Programming Projects

Benjamin Roth, Nina Poerner, Anne Beyer

Centrum für Informations- und Sprachverarbeitung
Ludwig-Maximilian-Universität München
beroth@cis.uni-muenchen.de

Common-sense Reasoning I

- Given: Pre-trained embedding vectors for syntactic **dependency paths** between entities.
  `COUNTRY-annexes-COUNTRY`
- Task: Predict whether one dependency path implies another dependency path
  - We have manually annotated data for training/testing (4000 instances).
  - Example:
    - ★ `COUNTRY-annexes-COUNTRY` ⇒ `COUNTRY-takes-control-over-COUNTRY` (TRUE)
    - ★ `COUNTRY-annexes-COUNTRY` ⇒ `COUNTRY-neighbors-COUNTRY` (FALSE)
- Possible design choices:
  - Concatenate vectors, then predict
  - Elementwise multiplication, then predict
  - Hidden layer before prediction
  - Regularization
  - ...

Common-sense Reasoning II

- Given: Tuples (Entity-Entity, dependency-path)
- Task: Learn embedding vectors for syntactic **dependency paths** between entities.
- We have data and evaluation scripts to check how good the vectors are
- $\Leftrightarrow$ the same as word2vec!
  - Entity-Entity $\sim$ context word
  - dependency-path $\sim$ target word
  - (or vice versa)
- Possible design choices:
  - experiment with different negative sampling strategies, e.g.:
    (Russia-Crimea, COUNTRY-annexes-COUNTRY, TRUE)
    (Russia-Crimea, PERSON-married-to-PERSON, FALSE)
    ...

Antonymy Detection

- Given: Pairs of Words (from Wiktionary), which are either synonyms, antonyms, or unrelated
- Task: Use pre-trained word-vectors, and predict which pairs are synonyms, which are antonyms, or unrelated.
- Possible design choices:
  - Word2Vec, GloVe, FastText, ...
  - Concatenation, element-wise multiplication
  - hidden-layer(s)
  - qualitative analysis: confusion matrix, which type of words are easy/hard
  - ...

Relation Argument Extraction

- Given: Sentence, query, relation type
- Task: Does the sentence contain an answer for the query and relation?

  "[Haig]$_{Query}$ attended the US army academy at Westpoint ."

  `works-for` $\Rightarrow$ Answer?

  `school-attended` $\Rightarrow$ Answer?

  `born-in` $\Rightarrow$ Answer?

  ...
- I have code that predicts the answer in three different ways. ("Open-Type Relation Argument Extraction")
- How does a model perform that combines the methods for answer extraction?

Relation Prediction

- Given: Sentence with marked entity pairs
- Task: Predict which of a fixed set of relations (or no_relation) holds between entity pair?
  - *"[Haig]$_{Query}$ attended the [US army]$_{Answer}$ academy at Westpoint ."*
    `works-for` ⇒ Yes/No?
  - *"[Haig]$_{Query}$ attended the [US army]$_{Answer}$ academy at Westpoint ."*
    `school-attended` ⇒ Yes/No?
  - *"[Haig]$_{Query}$ attended the US army academy at [Westpoint]$_{Answer}$."*
    `born-in` ⇒ Yes/No?
  - ...
- Use existing noisy, automatically labelled data for training.
- Use Stanford model for prediction (PyTorch).
- Design choices:
  - Number/weight of negative instances during training
  - Learning rate
  - number of updates
  - comparison to feature-based SVM classifier
- **Access to GPU necessary**

FOFE Character-based Encodings

- FOFE (`http://www.aclweb.org/anthology/P15-2081`)
  - **Character-based word vectors**
  - Every word is represented by a weighted sum of one-hot character vectors.
  - Weights decay exponentially from start/end (1 Parameter)
- Task: Implement FOFE as a deterministic word embedding method:
  - Input: index tensor of size (batch_size x sent_length x word_length)
  - Output: Word embeddings of size (batch_size x sent_length x char_encoding)
  - Evaluate on Tagging task from ATIS/tagging sheet (or other tagging dataset, e.g. POS)
- Task variants:
  - Pytorch module **or** Keras Layer
  - Make Parameter learnable
  - Combine/compare with standard embedding layer

FOFE Character-based Encodings

- FOFE (`http://www.aclweb.org/anthology/P15-2081`)
  - **Character-based word vectors**
  - Every word is represented by a weighted sum of one-hot character vectors.
  - Weights decay exponentially from start/end (1 Parameter)
- Task: Implement FOFE as a deterministic word embedding method:
  - Input: index tensor of size (batch_size x sent_length x word_length)
  - Output: Word embeddings of size (batch_size x sent_length x char_encoding)
  - Evaluate on Tagging task from ATIS/tagging sheet (or other tagging dataset, e.g. POS)
- Task variants:
  - Pytorch module **or** Keras Layer
  - Make Parameter learnable
  - Combine/compare with standard embedding layer

Detecting insincere questions on Quora

- Problem: Trolls use Q&A forums to ask questions that are purely rhetorical. Their goal is not to get a genuine answer, but to make a point/trigger a reaction.
- Task: **Binary** intent **classification** into sincere/insincere
  - Why is Jinping the biggest serial religion murderer and rapist leader of the world after Mao? **insincere**
  - How do sex workers handle and secure their cash earnings? **sincere**
- `https://www.kaggle.com/c/quora-insincere-questions-classification`
- Task variants:
  - Implement and test at least 2-3 **architectures** (e.g., CNN, LSTM, Self-Attention, etc.)
  - Implement the search over at least 4-6 **hyperparameters** (e.g., regularization weights, dropout probability, optimizer, learning rate, etc.). The search should be intelligent and efficient (not brute force)
  - Identify and incorporate **pre-trained resources**. Suggestion: pre-trained English FastText, BERT, ELMo (see e.g. `towardsdatascience.com/elmo-embeddings-in-keras-with-tensorflow-hub-7eb6f0145440`)

Cross-lingual misogyny detection

- Training Data: 4000 **English** tweets labelled misogynous or neutal
- Test Data: 4000 **Italian** tweets labelled the same way
- Task 1:
  - Implement and compare at least 2-3 architectures that read a tweet and classify it as 1 (misogynous) or 0 (neutral).
  - Initialize your embedding layers with pre-trained cross-lingual embeddings, e.g. MUSE (`https://github.com/facebookresearch/MUSE`)
  - Train on one language, test on other.
- Task 2:
- Create a cross-lingual word-embedding space, and use it for misogyny detection
  - Least squares error linear mapping (supervised, hint: scipy.linalg.lstsq)
  - Orthogonal mapping (supervised, hint: scipy.linalg.orthogonal_procrustes)
  - VECMAP (unsupervised) https://github.com/artetxem/vecmap

Domain Adaptation: Irony Detection

- Given: One larger data set (tweets, SemEval-2018 Task 3), one smaller data set (reddit comments, Kaggle) for irony detection.
- Task: Predict whether comments are ironic (e.g. LSTM+logistic regression). How can one domain (large data set) help prediction on another domain (smaller dataset)?
- Compare different settings:
  - Add data together, give different weights to instances from data A vs. data B.
  - Pretrain with data A, continue training with data B.
  - Train model for data A. Train model for data B, constrain it to be similar to model A (*possible stand-alone topic*).
  - Effect of Dropout/SpatialDropout1D: Make model more robust by removing words from the input during training.
  - Effect of pre-trained word embeddings.

Projects proposed by participants

- (discussed with me in advance)
- Attention for text generation from structured meaning representations (Ziad Elsayes)
- Poincare Embeddings for Food Corpus (Sameh Metias)