



ITMO UNIVERSITY

NLP – Latest Trends

Google BERT and others ...

Feb 1st, 2019

Outline of this unit

- ✓ Overview of current NLP approaches
 - Intro to DL-based NLP
- ✓ Start with flair NLP framework

Classical vs deep learning-based NLP

- ✓ Classical NLP tools use different approach for different problem. Eg: HMM or CRF for POS-tagging, SVM for text classification, etc.
 - Often use a lot of hand-crafted features
 - Often include various resources like WordNet
- ✓ DL-Based NLP:
 - Often trained end-to-end (what does this mean?)
 - Downside: typically needs **a lot of** training data (for supervised ML)

Neural Networks (simple overview)

- ✓ How do they basically work?
- ✓ How do they learn?
- ✓ What architectures are there?
 - Feed-forward
 - CNN
 - RNN
 - Attention
 - ...

Pretraining – Motivation

- ✓ **Problem of DL:** needs lots of training data. One solution: Pretraining and finetuning
- ✓ **In NLP**, so far models for different task types where most **trained from scratch**, such as models for sentiment detection, question answering, etc.
- ✓ When using deep learning, usually only the **first layer** – in the form of **pretrained** word embeddings – is re-used.
- ✓ For **image**-related task, the community successfully re-uses and **finetunes** big models trained on **ImageNet** data.

Pretraining Google BERT – Motivation (2)

- ✓ → it would be helpful to also have big and complex pretrained models, which can be re-used on various NLP tasks.
- ✓ **Google BERT and others** claim to provide such functionality, which has the following advantages:
 - Less training data needed for good results, as model doesn't need to be trained from scratch
 - State-of-the-art performance

Google BERT – Usage (Idea)

- 1) Use the existing pretrained model
- 2) Add only **one** (or few) additional layers
- 3) Train the model for the given task
("Finetuning")

flair

Goal:

- We want to get a working knowledge of a state-of-the-art toolkit for NLP
- **Practical Goal in the next units:**
 - Implement a state-of-the-art multi-label text classifier
 - What is multi-label text classification?

Resources:

- **Main entry point:**
<https://github.com/zalandoresearch/flair>
- We follow the Readme.md on github, and the complete Tutorial steps “Tutorial 1 – Tutorial 9”
- **Students that are not physically attending the class
→ read and implement Tutorial 1-9 yourself!**
- **Quick intro:**
<https://www.youtube.com/watch?v=e4ItiGVbels>

Exercise: Unit 1 of Tutorial

- Create a random sentence in flair
- Add some 'ner' tags to some tokens
- Display the tokens and tags
- Add labels to the whole sentence

Exercise: Unit 2 of Tutorial

- Create a sentence that contains named entities
- Do NER tagging on the sentence, print the entities and their spans
- Additionally apply POS-tagging to the same sentence (pos-fast)
- Do tagging on any document (apply sentence splitting)
- (Do sentiment tagging on some example sentences? Big model :()

Exercise: Unit 3 of Tutorial

- Just create a stack embedding with BPE-emb, and one own of your self-trained embeddings
- Embed a sentence with it and print the token vectors

Stopped here on 23.04.

- **Repetition**
 - How are sentences represented in Flair? What are some of the methods we discussed?
 - How to use a pretrained SequenceTagger?
 - How to embed words and sentences? Which embeddings did we discuss?
 - What are (global) vs contextual embeddings?
 - DocumentEmbeddings: difference between PoolEmb. And RNNEmb?

Google CoLab

- As alternative to running code on your own machine
- Free!
- You can use it for homeworks etc.
- Gives around 12.5GB of RAM and 350GB of disk space
- You can easily install packages, etc. For example:

```
!pip install flair
```
- Works very similar to .ipynb notebooks

Next steps

- Look in parallel at flair **HTML documentation** and **GitHub** in <https://github.com/gwohlgen/misc>
- Start with HTML (Tutorial 5)

Exercise: Unit 5 of Tutorial

- Create a few sentences / documents.
- Embed them with DocumentPoolEmbeddings
- Compute **document similarity** using **numpy** / **scipy** .. not torch

- Now look at **ipynb** of Tutorial 5
- Then at **HTML** of Tutorial 6
- Then **ipynb** of Tutorial 6
- **HTML** of Tutorial 7

Exercise: Unit 7 of Tutorial – Seq labeling

- Download any of the builtin sequence labelling datasets (try to find a small one)
- Use some embedding (eg Glove)
- Train a model on it
- Predict a sentence with it!

Next steps

- Show ipynb of Tutorial 7 (classification)
- HTML of Tutorial 8
- ipynb of Tutorial 8
- HTML Tutorial 9
- **Exercise Tutorial 9: train a flair embedding on a Russian corpus (eg a book)**
- Explain homework

Character Embeddings

- **Why:** OOV words, infrequent words, misspelled words, emoticons, small amount of vectors(!), etc.
- **Have a look here:**
<https://towardsdatascience.com/besides-word-embedding-why-you-need-to-know-character-embedding-6096a34a3b10>
- Basic Character-CNN, with 70 characters
- How? See link above

BERT and CoLab

- **Have a look here:**

<https://towardsdatascience.com/how-to-do-text-binary-classification-with-bert-f1348a25d905>

- **BERT CoLab notebook:**

https://colab.research.google.com/github/google-research/bert/blob/master/predicting_movie_reviews_with_bert_on_tf_hub.ipynb#scrollTo=xiYrZKaHwV81

- **GitHub:**

- https://github.com/wshuyi/demo-text-binary-classification-with-bert/blob/master/bert_text_classification.ipynb