Homework 10:
Relation Classification

Dr. Benjamin Roth
Computerlinguistische Anwendungen

Due: 26.6.2018, 23:59

In this homework you will implement training and prediction with a model for relation classification. To make the task easier for you, we have implemented two baseline models, one feature-based model using logistic regression, and another very simple neural network model. You can implement any model of your choice (but only use the Python packages that we used in the course). You can use parts of the baseline models, or implement everything from scratch. In any case, make sure that input and output formats correspond to the specification.

1 Task Description

We define the task of relation prediction in the following way:
Decide which relation(s) (of a fixed set of given relations) hold between two selected entities in a sentence.

Some remarks:

- We consider 39 relations defined in the TAC KBP evaluations[1]. Some examples are:
  - `per:employee_of`: Does (did) person X work for company Y?
  - `org:city_of_headquarters`: Is (was) company/organization X based in city Y?
  - `per:countries_of_residence`: Does (did) person X live in country Y?
  - `per:title`: Does (did) person X have the job title Y?
- In theory, several (or no) relations could hold between two entities. However, **our dataset is constructed in a way so that exactly one (or no) relation holds between the selected entity pairs** in the instances. Therefore, we can use a multiclass classifier (e.g. a softmax over all relations including a special output `no_relation`).

[1] https://tac.nist.gov/

The relation classifier we build in this project can be used together with a named entity recognizer to find all relations in a sentence. Consider the following example sentence: `The last remaining assets of bankrupt Russian oil company Yukos - including its headquarters in Moscow - were sold at auction for nearly 3.9 billion U.S. dollars Friday .`

A named entity recognizer might have found the following 4 entities: `Russian`, `Yukos`, `Moscow`, `U.S.`

We can then list all pairs of entities in the sentence (`Russian-Yukos`, `Russian-Moscow`, ...`U.S.-Moscow`) and solve the relation classification task for each pair as defined above.

2 Data Format

The data we use [2] contains as an instance a sentence with two pre-selected entites, and as a label the relation (or `no_relation`) that holds between them (according to human annotators). The sentence has been pre-processed so that the first relational argument (*relational subject*) is replaced by a dummy token `<SUBJ>`, and the second relational argument (*relational object*) is replaced by `<OBJ>`. Download the data from the course homepage, and unpack it into your `src/data/` folder.

The data is stored as a text file where each line corresponds to an instance, and the following fields are separated by tabulator:

1. Relational subject.
   Example:
   `Yukos`
2. Relation. $\Leftarrow$ **This is the label!**
   Example:
   `org:city_of_headquarters`
3. Relational object.
   Example:
   `Moscow`
4. Sentence with relational arguments replaced by dummies.
   Example:
   `The last remaining assets of bankrupt Russian oil company <SUBJ> - including its headquarters in <OBJ> - were sold at auction for nearly 3.9 billion U.S. dollars Friday .`

The task is therefore to predict the relation from the sentence (with the indicated relational arguments).

The sentences are already tokenized, and the tokens are separated by white space. Some special characters have been replaced in the data by their html-escape sequences,

[2] For more details about the provenance of the data see: `http://anthology.aclweb.org/D/D14/D14-1164.pdf`

e.g. `&#eacute; (é)`. To familiarize yourself with the data, look at the development file `dev_relations.tsv` in an editor.

3 Feature-based Baseline

We already provided a very simple classifier to solve this task (to some degree), have a look at `baseline_feature_relation_predictor.py`. This classifier uses logistic regression and bigram and trigram features (where tokens are marked whether the relational subject comes before the object or the other way around).

You can evaluate the accuracy of the predictions using `evaluate.py`. This classifier achieves an accuracy of 53.14%.

4 Word-vector Baseline

We also provide a skeleton implementation for using Keras and neural networks. Have a look at `baseline_neural_relation_predictor.py`. Our implementation does not do very much yet, it only learns word vectors, summarizes them using max-pooling, and predicts the output using softmax.

`evaluate.py` gives an accuracy of 42.89%.

5 Your Own Model

Implement your own model in `relation_predictor.py` and make sure that it can be called in the same way as the baseline implementations. We will train your model using the training data, and make predictions on the development data. We will evaluate your model on the resulting output using `evaluate.py`.

Depending on the accuracy that our run of your code achieves on the development data, you get the following number of points:

- **8 points** if your code achieves 53.6% or higher.
- **8 additional points** if your code achieves 54.1% or higher.
- **4 additional points** if your code achieves 55.1% or higher.
- **10 additional points** if you volunteer to present your approach in class (see course web page for details).

Your code must train within 10 minutes on a machine in the CIP Pool (AMD Phenom II X4 B95 3Ghz Processor, 8 GB RAM (remote access)). Do not forget to initialize all random seeds, if you want to have stable results. You can assume that the following packages are installed for python3.5 via pip3:

- Keras (2.0.6)

- TensorFlow (1.2.1)
- NLTK (3.2.1)
- NumPy (1.13.1/3.2.1)
- scikit-learn (0.17.1)
- SpaCy (2.0.11)

It is advised to test your code on a CIP machine to ensure that there are no problems when running the code in grading environment (CIP Pool). Please do not make use of any additional data other than the training data.

Please note: Of course it is not allowed to use any information of the development set labels for prediction (this would result in 0 points).

6 Leaderboard (just for fun)

We will also compute a leaderboard, listing team names and their performance on the **test set** (not the development set). (The test set file that you can download has the correct labels hidden). The best team will win a small price. If you want to participate in the leaderboard, overwrite the file `test_set_results_on_leaderboard.txt` with your predictions (one relation per line).