NLP Course Assignments, 2021

Requirements

- You need to complete **5 assignments** during this course.
- The assignments need to be written in **Python** in the format of a **Jupyter Notebook**. Please “Restart & Run All“ before sending.
- Each assignment is worth **20 points** so **100 points** can be collected in total. **Explain your work** with comments, it is part of the task.
- The **deadlines** for each assignment can be seen in the schedule below. You need to send your assignment before the class starts on the given date. In case you miss the deadline, you lose **1 point** every day of that assignment.
- The **minimum score** you need to collect in order to be successful is **60 points**. If you don’t reach 60 points, you will not be allowed to go for the exam.
- If you collect at least **95 points** you will get a compliment and you can get a **better grade on the exam**.

Schedule

No	Date	Lecture	Assignment	Deadline
1	Feb 10	Introduction		
2	Feb 17	Overview of the traditional pipeline	1.1 Tokenizaton, stopword filtering	
3	Feb 24	Tokenization		
4	Mar 3	N-gram based language modeling		
5	Mar 10	Classification and sequence tagging	1.2 SPAM classifiers, 2 POS-tagger	
6	Mar 17	Dependency parsing		Assignment 1
7	Mar 24	Lexical semantics + LSA		Assignment 2
8	Apr 7	Static neural word embeddings		
9	Apr 14	RNNs and RNN-based LMs	3 Neural LM + Search engine, 4 Fasttext	
10	Apr 21	MT + seq2seq + attention		Assignment 3
11	Apr 28	Contextual word representations	5 Transformer	Assignment 4
12	May 5	Addon: Chatbots??		
13	May 12			Assignment 5

Assignment description

1. SPAM classification using BOW representation and traditional methods

- Tokenize and POS-tag the input texts with **spaCy**.
- Do POS-tag based and list-based stop-word filtering using **spaCy**.
- Build and train 3 classical ML-based classifiers on TF-IDF BOW representations of the input using **scikit-learn**'s TfidfVectorizer and compare their performance. Models/algorithms to use:
  - a tree-ensemble (e.g., Random Forest),
  - Naive Bayes,
  - a linear model (e.g., SVM or Logistic Regression).
- ML framework to use: **scikit-learn**.

Dataset: SMS Spam Collection
http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/

2. Build a POS-tagger with a classical sequence tagging model

- Use **spaCy** for tokenization
- Define an appropriate feature template – use **spaCy** for feature extraction
- Build and train a POS-tagger with one of the classical methods discussed in the lecture (HMM, Maximum Entropy, CRF)
  - Recommended method: CRF with the sklearn-crfsuite package.

Dataset: Brown corpus with the universal tagset (available via NLTK).

3. Neural language model and toy search engine

- Build an LSTM-based language model for a given corpus.
  - Software: tokenization etc. with **spaCy**, neural modeling with **Keras**.
- Build a simple "search engine" for sentences of the corpus using the LSTM hidden states as representations.
  - Recommended software: the **Annoy** indexer/nearest neighborhood library (https://github.com/spotify/annoy)

Dataset: Brown corpus (available via NLTK).

4. Sentiment classification using the fastText neural classifier model

- Implement the classic CNN-based **fastText** text classification model in **Keras**.
- Use **spaCy** for tokenization and pretrained **fastText** embeddings.
- Use the same model but with subword tokenization and embeddings.
  - Recommended software **BPEmbed** (https://github.com/bheinzerling/bpemb)
- Compare the two model's performance.

Dataset: Stanford Large Movie Review Dataset:
https://ai.stanford.edu/~amaas/data/sentiment/

5. Sentiment classification with a pretrained Transformer

- Use a pretrained Transformer model to build a state-of-the art classifier model for the previous classification task.
  - Recommended library: **ktrain** (https://github.com/amaiya/ktrain).

Dataset: Stanford Large Movie Review Dataset:
https://ai.stanford.edu/~amaas/data/sentiment