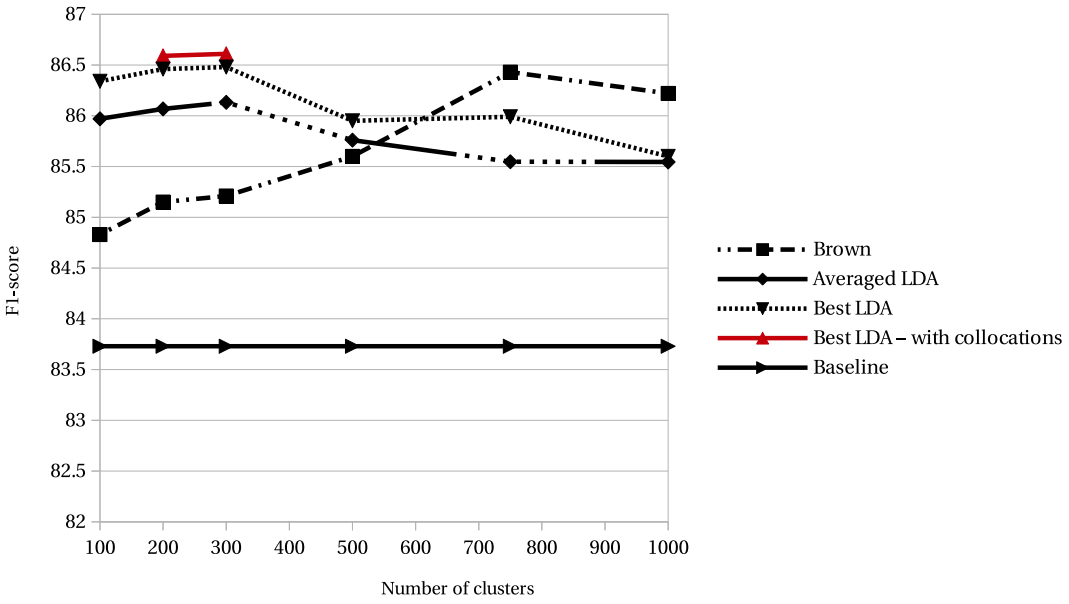


Figure 4.7 – Comparison of F1-score on NER for Brown and LDA methods

5 Application to Semantic Role Labeling

We will now investigate whether our semantic clustering could also be used to improve an existing Semantic Role Labeling component (from the Sony EuTEC Speech and Sound Processing Group NLP toolkit). After quickly presenting the logic of the classifiers used and defining the task in more detail, we will present and discuss the results we obtained with hierarchical and fuzzy clustering features.

5.1 Multinomial Logistic Regression (MaxEnt)

Logistic Regression is one of the most simple yet most popular probabilistic discriminative model, mainly used for classification tasks. When such a classifier is used as a binary classifier, the term *logistic regression* is preferred, whereas when there are more than two classes to discriminate, the term *MaxEnt*, *MEM*, *Multinomial Logistic Regression* or *Softmax classifier* is generally used. CRFs that we reviewed in Chapter 4 are in fact a direct application of logistic regression to sequences, the definition of this model will therefore seem familiar. The idea is the same; make use of a set of feature functions, which can bring useful information to discriminate between a set of classes.

X and Y are defined as two random variables (respectively corresponding to words and classes in our case). Given a class y, and a word x, we define m functions $f_1, f_2 ... f_m$ called feature functions. Each of them maps the parameters x, y to a real number, which in our case will be only 0 or 1. The same procedure as in Section 4.1 is used to build more complex feature functions which can take more than two values, by creating a one-hot representations of the values using multiple binary features. We also use a weight vector $\boldsymbol{\lambda}$, which is used to compute a "score" $\boldsymbol{\lambda} \cdot \mathbf{f}$ with $\mathbf{f} = f_1, f_2 ... f_m$, that is mapped to a probability value using the softmax function (the last feature of the vector $\mathbf{f}$ is set to 1 in order to incorporate a bias term implicitly). According to this model, the probability that a class y corresponds to a word x is

thus defined as followed;

$$P(y|x;\boldsymbol{\lambda}) = \frac{\exp(\sum_{j=1}^{m} \lambda_j f_j(x,y))}{\sum_{y' \in Y} \exp(\sum_{j=1}^{m} \lambda_j f_j(x,y))}$$

Just as it was the case for CRF, a variant of the gradient ascent algorithm (LM-BFGS [LN89]) is used to find the weights $\boldsymbol{\lambda}$ which maximizes the log-likelihood of this model (which also use L2 regularization) on our training set.

5.2 Task

We are considering the semantic dependency parsing aspect of the CoNLL 2008 shared task [SJM+08]. CoNLL 2009 task was similar but applied to several languages. The CoNLL task includes syntactic dependency parsing as well, but the component we are using does not address this part of the task. The training and test data are identical to the ones used in CoNLL 2009, as well as the evaluation measures (only semantic scores are relevant). More specifically, the task, that we already quickly defined in the introduction (Section 1.2.2), consists of identifying the predicates and labeling them with a sense, as well as identifying and labeling the corresponding arguments according to a propbank.

The component and baseline we are using consist of an implementation of the semantic dependency parser described in the article *A High-Performance Syntactic and Semantic Dependency Parser*, by Björkelund et al. [BBHN10]. It consists of 6 different log-linear classifiers, which respectively identify predicates (log-linear classifier), disambiguate them in case they have several senses (MaxEnt classifier), identify arguments (two binary log-linear classifiers, for nouns and verbs), and classify them (two MaxEnt classifiers, one for nouns and one verbs). The feature set used by the baseline for each of these classifiers is detailed in Table 5.1 for the single features, and Table 5.2 for the bigram features.

The system is implemented Java using the LIBLINEAR library [FCH+08].

5.3 Evaluation Metrics

F1-score, as well as precision and recall, are also used as evaluation metric for this task. These notions are defined in Section 4.2.2. The scores are computed in a similar way than in the CoNLL 2008 shared task [SJM+08]. We used both the unlabeled F1-score and the labeled F1-score to evaluate the component. When computing the unlabeled F1-score, we consider a dependency as correct if the positions of the tokens for both the predicate and the argument are correct. For the labeled F1-score, we consider a dependency as correct if the positions *and* the labels of the predicate and the argument are correct.

The baseline achieves an unlabeled F1-score of 83.57% and a labeled F1-score of 77.55% for

semantic dependencies labeling on the English CoNLL 2009 task. The unlabeled precision and recall are respectively 85.46% and 81.76%, and the labeled precision and recall respectively 79.30% and 75.88%.

	Predicate Identification	Predicate Classification	Noun Argument Identification	Noun Argument Classification	Verb Argument Identification	Verb Argument Classification
Predicate Word	X			X		
Predicate POS		X	X			X
Predicate Lemma	X		X	X		X
Predicate Parent Word	X	X			X	X
Predicate Parent POS	X	X			X	X
Children Dependencies	X	X				X
Children Words	X	X	X			
Children POS tags	X	X		X		
Children Dependencies concatenated	X	X				
Predicate relation to parent	X	X				
Dependency relation path to the argument			X		X	X
Argument Word			X	X	X	X
Argument POS			X	X	X	X
Right Word			X	X		X
Right Word POS			X	X		X
Left Word				X		
Left Word POS						X
POS path			X		X	X
Argument dependency relation					X	X
Sense				X	X	X
Position			X	X	X	X
Left sibling word				X		
Left sibling POS						X
Right sibling word				X	X	
Right sibling POS						
Predicate dependency relations						

Table 5.1 – Baseline SRL single features

5.4 Results

5.4.1 Brown Clustering

Using the same procedure and clusters as for the NER task, we came up with the features shown in Table 5.3, which lead to the best results among our experiments. The overall results are shown in Figure 5.1 and 5.2. As for the NER task, the best results are obtained with the highest number of clusters. However, we notice that the improvement is more limited, with a 0.7% increase in labeled F1-score and a 0.4% improvement in unlabeled F1-score. Using 1000 clusters (which provides the best results), the labeled precision and recall are respectively 79.99% and 76.56%, and the unlabeled precision and recall reach respectively 85.84% and 82.16%. By comparing these values to the ones of the baseline, we can see that recall is more positively affected than precision when using Brown clusters.

5.4.2 Fuzzy Clustering

Given our time constraints, and the relatively low improvement margin of semantic clusters on this task according to our experiments, we decided not to investigate collocation clustering for the Semantic Role Labeling (SRL) component. However we did some experiments using unigram LDA based fuzzy clusters, using the same setup as the one we used for the NER fuzzy clustering experiments, and the set of features shown in Table 5.4 (after a feature selection process based on recursive feature ablation). The results this time are not as encouraging. As

we can see on Figures 5.1 and 5.2 which contains the averaged F1-scores (Table 5.5 contains the corresponding standard deviations), we did not manage to get more than 0.69% improvement on labeled F1-score, and 0.23% improvement on the unlabeled F1-score, which is less than what we achieved using Brown clusters. These results were less good than expected, especially considering the fact that Chrupala [Chr11] could improve an SRL component by more than 3% using a similar technique. Several factors could explain it. First, we are once again limited by the library we are using, LIBLINEAR does not support continuous features, and unlike Chrupala we had to use the same kind of discrete features (cluster indices) than on the NER task, and we could not take benefit from the full information contained in the membership values of the clusters. Secondly, the SRL component we are using is already close to the state-of-the-art performance, and more importantly, is already including some semantic information in the features (see Table 5.1). Finally, due to time constraints, and a particularly long training time for this component (about 4 hours), we had to limit more our feature selection process.

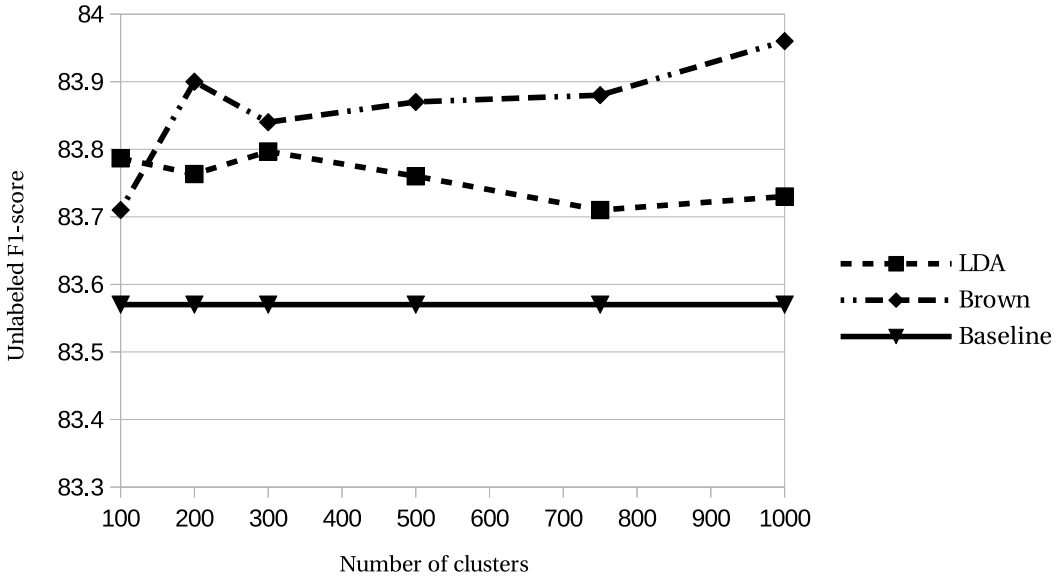


Figure 5.1 – Unlabeled F1-score for Semantic Role Labeling

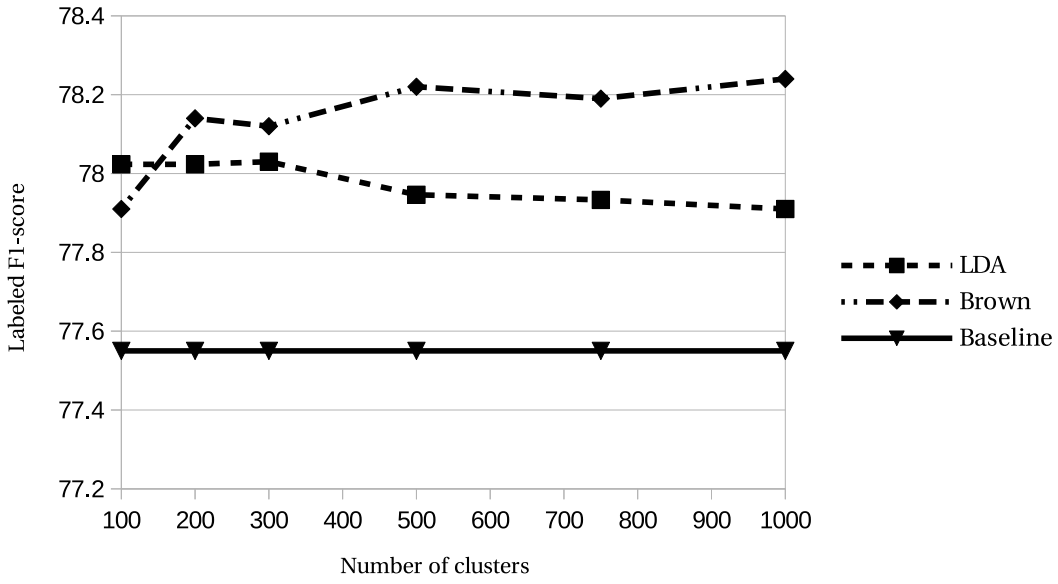


Figure 5.2 – Labeled F1-score for Semantic Role Labeling

	Predicate Identification	Predicate Classification	Noun Argument Identification	Noun Argument Classification	Verb Argument Identification	Verb Argument Classification
Predicate POS/ Predicate Parent POS		X				
ChildrenWords/ Children Dependencies		X				
Children POS tags/ Children Dependencies		X				
Children Words/ Predicate Parent Word		X				
Children Words/ Predicate Parent POS		X				
Children Words/ Predicate POS		X				
Children Dependencies Concatenated/ Predicate Parent Word		X				
Argument Dependency Relation/ Predicate Parent Word			X			
Argument POS/ Sense			X	X		
Argument Word/ Position			X	X		
Children Words/ Position			X			
Left word POS/ Predicate Word			X			
Sense/ Position			X	X		X
Argument POS/ Position				X		
Argument Word/ Sense				X		X
Left POS Word/ Right Sibling POS				X		
Left Sibling POS/ Sense				X		
POS Path/ Sense				X		
Position/ Right Sibling POS				X		
Argument Dependency Relation/ Argument POS					X	X
Argument Dependency Relation/ Children Dependencies					X	
Argument Dependency Relation/ Dependency Relation Path to Argument					X	
Argument Dependency Relation/ Predicate Dependency Relations					X	
Right Word POS/ Left Word POS					X	
POS Path/ Right Sibling POS					X	
Argument Dependency Relation/ Right Word POS						X
Argument Dependency Relation/ Sense						X
Argument POS/ Argument Word						X
Argument POS/ Predicate Lemma						X
Children dependencies/ Position						X

Table 5.2 – Baseline SRL combined features

	Predicate Identification	Predicate Classification	Noun Argument Identification	Noun Argument Classification	Verb Argument Identification	Verb Argument Classification
Argument cluster prefix of length 12			X	X	X	X
Argument cluster prefix of length 8			X	X	X	X
Argument cluster prefix of length 4			X	X	X	X
Predicate cluster prefix of length 12	X	X	X	X	X	X
Predicate cluster prefix of length 8	X	X	X	X	X	X
Predicate cluster prefix of length 4	X	X	X	X	X	X
Right-most argument cluster prefix of length 12			X	X	X	X
Right-most argument cluster prefix of length 8			X	X	X	X
Left-most argument cluster prefix of length 12			X	X	X	X
Left-most argument cluster prefix of length 8			X	X	X	X
Parent of predicate cluster prefix of length 12	X	X	X	X	X	X
Parent of predicate cluster prefix of length 8	X	X	X	X	X	X
Children dependencies cluster prefix of length 12	X	X	X	X	X	X
Children dependencies cluster prefix of length 8	X	X	X	X	X	X
Right sibling cluster prefix of length 12				X	X	X
Right sibling cluster prefix of length 8				X	X	X
Left sibling cluster prefix of length 12				X	X	X
Left sibling cluster prefix of length 8				X	X	X

Table 5.3 – Brown clustering feature for Semantic Role Labeling

	Predicate Identification	Predicate Classification	Noun Argument Identification	Noun Argument Classification	Verb Argument Identification	Verb Argument Classification
Argument top-1 cluster index			X	X	X	X
Argument top-2 cluster index			X	X	X	X
Argument top-3 cluster index			X	X	X	X
Predicate top-1 cluster index	X	X	X	X	X	X
Predicate top-2 cluster index	X	X	X	X	X	X
Predicate top-3 cluster index	X	X	X	X	X	X
Right-most argument top-1 cluster index			X	X	X	X
Right-most argument top-2 cluster index			X	X	X	X
Left-most argument top-1 cluster index			X	X	X	X
Left-most argument top-2 cluster index			X	X	X	X
Parent of predicate top-1 cluster index	X	X	X	X	X	X
Parent of predicate top-2 cluster index	X	X	X	X	X	X
Children dependencies top-1 cluster index	X	X	X	X	X	X
Children dependencies top-2 cluster index	X	X	X	X	X	X
Right sibling top-1 cluster index				X	X	X
Right sibling top-2 cluster index				X	X	X
Left sibling top-1 cluster index				X	X	X
Left sibling top-2 cluster index				X	X	X
Children dependencies/ Predicate top-1/2 clusters		X				
Argument Word/ Predicate top 1/2 clusters			X	X	X	X
Argument dependency relation/ Argument top 1/2 clusters			X		X	X

Table 5.4 – LDA clustering features for Semantic Role Labeling

Number of clusters	100	200	300	500	750	1000
Unlabeled F1-score standard deviation	0.015	0.064	0.045	0.044	0.045	0.032
Labeled F1-score standard deviation	0.030	0.090	0.01	0.035	0.073	0.07

Table 5.5 – Standard deviation of the SRL component F1-Score with LDA clusters

6 Conclusion

Extracting collocations from a corpus in an unsupervised way was an unsurprisingly difficult task, which is even more difficult to evaluate given the lack of a proper corpus annotated with collocations, and the lack of an actual widely accepted definition for the notion of collocations. However, we combined two of the most commonly used techniques, namely a syntactic rule-based approach and test statistics using association measures to build a new collocation extraction system. We evaluated it against the Wiki50 corpus which appears to be, as far as we know, the most relevant gold data for this task. We achieved an F1-score of 32% on bi-gram collocations extraction using log-frequency biased mutual dependency measure which jumps to nearly 50% when using a syntactic filter in addition. The system is also able to retrieve collocations of any length higher than two, but at the cost of a much lower precision.

The core of our work is based on the fuzzy clustering of words using the method developed by Grzegorz Chrupala [Chr11]. We generalized this approach to collocation clustering. We also introduced a simple heuristic method which allows to map clusters to a Wordnet synset (and name them) using hypernym dependency graphs that we applied to the output of the LDA based clustering algorithm.

Finally these clusters were used to build new features for two existing NLP components, and were compared to Brown hierarchical clusters, in terms of how it could be used to improve the performance of the system and at what cost. On the NER component, we were able to reproduce the results from the literature and improve the component F1-score performance by nearly 3%. It appears that fuzzy clusters have some notorious advantages over hierarchical clusters both in terms of performance and computational cost. The clusters corresponding to collocations slightly improved the performance of the component when used in addition to the word clusters with a carefully chosen feature set, but could not increase the performance when they were used as a replacement of word clusters. On the semantic role labeling component, the improvement induced by fuzzy clusters was less noticeable, and hierarchical clusters outperformed them by 0.2%, improving the overall labeled F1-score by 0.7%.

We identified several interesting topics which could be approached in the future starting from

our work and the current literature:

Multilingual applications

One could investigate how the semantic clusters and the collocation extraction technique we described could be adapted to other languages. Indeed, for these tasks, unsupervised methods are used, so they should be easily transposable to other languages than English. Both CoNLL 2003 NER task and CoNLL 2009 SRL tasks are also defined for various languages. Some work in this direction has already been done by Konkolet al. [KBK15], which evaluate various features based on latent semantics for a NER component in English, Spanish, Dutch and Czech.

More NLP components

Other NLP components could potentially benefit from semantic clusters; our system was used to improve a CRF based target detection component for sentiment analysis, by using similar features than the ones we considered for Named Entity Recognition, which led to a 1% F1-score improvement. Literature also claim that parsing performance could be enhanced using such clusters [Chr11].

Hierarchical Fuzzy Clustering

We investigated separately hierarchical and fuzzy clustering, but an ideal approach would be to consider a hierarchical fuzzy clustering algorithm, which could be able to model both hierarchical semantic relations (meronymy/holonymy, hypernymy/hyponymy) *and* polysemic and homonyic behaviors. More specifically, the LDA based clustering method we are using could be extended to use a non-parametric prior which can model topic hierarchies [GT04, GB12].

Phrases Clustering

Focusing on collocations is an important restriction, and we showed that the resulting improvement on NLP components is limited. However, it seems that semantically clustering every phrase without further selection could be much more useful. This was actually used in a CRF based implementation of a Named Entity Recognizer [LW09] which offers the current best performance on the CoNLL 2003 task. Applying the LDA based fuzzy clustering to phrases would be costly but might lead to interesting results.

Representing the semantics of words and phrases is still an open problem that has been embraced by researchers for a long time, and clustering is one popular and efficient way of addressing it. However, research in this field is still very active and the recent interest in deep learning applications in NLP led to a wide new range of techniques to model the semantics of words. Language models based on recurrent neural networks [BDVJ03] can produce high quality word embeddings, i.e. low dimensional representation of words' semantic and syntactic properties. More recently, it is worth mentioning the work of Mikolov on *word2vec*, a novel RNN based architecture to build semantic embeddings that reflect the semantics of words and phrases [MCCD13, MSC+13] or this unexpected yet impressive character based

deep convolutional neural network architecture for text understanding [ZL15], which can give us a good idea of the direction that this field might take in the following years.

Bibliography

[AH09] Anastasiou and Kim Hashimoto, Nakov. Multiword expressions: Identification, interpretation, disambiguation, application. In *Proceedings of the 2009 Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation, Application*. Suntec, Singapre. ACL., 2009.

[ARPS13] Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 183–192, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[BBHN10] Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. A high-performance syntactic and semantic dependency parser. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 33–36. Association for Computational Linguistics, 2010.

[BDM+92] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

[BDVJ03] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

[BEF84] James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2):191–203, 1984.

[BJ01] Don Blaheta and Mark Johnson. Unsupervised learning of multi-word verbs. In *Proc. of the ACL/EACL 2001 Workshop on the Computational Extraction, Analysis and Exploitation of Collocations*, pages 54–60, 2001.

[BNJ03] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[BP66] Leonard E Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, pages 1554–1563, 1966.

[BR73] Godelieve Berry-Rogghe. The computation of collocations and their relevance in lexical studies. *The Computer and Literary Studies*, pages 103–112, 1973.

[Bré04] Michel Bréal. *Essai de sémantique:(science des significations)*. Hachette, 1904.

[BZL11] Fan Bu, Xiao-Yan Zhu, and Ming Li. A new multiword expression metric and its applications. *Journal of Computer Science and Technology*, 26(1):3–13, 2011.

[CH90] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990.

[Cha11] Jean-Cédric Chappelier. Modèles génératifs à base de thèmes pour l'accès à l'information textuelle. In *Modèles statistiques pour l'accès à l'information textuelle*. 2011.

[Cho88] Yaacov Choueka. Looking for needles in a haystack or locating interesting collocational expressions in large textual databases. In *RIAO 88:(Recherche d'Information Assistée par Ordinateur). Conference*, pages 609–623, 1988.

[Chr11] Grzegorz Chrupala. Efficient induction of probabilistic word classes with lda. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 363–372, 2011. Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[Dic45] Lee R Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[dSL99] J Ferreira da Silva and G Pereira Lopes. A local maxima method and a fair dispersion normalization for extracting multi-word units from corpora. In *Sixth Meeting on Mathematics of Language*, pages 369–381, 1999.

[FCH+08] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[FGM05] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

[Fir57] JR Firth. A synopsis of linguistic theory 1930-1955, volume 1952-59. the philological society, 1957.

[GB12] Samuel J Gershman and David M Blei. A tutorial on bayesian nonparametric models. *Journal of Mathematical Psychology*, 56(1):1–12, 2012.

[Gri02] Tom Griffiths. Gibbs sampling in the generative model of latent dirichlet allocation, 2002. https://people.cs.umass.edu/~wallach/courses/s11/cmpsci791ss/readings/griffiths02gibbs.pdf.

[GT04] DMBTL Griffiths and MIJJB Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. *Advances in neural information processing systems*, 16:17, 2004.

[Har54] Zellig S Harris. Distributional structure. *Word*, 1954.

[Hof99] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

[HPW+04] Kadri Hacioglu, Sameer Pradhan, Wayne Ward, James H Martin, and Daniel Jurafsky. Semantic role labeling by tagging syntactic chunks. In *CoNLL-2004 Shared Task*, 2004.

[HT11] A.S. Hornby and J. Turnbull. *Oxford Advanced Learner's Dictionary of Current English*. Oxford University Press, 2011.

[KBK15] Michal Konkol, Tomáš Brychcín, and Miloslav Konopík. Latent semantics in named entity recognition. *Expert Systems with Applications*, 42(7):3470–3479, 2015.

[Kud07] Taku Kudoh. Crf++. http://crfpp.googlecode.com/svn/trunk/doc/index.html, 2007.

[LCL+05] Ting Liu, Wanxiang Che, Sheng Li, Yuxuan Hu, and Huaijun Liu. Semantic role labeling system using maximum entropy classifier. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 189–192. Association for Computational Linguistics, 2005.

[LG14] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185, 2014.

[Lia05] Percy Liang. *Semi-supervised learning for natural language*. PhD thesis, Massachusetts Institute of Technology, 2005.

[Lia12] Percy Liang. brown-cluster. https://github.com/percyliang/brown-cluster, 2012.

[Lin] Dekang Lin. Extracting collocations from text corpora. Citeseer.

[Lin99] Dekang Lin. Automatic identification of non-compositional phrases. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 317–324. Association for Computational Linguistics, 1999.

[LMP01] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[LN89] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[LW09] Dekang Lin and Xiaoyun Wu. Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1030–1038. Association for Computational Linguistics, 2009.

[M+67] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[McC02] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[MS99] Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

[MSC+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[NN06] Richard Nock and Frank Nielsen. On weighting clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(8):1223–1235, 2006.

[Pea02] Darren Pearce. A comparative evaluation of collocation extraction techniques. In *LREC*, 2002.

[Pec05] Pavel Pecina. An extensive empirical study of collocation extraction methods. In *Proceedings of the ACL Student Research Workshop*, pages 13–18. Association for Computational Linguistics, 2005.

[Pec10] Pavel Pecina. Lexical association measures and collocation extraction. *Language resources and evaluation*, 44(1-2):137–158, 2010.

[RDAV12] Carlos Ramisch, Vitor De Araujo, and Aline Villavicencio. A broad evaluation of techniques for automatic acquisition of multiword expressions. In *Proceedings of ACL 2012 Student Research Workshop*, pages 1–6. Association for Computational Linguistics, 2012.

[SBB+02] Ivan A Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. Multiword expressions: A pain in the neck for nlp. In *Computational Linguistics and Intelligent Text Processing*, pages 1–15. Springer, 2002.

[Ser11] Violeta Seretan. *Syntax-based collocation extraction*, volume 44. Springer Science & Business Media, 2011.

[SFK06] György Szarvas, Richárd Farkas, and András Kocsor. A multilingual named entity recognition system using boosting and c4. 5 decision tree learning algorithms. In *Discovery Science*, pages 267–278. Springer, 2006.

[SIJ06] Mika Sato-Ilic and Lakhmi C Jain. Fuzzy clustering based principal component analysis. In *Innovations in Fuzzy Clustering*, pages 9–44. Springer, 2006.

[SJ01] Patrick Schone and Daniel Jurafsky. Is knowledge-free induction of multiword unit dictionary headwords a solved problem. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 100–108, 2001.

[SJM+08] Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177. Association for Computational Linguistics, 2008.

[SM11] Charles Sutton and Andrew McCallum. An introduction to conditional random fields. *Machine Learning*, 4(4):267–373, 2011.

[Ste10] A. Stevenson. *Oxford Dictionary of English*. Oxford reference online premium. OUP Oxford, 2010.

[TFK02] Aristomenis Thanopoulos, Nikos Fakotakis, and George Kokkinakis. Comparative evaluation of collocation extraction metrics. In *LREC*, volume 2, pages 620–625, 2002.

[TRB10] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[VNB11] Veronika Vincze, István Nagy, and Gábor Berend. Multiword expressions and named entities in the wiki50 corpus. In *RANLP*, pages 289–295, 2011.

[WC11] Quan Wen and M Emre Celebi. Hard versus fuzzy c-means clustering for color quantization. *EURASIP Journal on Advances in Signal Processing*, 2011(1):1–12, 2011.

[WSN10] Eric Wehrli, Violeta Seretan, and Luka Nerima. Sentence analysis and collocation identification. COLING, 2010.

[ZL15] Xiang Zhang and Yann LeCun. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*, 2015.