ANLPIR 2017 Projects

Contact person: Massimo Nicosia m.nicosia@gmail.com

Quora Duplicate Detection

The following projects are within the scope of the Quora Duplicate Detection task: given a pair of questions, a machine learning model must determine if the two questions are similar or not, i.e. they have the same answer.

The dataset that we will use is a split of the Quora Duplicates dataset released by some IBM researchers. They provide train, dev, and test splits of the original dataset, and the results of their model is reported in their research paper.

More details and guidance will be given during the project execution.

I will consider providing the interested students with the baseline neural model described in the following sections, since it is common to all the projects and a common foundation to build upon can be useful. For sure, it will be written in Python and based on the Keras framework. If all the students agree, I may provide a model in pure TensorFlow.

References:
Official Quora release: https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs
IBM dataset splits: https://drive.google.com/open?id=0B0PlTAo--BnaQWlsZl9FZ3l1c28
IBM research paper: https://arxiv.org/abs/1702.03814

1) Developing strategies for handling out of vocabulary words

The project will consist in developing:

- a simple neural network for classifying two sentences
- studying and evaluating strategies to handle out of vocabulary (OOV) words

The baseline model that the student will develop will embed each sentence into a fixed size vector using the neural bag of words approach (averaging of word embeddings). The representations will be concatenated (or further processed to derive additional representations for the pair) and fed to a multi layer perceptron. Word embeddings are trained on large textual corpora. We will use GloVe vectors as our pretrained word vectors. They are distributed as a textual file containing a list of words and their corresponding numerical vector dimensions.

Not all the words in the Quora dataset are contained in the GloVe vectors vocabulary. Typically an OOV word is mapped into a randomly initialized vector.

The goal of the student is to provide a more sensible initialization for unseen words.

Many OOV words are misspellings of valid words. The following are possible ideas to treat them:

- a spellchecker can be used to fix common misspellings;
- it may happen that a misspelled word *w* in a sentence is correctly spelled in the other sentence, therefore a distance measure between *w* and all the words in the other sentence may be computed to find the closest word to match. A simpler alternative would be to extract candidate corrections from a spellchecker and check if those are present in the other sentence.
- Check if the OOV words are present in other embeddings distribution, such as word2vec Google News vectors, or Facebook FastText embeddings. The latter can also produce embeddings from subword units, and thus vectors can be generated for all the OOV words. In any case, the vectors from one model must be transformed to be compatible with the GloVe embeddings. For this purpose, a linear model will be developed to remap the dimensions of the vectors.

References:
Baseline network: https://nlp.stanford.edu/pubs/snli_paper.pdf
GloVe vectors: http://nlp.stanford.edu/data/glove.42B.300d.zip
Google News vectors: https://github.com/mmihaltz/word2vec-GoogleNews-vectors
FastText vectors: https://github.com/facebookresearch/fastText

2) Studying the effects of syntactic features

The project will consist in developing a neural network baseline as described in Project 1. The sentences will be parsed with a dependency parser and syntactic features for each word will be extracted.

The difference with the network baseline in Project 1 will be in the input vectors. Each word will be represented as the concatenation of the GloVe word vector with other vectors. The latter could be feature embeddings or embeddings of related words.

Indeed, a categorical feature of a word, e.g. its part-of-speech tag, can be mapped to a numerical vector, similar to what happens with words.

Therefore, the final representation of the word will be the concatenation of two vectors: [word vector; pos vector].

The encoding complexity may grow. For example, we may concatenate to a word vector, the embedding vector of its head in a dependency relation.

The effect of different representations will be evaluated on the final task.

Bonus points, if after successfully developing a set of different representations, the tests will be repeated using a recurrent model to encode the sentences.

3) Automatic generation of additional examples

This project will focus on generating additional examples for the task, and evaluating the generation strategy on the neural baseline described for the previous projects.

The students can propose ideas for generating examples.

Here I propose two strategies:

1. Generating new examples by substituting adjectives, verbs and nouns with synonyms
2. Exploiting the redundancies of the dataset to discover chain of similar examples, e.g. if A is similar to B, and B is similar to C, we can construct a new pair A similar to C
3. Noise can be added to embedded sentences

For strategy 1, a part-of-speech tagger can be used to select adjectives, verbs and nouns. Then, closest synonyms (that have a vector in the GloVe embeddings) can be used to replace those words. The N word to be replaced can be selected randomly, and the optimal value of N can be studied.

References:
Data augmentation technique: https://arxiv.org/pdf/1509.01626.pdf
Noise in feature space: https://openreview.net/pdf?id=HyaF53XYx