summary_course

Daniel Saggau

2/4/2021

topics

Motivation word embeddings

Deep Learning models understand numerical representations Texts are sequences of symbolic units Step 1 in any NLP deep learning model: Turn symbols (e.g., words) into vectors

word embeddings are dense l= sparse), trainable word vectors Allows us to calculate similarities between words, e.g., with cosine similarity:

Word embeddings can be trained from scratch on supervised data: Initialize W randomly Use as input layer to some model (CNN, RNN, ...) During training, back-propagate gradients from the model into the embedding layer, and update W Result: Words that play similar roles in the training task get similar embeddings For example: If our training task is sentiment analysis, we might expect (awesome,great) ~1

Supervised training sets tend to be small (labeled data is expensive) Many words that are relevant at test time will be infrequent or unseen at training time The embeddings of infrequent words may have low quality, unseen words have no embeddings at all. We have more unlabeled than labeled data. So let's pretrain embeddings on the unlabeled data first.

Word2Vec

Mikolov et al. (2013): Efficient estimation of word representations in vector space.

distributional hypothesis: "a word is characterized by the company it keeps"'

- Skigram

The skipgram task is to maximize the likelihood of the context words, given their center word: Expressed as a negative log likelihood loss:

- CBOW Task

The continuous bag of words (CBOW) task is to maximize the likelihood of the center words, given their context words: Expressed as a negative log likelihood loss:

- Naive softmax model
- Hierachical softmax model
- Negative sampling model

Instead of predicting a probability distribution over whole vocabulary, predict binary probabilities for a small number of word pairs. This is not a language model anymore..... but since we only care about the word vectors, and not the skip gram/CBOW predictions, that's not really a problem.

- FastText

Even if we train Word2Vec on a very large corpus, we will still encounter unknown words at test time

Extension of Word2Vec by **Bojanowski et al. (2017):** Enriching Word Vectors with Subword Information.

Design choice: Fine-tune embeddings on task or freeze them?

- Pro fine-tuning: Can learn/strengthen features that are important for the task
- Pro freezing: We might overfit on training set and mess up the relationships between seen and unseen words
- Both options are popular

Applications of pre-trained word embeddings

CNN

An architecture is an abstract design for a neural network

Examples of architectures: * Fully Connected Neural Networks * Convolutional Neural Networks (CNNs) * Recurrent Neural Networks (RNNs) * Transformers (self-attention)

The choice of architecture is often informed by assumptions about the data, based on our domain knowledge

translation invariance: The features that make up a meaningful object do not depend on that object's absolute position in the input.

sequentiality: NLP: Sentences should be processed left-to-right or right-to-left. This one is falling out of fashion, since people are replacing recurrent neural networks with self-attention networks.

Are these assumptions true?

Of course not. But they are a good way of thinking about why certain architectures are popular for certain problems. Also, for limiting the search space when deciding on an architecture for a given project.

RNN

LSTM

Attention

Transformers

Hyperparameter Optimization

Transfer Learning

BERT

Alternatives to BERT

T5

XLNet

Benchmarks

Multilingual models