Semantic Similarity - Classification by various measures

Submission date: 30/1

Submit your questions to Meni

Abstract

In this assignment you will experience with a research paper, modify its algorithm, re-design it for map-reduce pattern, and evaluate the results.

Your Job

1. Read the paper of Ljubešić *et al.*: Comparing Measures of Semantic Similarity (the pdf is given in the assignment's folder as well).
2. Make sure you understand the various measures of association with context and vector similarity.
3. Make sure you understand the algorithm modifications described above.
4. Examine the various aspects of the system, and identify components that may utilize the map-reduce pattern.
5. Design a parallel algorithm, based on the Map-Reduce pattern.
6. Implement the system.
7. Run the experiments described bellow.
8. Analyse and report your results, as defined bellow.

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

Algorithm Modifications

The paper compares various measures of semantic similarity. We change some of the aspects of the paper, as follows:

- Input corpus

  Instead of the Vjesnik corpus, used in the paper, we take as an input the *English All - Biarcs* dataset of Google Syntactic N-Grams, which provides syntactic parsing of Google-books N-Grams. The format of the corpus is described in the README file.

- Methods for building cooccurrence vectors (Section 1)

  Instead of a 'window' of words, the cooccurrence vector will be based on the words which are directly connected to the given word in the syntactic trees. For example, given the sentence "My dog also likes eating sausage", a window of width-1 will pick the words 'also' and 'eating' for the word 'likes', excluding the word 'dog' which is out of the window. Our syntactic-based method, in contrast, will pick 'dog' for 'likes', since there is a 'subj' edge between them.

- Features (Section 1)

  Instead of taking the lexemes of the context words as features, we define the features as pairs of words and dependency-labels, *i.e.,* the words which are connected in the syntactic tree to the given word and the label on the edges between them. For instance, the 'dog' feature of the word 'likes', in the example above, will be 'dog-subj', rather than 'dog'.

- Gold-standard Dataset (Section 2)

  We will use our own *gold-standard dataset*: word-relatedness.txt (given in the assignment's folder), composed of about 15K word pairs. Each row in this file is composed of two words and True/False indication of their relatedness.

- Classification

  The paper compares 24 combinations of context-association (equations 5,6,7,8) and vector-similarities (equations 9,10,11,13,15,17), with refer to some test set. We will go one step beyond, **building a classifier**, based on the scores of each combination:

    - For each pair of words in the gold-standard dataset, we define a 24-dimensions vector, where each entry denotes the similarity score for these two words according to one combination of association with context and vector similarity measures (4X6).
    - Given a training-set (taken from the gold-standard dataset), a classification model is trained, based on the vectors of the

word pairs in the set and their annotation (similar/not-similar).
    - Given a test-set (taken from the gold-standard dataset), the word-pairs are classified (similar/not-similar) by the classifier.

- Evaluation (Sections 2,3)

  We evaluate the trained model by applying 10-fold cross validation on the provided dataset. The final result are the Precision, Recall and F1 scores of the classifier.

Memory Assumptions

- You **can** assume that the word-pairs in the gold-standard dataset can be stored in memory.
- You **cannot** assume that a container composed of all cooccurrence vectors of the word pairs in the gold-standard can be stored in the memory.

Tools

- For classification, you can use the WEKA software (or any other software you wish). You can choose any classification algorithm you wish (in addition, the library provides an automatic cross-validation running, calculating the Precision, Recall and F1-score).
- 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 your model will be stemmed nouns, and the dependency paths will be composed of stemmed words. You can use any stemmer for English you wish, Porter Stemmer for example (Java code).

Experiments

You should run the whole process on two settings: (1) 10% of the corpus (10 file of Google syntactic n-grams); (2) 100% of the corpus (the 100 files of Google syntactic n-grams).

Reports

Your work should be followed by three reports:

- Design

  You should provide a very short 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 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 two runs (10% and 100% of the input).

- Results

  The Precision, Recall and F1 measure of each of the two runs.

Submission

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

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

Grading policy

- Article understanding - 30%
- System design - 25%
- System implementation - 25%
- Experiments - 10%
- Report - 10%

Good Luck!