Ambiruptor
The Lexical Ambiguity Interruptor

Final Report

Maria Boritchev
Boumediene Brikci Sid
Victor Hublitz
Simon Mauras
Pierre Ohlmann
Ievgeniia Oshurko
Samir Tendjaoui
Thi Xuan Vu

May 13, 2016

Abstract

The main point of our project is to develop a word-sense disambiguation tool. Our aim is to be able, given a certain text, to map each ambiguous word to the meaning it has in this context. To this end, we use Wikipedia, and more specifically, its internal links, in order to produce automatically an annotated corpus from which a machine learning framework is developed. Using the obtained tool, we created an interactive application, giving users the possibility to improve the word-sense disambiguation mapping.

Contents

1 Presentation **1**
1.1 The Ambiruptor Project . . . 1
1.2 The Ambiruptor Team . . . 2
1.3 Home . . . 2

2 Research & Design **3**
2.1 Word-Sense Disambiguation . . . 3
2.2 Machine Learning . . . 3
2.3 Feature extraction . . . 4
2.4 Data Mining . . . 4

3 Implementation **6**
3.1 Design . . . 6
3.2 Data Mining . . . 7
3.3 Features . . . 7
3.4 Learning Models . . . 8
3.5 Interfaces . . . 8
3.5.1 Web Application . . . 8
3.5.2 Firefox Plugin . . . 8

4 Results & Applications **10**
4.1 Examples . . . 10
4.2 Statistics . . . 10
4.3 Conclusion . . . 12

Bibliography **13**

1. Presentation

Word-Sense Disambiguation is a Natural Language Processing task that lies in the assignment of the appropriate meaning to a word according to a given context, separating this meaning from all other possible ones. Since the 1940s, this problem has proved its difficulty and the lack of databases has forced the researchers working on it to label each word manually.

Nowadays, the Internet creates new possibilities to get big databases. The use of new machine-learning methods combined to these databases has given more efficient results on this open problem.
There are several possible applications of Word-Sense Disambiguation:

- Machine Translation;
- Information Retrieval;
- Semantic Parsing;
- Speech Synthesis and Recognition.

Indeed, one can think of Word-Sense Disambiguation in the context of Artificial Intelligence research, as human speech recognition is essential in this field.

1.1 The Ambiruptor Project

The Ambiruptor project was created while discussing wordplays: what makes a wordplay funny? Part of the answer lies in the existing ambiguity of words; the same word, depending on its context, can have several meanings. This observation gave birth to the Ambiruptor project: we wanted to create a tool able to automatically recognize "ambiguous" words and assign them the right meaning according to the context.

Wikipedia's internal structure contains some disambiguation pages, indexing which helps us identify ambiguous words. As these pages group links corresponding to Wikipedia pages of different meanings of the ambiguous words, our task can be summed up as assigning the right link.

The Natural Language Toolkit (NLTK) is a Python package for natural language processing (NLP). It provides over 50 corpora and lexical resources, along with a suite of libraries and programs for symbolic and statistical NLP such as classification (maximum entropy, naive Bayes, k-means, etc), tokenizing (splitting paragraphs into sentences, splitting sentences into words), part-of-speech tagging corresponding to each words, etc.

The main objective of the **Ambiruptor project** is to produce an efficient tool that gives the correct meaning of ambiguous words in a text. Our tool is based on several supervised machine learning concepts, coded using NLTK. We use Wikipedia to build our learning corpus and to annotate it according to its internal links.

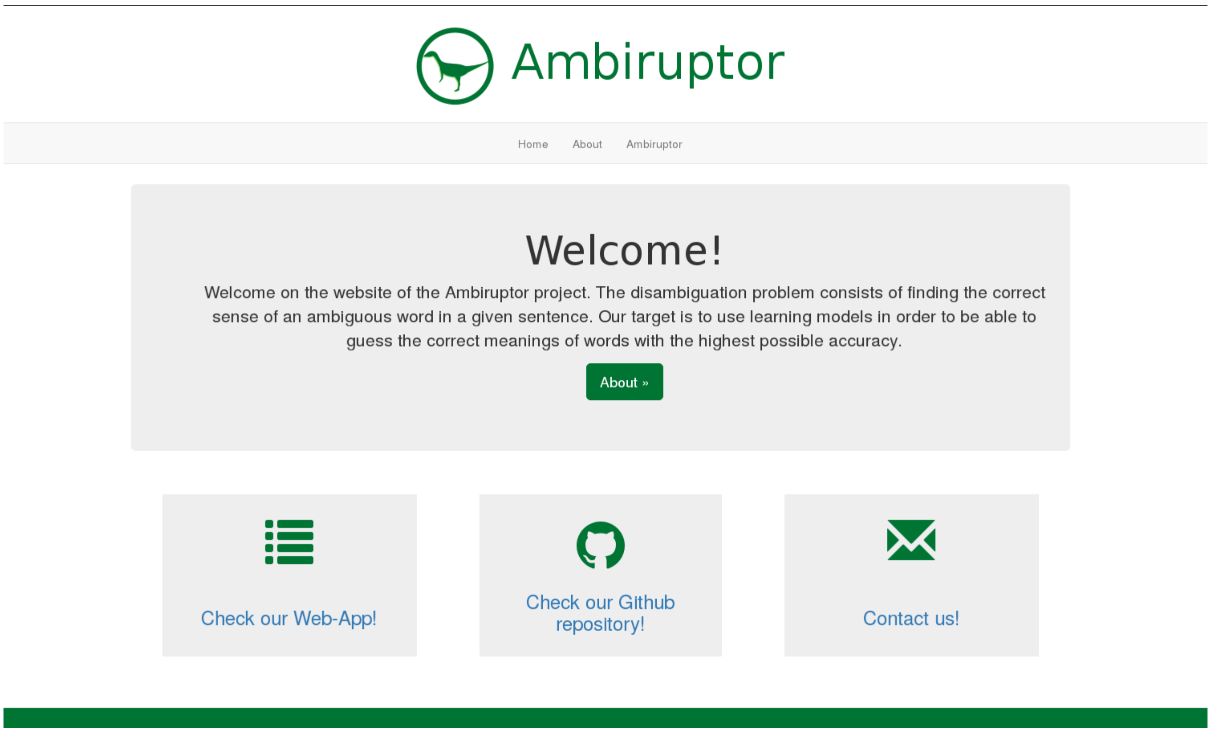
All the code we produce is under the GNU GPLv3 license.

1.2 The Ambiruptor Team

Our team is composed of 8 master students of the ENS of Lyon: Maria Boritchev, Boumediene Brikci Sid, Victor Hublitz, Simon Mauras, Pierre Ohlmann, Ievgeniia Oshurko, Samir Tendjaoui and Thi Xuan Vu. The project's coordinators are Simon Mauras and Ievgeniia Oshurko.

1.3 Home

Our project is materialized with a web application: `http://37.187.123.90:5000/`.



Welcome!

Welcome on the website of the Ambiruptor project. The disambiguation problem consists of finding the correct sense of an ambiguous word in a given sentence. Our target is to use learning models in order to be able to guess the correct meanings of words with the highest possible accuracy.

2. Research & Design

First of all, we explored the state of the art of word-sense disambiguation, data mining and machine learning. Then, we focused on the problem of matching those different modules together to choose the parameters.

2.1 Word-Sense Disambiguation

There are several approaches to the Word-Sense Disambiguation problem. We can split them in three different categories. Are usually considered:

- Dictionary-based methods;
- Unsupervised methods;
- Supervised methods.

Supervised learning is currently the most effective method, but it requires an annotated corpus in order to train the algorithm. Our goal is to provide a tool using a supervised learning algorithm on automatically built corpora. The advantage of this approach is that our tool retains the accuracy of supervised methods and can easily be adapted to different situations (e.g. different languages).

We considered two possible approaches to the fact that we need to disambiguate several words. We could either get one single model that gives the correct meaning for every word, or get one model per ambiguous word. The second approach was chosen for several reasons:

- The computations can be easily distributed;
- The feature extraction can be specific to the ambiguous word;
- The corpus for each model is smaller.

2.2 Machine Learning

Supervised approaches to WSD are based on machine learning, a method of data analysis that automates analytical model building. Machine learning explores the study and construction of algorithms that can learn from data and make predictions on data. Here, we focus on classifiers: a particular class of algorithms used to identify (classify) which category a new input belongs to, based on the knowledge of a classification for already-known data (also known as training set). When using a classifier, the first step is to fit (or train) a model using labeled data. Then we are able to predict the class of unlabeled data using similarities between the corpus and the request.

Let us consider a small example: classification of data within two classes. Let E be a set and $S \subseteq E$ be the set of elements of the first class (then, second class is $E \setminus S$). Our input is

$(X_i, y_i)_{1 \leq i \leq n} \in (E \times \{0,1\})^n$ such that $y_i = \mathbb{1}_S(X_i)$ for all i. A model is a set $\{H_n\}_n$ of subsets of E. The objective is to find $n \in \mathbb{N}$ such that H_n and S are as close as possible (it is the *fitting* part). Then our classification function is $\mathbb{1}_{H_n}$.

In Natural Language Processing, Support Vector Machines are usually quite efficient. We consider a vector space E and $\{H_n\}$, the set of hyperplanes. We tried several other learning models (see section 3.4)

In order to classify data, we need to extract interesting values (features) which will help to characterize the input, from raw text data. This process is called feature extraction and is explained further.

2.3 Feature extraction

Word-Sense Disambiguation is a Natural Language Processing task for which the context of the considered word is of major importance. In order to process this context, one needs to define some *features*. These are key points to look for in the input sentence. Features help us catch information and knowledge about the context of the target words to be disambiguated. The process of disambiguation cannot be done without these. Features that can be considered are part-of-speech labelling, morphological form identification and frequency considerations (see [1]).

Meaning	**Related words**
Living plant	green, algae, land, water, food, cell, ...
Manufacturing plant	factory, industry, manufactory, build, product, engine, process, artisan, chemical, ...

Table 2.1: Related words for “plant”

If we want to disambiguate an occurrence of the word “plant” in a text, the presence of words related to one of the meanings is a rather good hint.

2.4 Data Mining

The supervised learning approach for text disambiguation implies having a corpus with pre-labelled ambiguous words. We have two ways of obtaining such a corpus: either by manually labelling ambiguous words or by using existing resources to build our data automatically. The first solution is more accurate but requires much more time, therefore we chose the second one.

Manual use of Wikipedia data for disambiguation has already been done in [2]. The important point in our work is the fact that no human annotations are required. The main idea is to consider that each meaning of an ambiguous word is represented by a wiki-page. The disambiguation page allows us to get the different meanings of a given word. Links between wiki-pages are considered labelled words. The Figure 2.1 describes how we build a corpus to disambiguate a word.

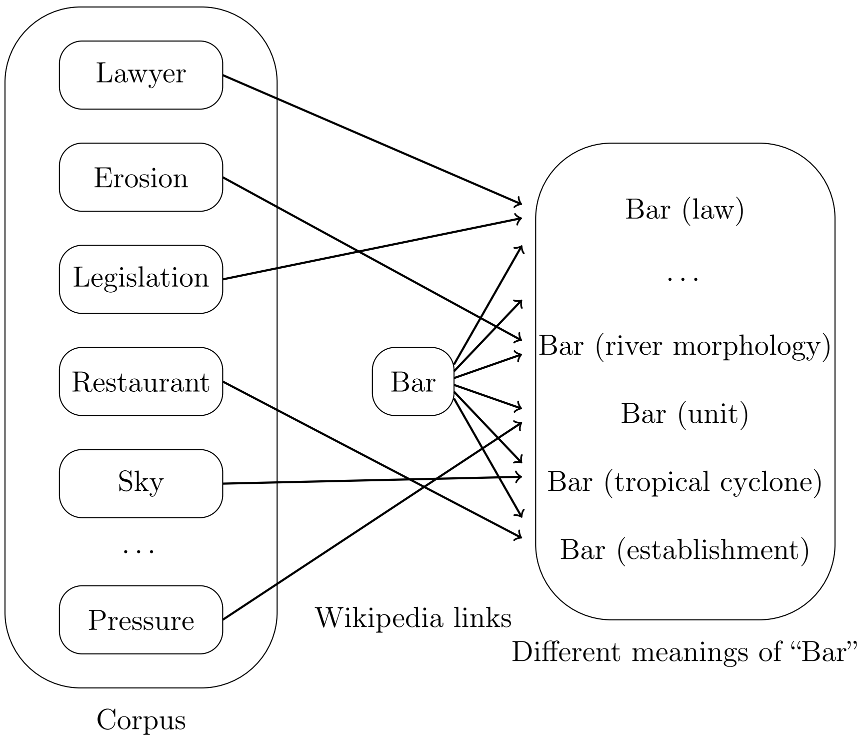


Figure 2.1: Corpus associated to the ambiguous word “Bar”.

3. Implementation

3.1 Design

Our goal was to develop the design of a library which would be easy to use, compatible with other Python libraries, and which would allow us the simultaneous development of different sub-modules of the project and ensure the re-usability of implemented features. Figure 3.1 gives a global overview of the disambiguation process that was adopted.

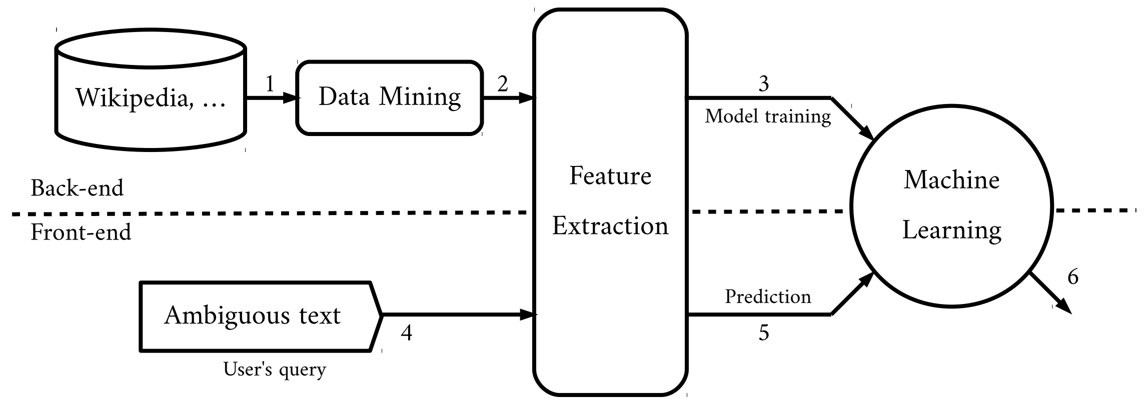


Figure 3.1: General pipeline

The library is divided into three modules:

- Module `miners` includes tools for mining and formatting training corpus (currently, Wikipedia mining tools are implemented).
- Module `preprocessors` includes data structures for representation of training data and ambiguous text. It also encapsulates text preprocessing tools and feature extractors for various features.
- Module `learners` consists of learning models that we use to build disambiguation model.

One of our goals was to allow people to use the front-end of our library without having to start over data-mining, feature extraction and model fitting. Figure 3.2 illustrates how the front-end can be used to disambiguate words.

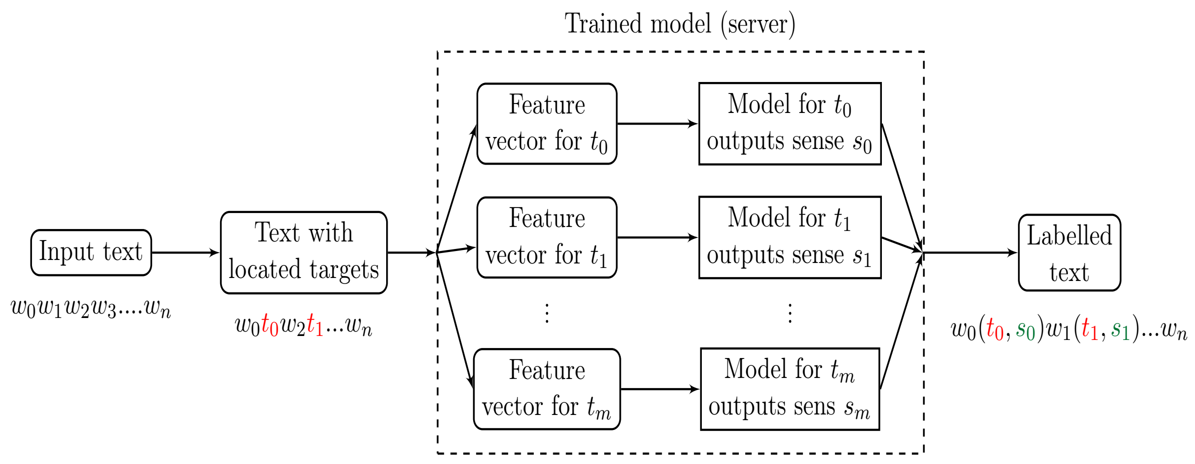

Trained model (server)
Input text
$w_0w_1w_2w_3....w_n$
Text with located targets
$w_0t_0w_2t_1...w_n$
Feature vector for t_0
Feature vector for t_1
Feature vector for t_m
Model for t_0 outputs sense s_0
Model for t_1 outputs sense s_1
Model for t_m outputs sens s_m
Labelled text
$w_0(t_0, s_0)w_1(t_1, s_1)...w_n$

Figure 3.2: Pipeline of the front-end library

3.2 Data Mining

In section 2.4 we described how we build the corpus for an ambiguous word using internal links. Let's explain our implementation choices.

Dumps of wikipedia articles can be downloaded at the url `https://dumps.wikimedia.org/enwiki/latest/`. A dump file of the content of every english article (without the history) is a ∼120 GB XML file. The first problem is that such a file can't fit into the memory of any computer we have. We therefore chose to store the articles and the links in a SQLite database, as:

- No need of any external SQL server;
- One database is stored in one file (∼160GB);
- Fast queries are possible using indexes (B-trees).

We used Python modules `xml.sax.xmlreader` and `sqlite3` to parse the XML file and build the database. Then we sanitized (remove wikipedia tags) articles using the `mwparserfromhell` package.

3.3 Features

We implemented several features among those presented in [1]. Source code can be found in the module `preprocessors`.

First, the parts of speech of the words in a fixed window around the ambiguous word gives informations on the structure of the sentence. For example we can disambiguate "in a bar" using the fact that "bar" follows the preposition "in". This is implemented in the class `PartOfSpeechFeatureExtractor`.

Another set of features are the typical words. As we build one model for each ambiguous word, we can have features that depend on words we want to disambiguate (target word). Typical words are words that are often used close to the target one. For example, the word "tree" close to "plant" is a big hint for the actual meaning of "plant". This is implemented in the class `CloseWordsFeatureExtractor`.

3.4 Learning Models

The following supervised Machine Learning techniques were used:

- Gaussian Naive Bayes;
- Decision Tree Classifier;
- Random Forest Classifier;
- K-Nearest Neighbors Classifier;
- SVM with Linear Kernel;
- SVM with RBF Kernel.

Each of the implemented learning models uses `scikit-learn` models as a kernel. It also allows us to evaluate models with help of various estimators provided by `scikit-learn`.

3.5 Interfaces

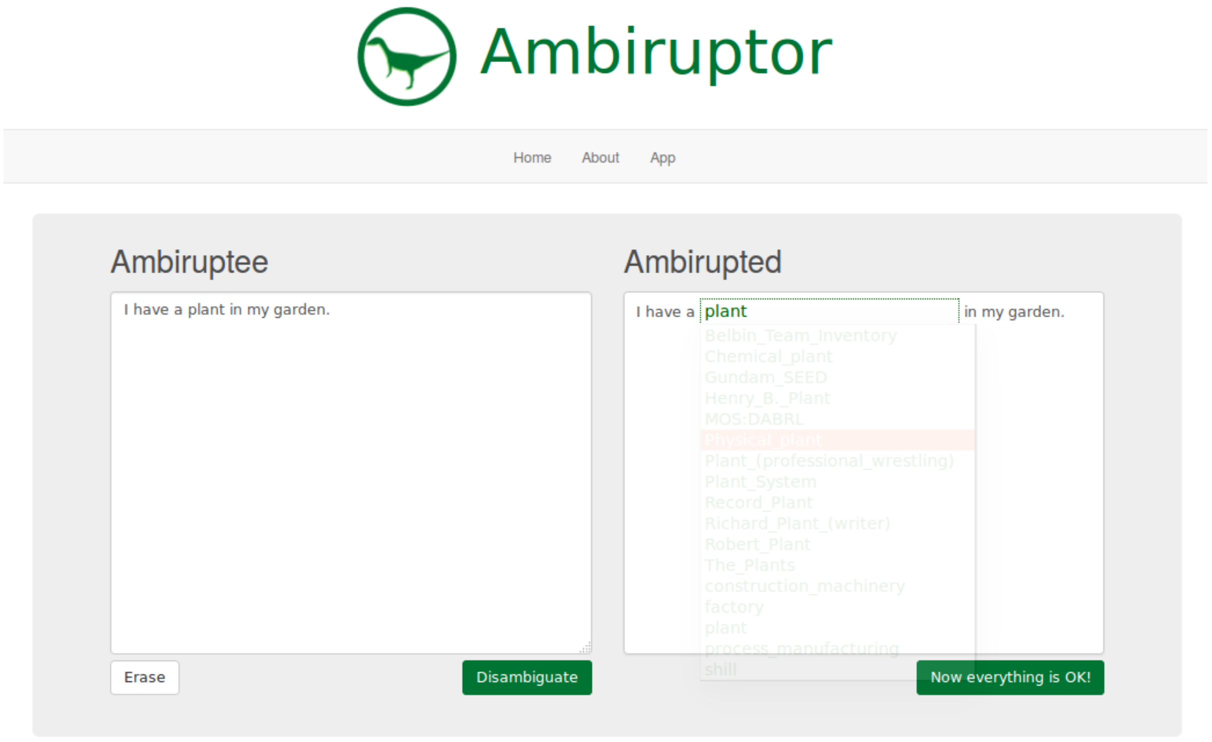
We tried to develop several user-friendly interfaces to allow people to test our disambiguation tool. One back-end has been implemented using Python and the micro-framework `Flask`. Applications follow a *fat-client* paradigm, it means that almost all functionalities are provided by the front-end. Requests are sent to the server using http protocol, then JSON containing disambiguated words is returned to the client.

3.5.1 Web Application

Our web-app can be found at the url `http://37.187.123.90:5000/`. We added a check-mode that allows people to contribute to the efficiency of our tool. Whenever the guessed sense of a word is wrong, users have the possibility to report the error and choose the correct definition of the ambiguous word. The server then logs the corresponding sentence into a database. We manually update the corpus and re-train our models to take those contributions into account. Figure 3.3 is a screenshot of the Web-App. We used HTML, CSS and `Bootstrap` (a JS framework) so that the Web-App could be used on several platforms (mobiles, tablets).

3.5.2 Firefox Plugin

We tried to integrate our aplication in several browsers, especially well spread ones, such as Mozilla Firefox. The user would select the word to disambiguate, and using the right click, then choosing "disambiguate" in the menu, he or she would be able to get the Wikipedia page corresponding to the right definition of the word, as shown in figure 3.4.

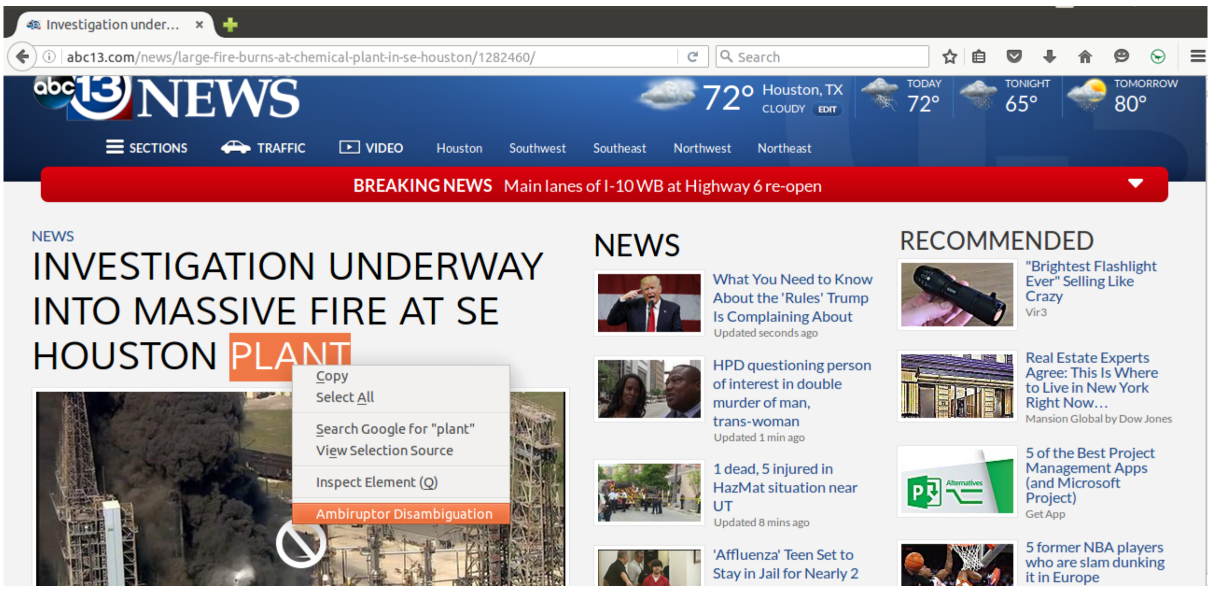


I have a plant in my garden.

I have a plant in my garden.

Belbin_Team_Inventory
Chemical_plant
Gundam_SEED
Henry_B._Plant
MOS:DABRL
Physical_plant
Plant_(professional_wrestling)
Plant_System
Record_Plant
Richard_Plant_(writer)
Robert_Plant
The_Plants
construction_machinery
factory
plant
process_manufacturing
shill

Figure 3.3: Web-app



BREAKING NEWS Main lanes of I-10 WB at Highway 6 re-open

NEWS

INVESTIGATION UNDERWAY INTO MASSIVE FIRE AT SE HOUSTON PLANT

What You Need to Know About the 'Rules' Trump Is Complaining About
Updated seconds ago

HPD questioning person of interest in double murder of man, trans-woman
Updated 1 min ago

1 dead, 5 injured in HazMat situation near UT
Updated 8 mins ago

'Affluenza' Teen Set to Stay in Jail for Nearly 2

"Brightest Flashlight Ever" Selling Like Crazy
Vir3

Real Estate Experts Agree: This Is Where to Live in New York Right Now…
Mansion Global by Dow Jones

5 of the Best Project Management Apps (and Microsoft Project)
Get App

5 former NBA players who are slam dunking it in Europe

Figure 3.4: Firefox plugin

4. Results & Applications

After working on our project for a year, we managed to get some results. We succeded in creating a user-friendly tool, easy to master for non-coders. One could think of some applications (especially in Human-Machine Interaction) of our work, and we hope that the Ambiruptor project will continue its expansion.

4.1 Examples

Some examples can be found in the figure below. It illustrates the strengths but also the weaknesses of our approach. We can notice that each typical word has an influence on the result of our algorithm.

Sentence	Guessed Sense	Explanation
This tire has a pressure of 5 **bars**	Unit, pressure	*pressure* is a typical word.
This lawyer works at the **bar**.	Law	*lawyer* is a typical word.
I'm going to drink a beer in a **bar**.	Establishment	*beer* is a typical word.
I'm going to drink a cognac in a **bar**.	City, Montenegro	No typical words → defaut answer.
A lawyer has a drink in a **bar**.	Law	*lawyer* is a typical word.

4.2 Statistics

Figures 4.1 and 4.2 contain the different scores we obtained using our tool on the words "bar" and "plant". We computed those scores using cross validation. The idea is to test our learning model on labeled data that is not in the corpus.

During model step, we tried to find the best classifier and the best parameters. We mostly used F1 score to estimate the efficiency of an model. It is an estimator that considers both precision and recall. The *precision score* is the proportion of correctly guessed samples among the samples that have been classified in one particular class; the *recall* is the proportion of correctly guessed samples among one class.

Some algorithms achieved a very good accuracy but had a poor recall. This can be explained by the fact that our corpora is not balanced at all. Indeed, if a corpus contains 90% of samples for one meaning of one word, a naive classifier can classify everything in one class. The precision score will be 90% which is not a good estimation of the efficiency of the model. When using F1 score and therefore recall, we can fix this bias.

We chose to use Support Vector Machine with a linear kernel. This result confirms what we had read during the research part on the state of the art. We can compare our scores to those obtained by the pre-existing disambiguation tools. When they achieved a 83.12% accuracy on the word "bar" using a manually labeled corpus (see [2]), we get a score of 57.72% using an automatically build corpus.

Those first results are satisfactory as there are a lot of possible improvements.

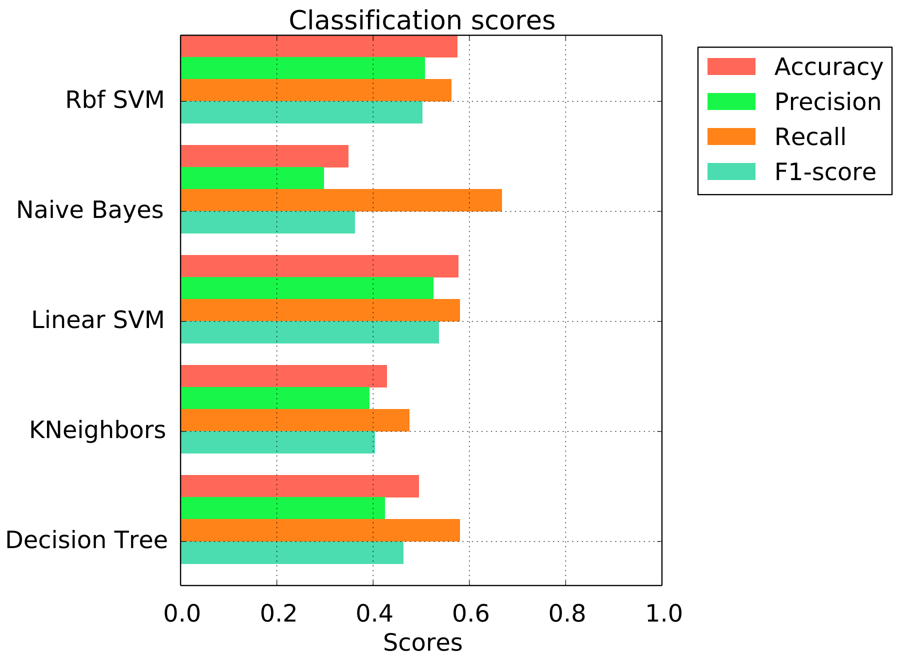


Figure 4.1: Scores for the word “bar”

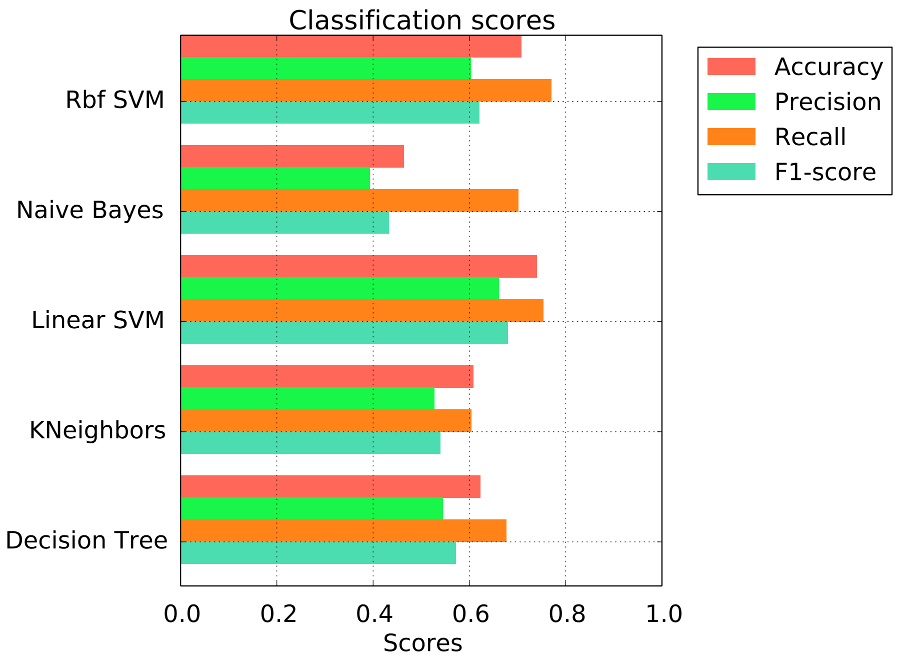


Figure 4.2: Scores for the word “plant”

4.3 Conclusion

Our disambiguation tool is still a prototype and is therefore not completely functional. However, everything is now ready to deploy at a bigger scale. With more resources, we could distribute computations and be able to disambiguate more and more ambiguous words using more and more features.

This project has been a great opportunity for all of us to discover Natural Language Processing. We worked on a concrete problem that involved several domains such as Machine Learning or Data Mining, with many applications (e.g. machine translation, semantic parsing). All of us had fun working on this project, therefore we would like to thank our supervisors, Olga Kupriianova and Eddy Caron.

Bibliography

[1] Hwee Tou Ng and Hian Beng Lee. Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 40–47. Association for Computational Linguistics, 1996.

[2] Rada Mihalcea. Using wikipedia for automatic word sense disambiguation. In *HLT-NAACL*, pages 196–203, 2007.