Named Entity Recognition

Machine Learning for Language Processing Project 2018/2019

1 Project Description

Named entities are phrases that contain the names of persons, organizations, locations, times and quantities. For example, in the following sentence :

U.N. official Ekeus heads for Baghdad .

We have to recognize that :

1. U.N is an Organization (class ORG)
2. Ekeus is a Person (class PER)
3. Baghdad is a location (LOC)

We will concentrate on four types of named entities : persons (PER), locations (LOC), organizations (ORG) and names of miscellaneous entities that do not belong to the previous three groups (MISC). The goal is to develp a named-entity recognition system using a deep neural network.

2 Data Description

We will use the CoNLL-2003 shared task data files. These files contain four columns separated by a single space. Each word has been put on a separate line and there is an empty line after each sentence. The first item on each line is a word, the second a part-of-speech (POS) tag, the third a syntactic chunk tag and the fourth the named entity tag. The chunk tags and the named entity tags have the format I-TYPE which means that the word is inside a phrase of type TYPE. Only if two phrases of the same type immediately follow each other, the first word of the second phrase will have tag B-TYPE to show that it starts a new phrase. A word with tag O is not part of a phrase.

The data consists of three files : one training file (about 15k sentences) and two test files testa and testb. The first test file will be used in the development phase for finding good parameters for the learning system. The second test file will be used for the final evaluation.

Data are available at :

train : `http://perso.univ-lemans.fr/~fbouga/eng.train`
testa : `http://perso.univ-lemans.fr/~fbouga/eng.testa`
testb : `http://perso.univ-lemans.fr/~fbouga/eng.testb`

3 Evaluation

Systems are evaluated on classification accuracy (the percent of labels that are predicted correctly) for every parsed phrase.

4 Project Roadmap

1. Preprocess and prepare the training data
2. Split the data (train/dev)
3. Train and evaluate a vanilla deep recurrent neural network (RNN)
4. Use the Pytorch framework to train the RNN network.
5. Optimize the model and propose enhancement (Regularization, network init, new architecture)
6. Prepare the final defense