iContextual Word Embeddings

Shubham Gupta

March 4, 2020

1 Introduction

- Word vectors: word2vec, GloVe, fasText
- Pre-trained word embeddings better than random embeddings
- Unknown words
  - Use ¡UNK¿ for OOV words
  - No way to disguish words though
  - Solution: Use character embeddings
  - At test time, unsupervised word embeddings with be bigger. Use the vector as it is.
  - Other words =¿ Assign random vector =¿ Add them to vocabulary
- Representations of word
  - Always same representation for word. Difficult to do word sense disambiguation
  - Same word can have different meanings
- Solution: **NLM**
  - In NLM, word vectors sent to LSTM layer
  - LSTM layers predict next word
  - BUT, these are context specific word representations at each position

2 TagLM "Pre-ELMo"

- Semi-supervised approach
- NLM on large unlabelled corpus
- Train word embedding model and NLM model together
- Concatenate hidden states with word embedding to sequence tagging model for NER
- Dataset: CoNLL
- TagLM was SOTA in 2017.
- Useful to have big and bidirectional LM

3 ELMo

- Breakout version of word token vectors and contextual word vectors
- Similar to TagLM
  - 2 biLSTM layers
  - Char CNN to build initial word representation
  - Larger hidden dim
  - Residual connection
- TagLM: Only top-level of neural model stack fed into the trained model
- ELMo: Use ALL LAYERS
- For a particular word, take hidden states at each level, learn weight and use that as a representation.
- Use ELMo representations for any task
- Moar performance increase
- Two biLSTM layers
  - Lower layer: Better lower-level syntax. NER, etc
  - Better for higher level sematics. QA,

4 ULMfit

- Train LM on big corpus
- Tune LM on target task data
- Fine-tune as classifier on target task
- Moar performance improvements
- Fine-tuning helped
- Scaling it up led to GPT, BERT, GPT-2, etc
- All of the new models based on **Transformers**

5 Transformers

- Annotated Transformer Havard. Link here
- **Motivation**: Parralization BUT RNN are sequential
- Maybe just use attention and remove RNN

5.1 Attention is All you need

- Non-recurrent seq-to-seq encoder decoder model
- **Dot product attention**: Use attention everywhere
  - Inputs: q and set of k-v airs
  - All vectors
  - Looking up k-v pairs. Calculate similarity based on dot product of query and key. Use this to key attention based weighting to v.

$$A(q, K, V) = \sum_i \frac{e^{q.k_i}}{\sum_j e^{q.k_j}} v_i \tag{1}$$

- Use all vectors
- **Multiple Attention Heads**: Linear proj by mutiple attention layers
- 2 sublayers
  - Multihead attention
  - 2-layer feed-forward NNet
- Allows parallelization
- Adds positional encoding as well.

6 BERT

- BiDirectional encoder representation transformer
- Mask out k percent of words. k = 15
- Perdict masked words
- Other objective is to do next sentence prediction
- **Flair** beats BERT on NER