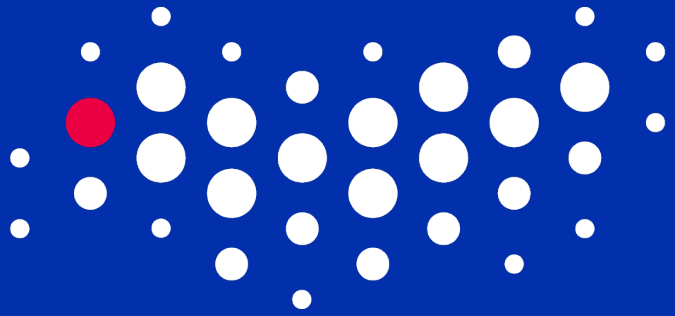

ITMO UNIVERSITY

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
- Explain homework

Unit 07.05.2019 – start with repetition

- Flair: Training of your own models – how does it work?
  - What ingredients are needed?
  - Which steps are included?
- What is hyperparameter optimization – what did we discuss?

Next steps

- HTML Tutorial 9
- Exercise Tutorial 9: train a flair embedding on a Russian corpus (eg a book)
- Switch to other content // Lexical Resources

Lexical Resources / Intro

- In the past few lectures we looked at ways to train models for NLP tasks ourselves (from raw text)
- **Traditionally** NLP also involved the usage of many resources, eg. lexical resources (dictionaries, taxonomies, etc) – which we will discuss a bit more today
- Many also see the **integration of resources** (knowledge bases, semantic data, etc) **into deep learning** as a next step to improve those models – although it is currently not clear yet how to best combine the two

Lexical Resources – NLTK

- Let’s have a look at the NLTK book, chapter 2: https://www.nltk.org/book/ch02.html
- We are now interested in **sections 4 and 5 of ch. 2.**
- Students that are not present in the course, please have look at chapter 2 / sections 4 and 5 yourself.

WordNet Exercise 1 – Simple Word Sense Disambiguation

- Select a **word** that is ambiguous, like “bank”.
- Create example sentence context(s) (not too short) to with an example usage of the word – 1 or 2 sentences.
- **Disambiguate** the **word** (using the context) to find the **best WordNet sense**
- You can for example use the Jaccard similarity of the context words and words in the definition, example sentence and lemmas.

WordNet Exercise 2 – Word Similarity / path length

- **Create a function that has 2 words as input, and returns the path length between the words.**
- This path length can later be used as a measure of word similarity
- We don’t worry about disambiguation here, just take the first sense of the word

WordNet Exercise 3

- 13 ◑ What percentage of noun synsets have no hyponyms? You can get all noun synsets using wn.all_synsets('n').

Next Unit

- Present homeworks – not necessarily all of them, but as many as possible
- We will briefly discuss some aspects, so that it's clear you understand the code / the solution