Transformers

Popular Models

Overview

- Motivations
- Architecture
- **Popular Transformer-based models: BERT, GPT, RoBERTa, etc.**

BERT and its friends: Autoencoders

- LM task: Masked Language Modeling
  - *Fly me to [MASK] moon* -> *Fly me to the moon*
- input: static embeddings + positional encodings
- 12 transformer encoder layers: Attention and FFN
  - after each layer, the vector for each word is more and more contextualized
- Linear layer for predicting input words
  - output dimensions = vocabulary size

Popular language models

- LMs are (mostly) triples of
  - Neural architecture
    - Certain (transformer) architecture
    - HPs such as No. of layers. Often: Number of parameters
  - Training data:
    - English Wikipedia? All Wikipedias? Books? Twitter?
  - Pre-training objective:
    - MLM
    - Next Sentence Prediction,
    - Next Word Prediction
    - Distillation

Popular LMs

- Some LMs are not public, or so big you can’t run them
- Frequent distinctions
    - Cased vs. uncased: Are “boooom” and “BOOOOM” the same?
    - large vs. base (vs. small): Model size, No. of params
    - Example model name: “bert-base-uncased”
- English: BERT, RoBERTa, DistilBERT, GPT, GPT-2, XLNet
- Multilingual: BERT-multilingual, XLM-RoBERTa

Model distillation

- **Why**? Big models are slow and need a lot of resources
- **Idea**: Build a small model that mimics a larger model
- **Example**: DistilBERT (Sanh et al. 2020)
  - half the size of BERT base (6 vs. 12 layers)
  - 97% of performance on downstream tasks retained
- **How?** Train a student (small LM) to behave like a teacher (large LM)
  - Uses a triple loss for the student LM
  - MLM loss
  - Minimize cos distance between final predictions of student and teacher
  - Minimize cos distance between *i*-th student layer and *(2*i)*-th teacher layer
- In practice, this can help you a lot!

Task

1. Visit 🤗 Huggingface (www.huggingface.co) and open the etherpad: http://etherpad.phil-fak.uni-duesseldorf.de/kpxxRanMSU .
2. Find some models from the previous slide. Can you also find models trained on German data?
3. Identify important model characteristics (training data, pre-training objective, neural architecture and hyperparams). The papers formally introducing the models are often linked, you can also look there!
4. Run the models with examples and look at the predictions. You can use the Inference API on the 🤗 site of the model. Can you find examples where the models perform really well or really bad?