Ambiruptor

The Lexical Ambiguation Interruptor

Mid-term Report

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

December 22, 2015

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 determine the actual meaning of each ambiguous word. To this end we use Wikipedia, and more specifically its internal links, in order to produce an annotated corpus from which a machine learning framework is developed. At the present time, we are clear about our objectives and we have chosen the abstract design of our future code. Furthermore, the coding part is now well advanced as we achieve mining Wikipedia and have already implemented several feature extractors.

Contents

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, and its separation from other possible meanings. Since the 1940s, this problem has proved its difficulty and the lack of database has forced people to label each word manually.

Nowadays, the Internet creates new possibilities to get sufficiently big databases and the use of new machine-learning methods 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

1.1 Ambiruptor project

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 will be based on several supervised machine learning concepts. 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 Work team

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

1.3 Work advancement

In the project proposal, we presented the project split in several steps:

- Research
- Design & Implementation (API)
- Test (evaluation of performances)
- Integration (plugins & apps)

We are currently working on the implementation part. The decisions made during the research and design parts are listed below.

1.4 Following us

You can follow us using our website (`http://ambiruptor.github.io`) or contact us by email at `ambiruptor@ens-lyon.fr`.

2 Research

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

2.1 Natural Language Processing

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 so-called *features*, that are key points to look for in the input sentence. The disambiguation cannot be done without these. Features that can be considered are part-of-speech labelling, morphological forms identification and frequency considerations (see [**?**]).

2.2 Machine Learning

In order to solve the Word Sense Disambiguation problem, the following methods are usually considered:

- Dictionary-based methods
- Unsupervised methods
- Supervised methods

The 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 (different languages, ...).

2.3 Our choice

There are several approaches to solve the Word Sense Disambiguation problem, many of which achieve good results. We try to choose and to implement several learning models and feature extractors to be able to compare them and to pick the best ones afterwards.

2.3.1 Learning models

The objective is to associate the right sense to an ambiguous word. We consider two possible solutions: we can either get one single model that gives the correct meaning for every word, or get one model per ambiguous word. The second option is 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.

2.3.2 Features

Using a learning model accurately requires a lot of experiments; indeed, some features are better than others. In addition to usual features used for natural language processing (part-of-speech labelling), we decided to implement the one introduced in [?], and also some features we considered important, such as the type of prepositions in the sentence (time, place, etc.).

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

Table 1: Related words for "plant"

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

2.3.3 Text corpora and data mining

The supervised learning approach for text disambiguation implies having a corpus with already labelled ambiguous words. We have two choices for getting 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 decided to choose the second one.

Manual use of Wikipedia data for disambiguation has already been done in [?]. 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 1 describes how we build a corpus to disambiguate a word.

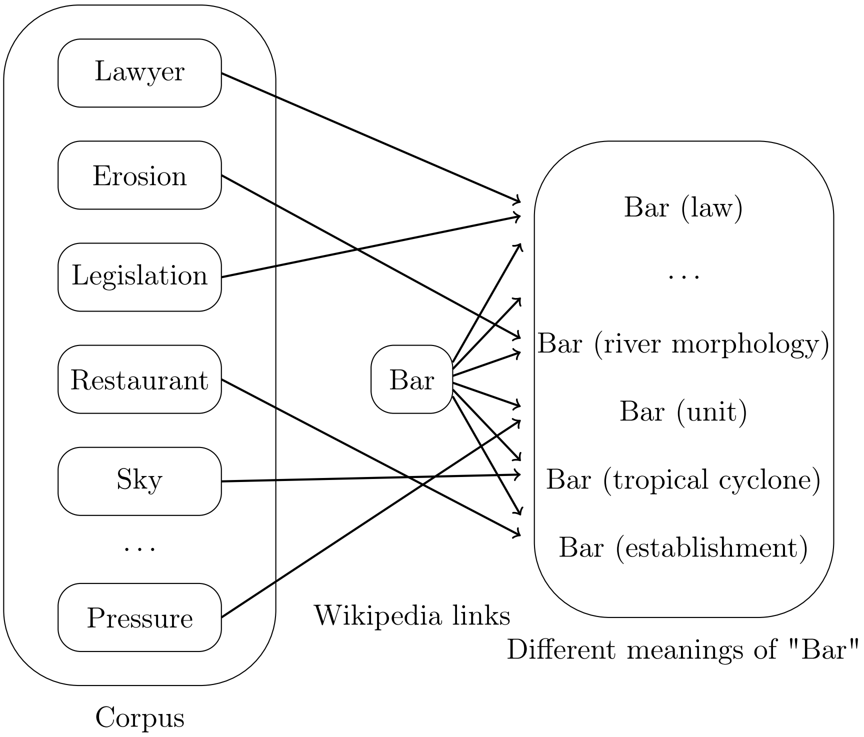

Figure 1: Corpus associated to the ambiguous word "Bar".

3 Design & Implementation

The purpose of our tool is to be used as an API for the word sense disambiguation problem. As the implementation is easier in a high level language, we decided to use `Python 3`.

3.1 Chosen tools

About implementation maybe say that we will use the models implemented in scikit learn and make the test and validation of the models, and then maybe we will try to enhance them Many open-source tools exist in Python. Here is a non-exhaustive list of those that we are going to use, among all the available packages:

- `NLTK` (Natural Language Toolkit): suite of text processing libraries for natural language processing, provides graphic demonstrations as well as sample data.
- `scikit-learn`: machine learning library equipped with algorithms for classification, regression and clustering.

Using these pre-implemented models will help us to make the test and validate our own models. After the end of the test part, we will try to enhance them.

3.2 UML Diagram

You can find the UML Diagrams of the project in the appendix (figures 2, 3 and 4)

4 Future work

Our short-term objectives include finishing implementation and testing. Later on, we will focus on integrating our device in several interfaces.

Implementation We have already implemented the major part of our tool. We now have to implement other learning methods to be able to compare their accuracy.

Tests The testing part is planned to begin by mid-January. To this end, we will need a distributed testing protocol.

Integration As soon as the testing part starts giving positive results, a small team will be assigned to develop interfaces for our tool (PDF plugin, Web app, Mobile app, ...)

A UML diagrams

For the first stage of development the design presented on the following diagrams was adopted (ellipsis on the diagrams stands for other children classes omitted).

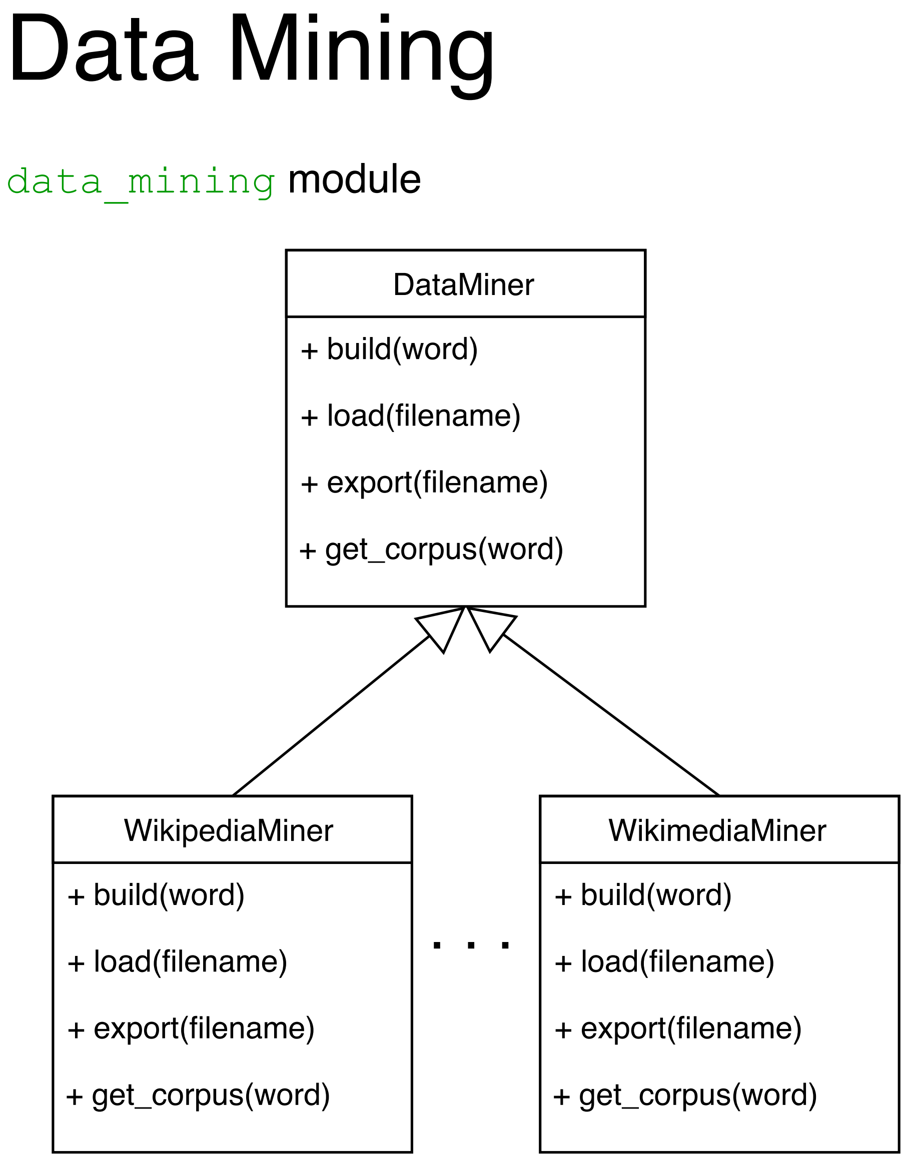

Figure 2: UML Diagram of data mining module

Feature Extraction

`feature_extraction` module

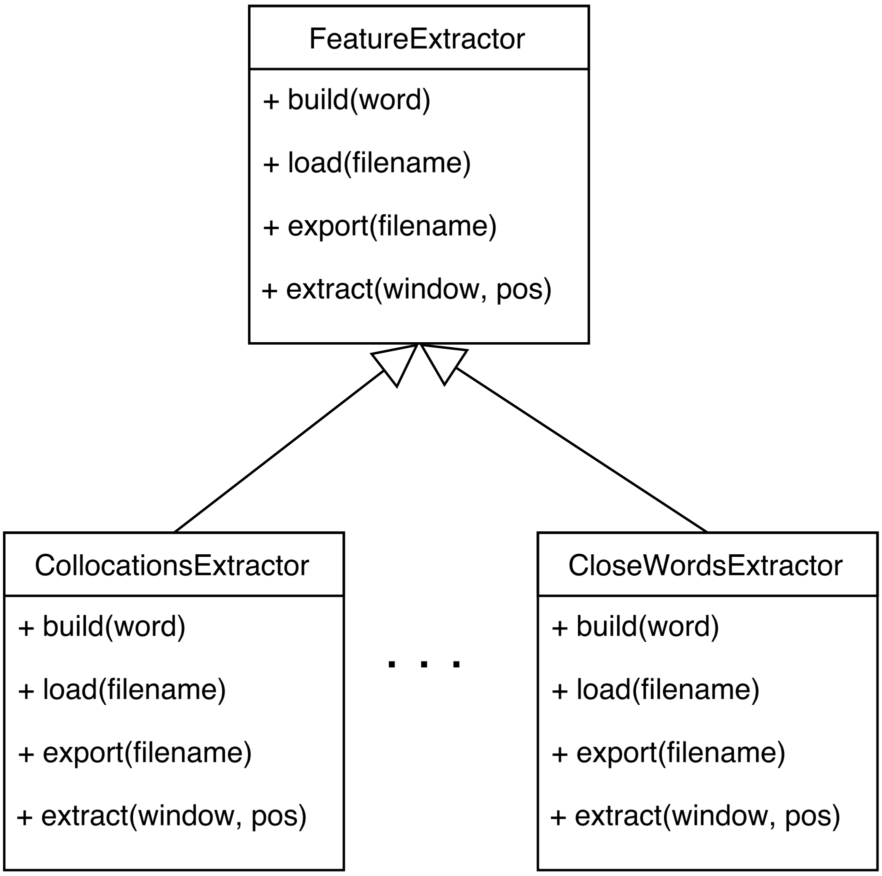

Figure 3: UML Diagram of feature extraction

Machine Learning

`learning` module

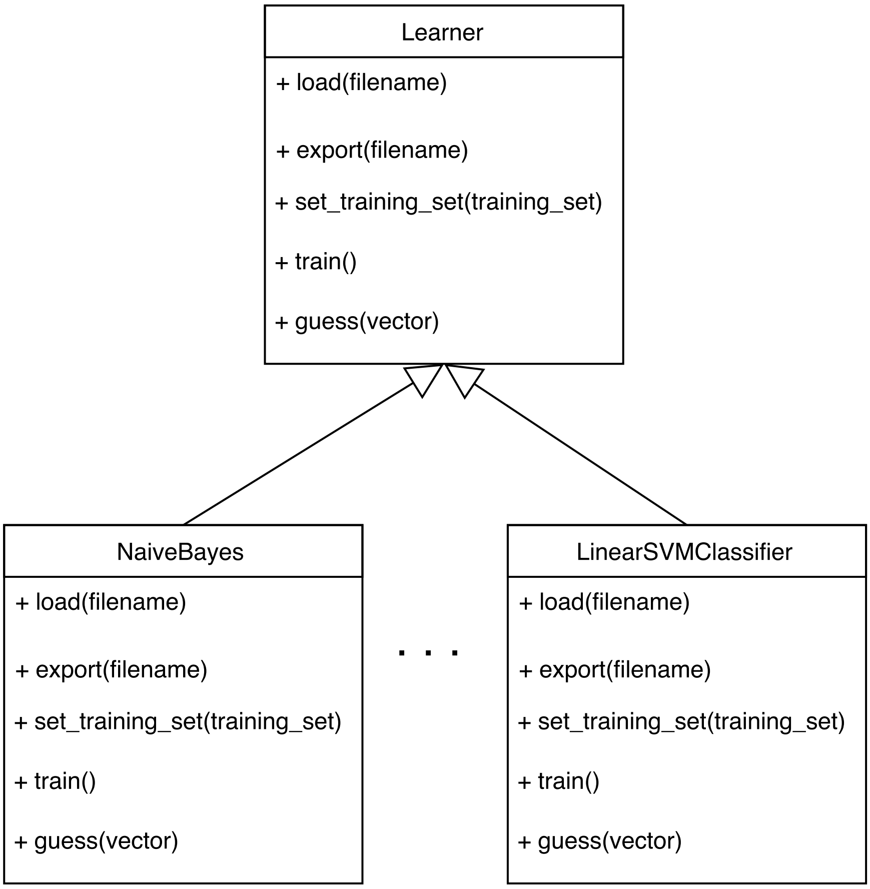

Figure 4: UML Diagram of learning module