Hypernym Detection

Abstract

In this assignment you will experience with a research paper, re-designing its algorithm for the map-reduce pattern and experimenting its quality on a large-scale input.

Your Job

1. Read the paper of Rion Snow *et al.*: Learning syntactic patterns for automatic hypernym discovery, and make sure you understand it.
2. Go carefully over the implementation notes.
3. Examine the various aspects of the system, and identify components that may utilize the map-reduce pattern.
4. Design a parallel algorithm, based on the Map-Reduce pattern.
5. Implement the system.
6. Run the experiments described bellow.
7. Analyze and report your results, as defined bellow.

The frontal check will cover the various aspects of the assignment (see the grading policy bellow).

Notes on the implementation of the article

Simplifications

We offer some quite significant easing for the implementation:

- Parsing the text (Section 2, page 2): Instead of parsing the corpus with MINIPAR, use Google Syntactic N-Grams as an input, as described bellow.
- Regarding the dependency paths, you can simply use the following definition, instead the one in the paper (the last paragraph of section 2): Given two nouns in a sentence, their dependency path is composed of the nodes and the edges (i.e., the words and the dependency labels) in the shortest path between them.
- Producing train and test sets (Section 3): You **don't** have to apply the described procedure for acquiring an annotated set from WordNet, nor to manually annotate randomly-selected pairs. Instead, use the sets we provide you - hypernym.txt - composed of word pairs and their annotations

(True - the second word is a hypernym of the first word, False - the second word is not a hypernym of the first word).

- You **don't** have to evaluate the extracted features, as described at section 4.
- You **don't** have to implement the simple classifiers, described at the second part of the second paragraph of section 5.
- You **don't** have to implement the usage of coordinate terms (Section 6).

Extensions

The words in the Google Syntactic N-Grams dataset are not *stemmed*. For example, the words *boy* and *boys* are considered different words even though they share the same root/lexeme - 'boy'. In order to improve the model, you should use a *stemmer*, which unifies suffix-based inflections of a same lexeme/root (boy-boys, walk-walks-walked). As a result, the noun pairs of you model will be stemmed nouns, and the dependency paths will be composed of stemmed words. You can use any stemmer for English you wish, for example this package.

Resources and Tools

- Here you can find the list of the part of speeches that appear in the syntactic ngrams.
- For classification, you can use WEKA or Scikit-Learn softwares. You can choose any classification algorithm you wish.

The System

Input

The input of the system is:

1. The Biarcs dataset of the syntactic NGRAM resource, released by Google (generated by Yoav Goldberg - one of the course's formers).

   The format of the datasets is described in the README file.

2. The annotated set of noun pairs.

Output

The output of the system is the Precision, Recall and F1 measures for 10-fold cross validation of the hypernym annotated set.

Input Parameters

- **DPMin** The minimal number of unique noun pairs for each dependency path. Dependency path with less distinct noun-pairs should not be considered as a feature (as described at the first paragraph of section 4). In the paper, the authors defined this parameter as 5, you should tune this parameter with refer to your input corpus.

Experiments

You should run the system with a reasonable DPMin parameter, on at least one of the 99 files (the first one). In case you have enough Amazon credit, we encourage you to extend the input of the training as much as you can. For the training of the classifier, run 10-fold cross validation on the annotated set we provide you (hypernym.txt) and calculate the Precision, the Recall and the F-measure.

Reports

Your work should be followed by four reports:

- Design

  You should provide a document which describes the design of your system: the various components of the system and their functionality in terms of input and output. In case a component is based on Map-Reduce, you should describe the characteristic of the keys and the values of the mappers and the reducers, with an estimation on the number of the key-value pairs and the memory usage.

- Communication

  The number of key-value pairs, and their size, sent from the mappers to the reducers, for each of the four runs.

- Results

  The Precision, Recall and F1 measures.

- Analysis

  Choose 5 noun-pairs for each true-positive, false-positive, true-negative, and false-negative classes. Examine the results and try to provide some explanations for the false classes, with refer to the features of their noun pairs.

Submission

You should submit the design document, the code, and the analysis report.

For the frontal check, you should come with the output of your system (including the feature vectors of the noun pairs).

During the frontal check you will be required to run the system on a very small corpus.

Grading policy

- Article understanding, with references to issues taught in class - 30%
- System design - 25%
- System implementation - 25%
- Analysis report - 20%

Good Luck!