Natural Language Processing using Deep Learning

Chapman Siu

August 2017

Agenda

1. Preliminaries (why?): differing representations
   - No Free Lunch Theorem
2. Understanding When to use Word2Vec
   - Term frequency
   - Topic models
   - Vectors?
3. Building Word2Vec from scratch
   - Unsupervised Learning
4. Coming back to CNNs. . . FastText model
   - Transfer Learning?

1. Motivation

To understand the different parameters in word2vec models:

- What is skip-gram or CBOW?
- What is negative sampling?

This session is not...

- Detailed introduction to Neural Networks
- About deep learning

... though hopefully you will learn a bit about these things

(let me know if you want more theoretical sessions)

1. No Free Lunch

(AI theory)

> *There is no representation/algorithm/model that will outperform all other algorithms for any problem (paraphrased)*

Sometimes term frequency/topic models/word2vec models are better, other times they are not.

1. Consequences of No Free Lunch (NFL)

Several Scenarios I recently encounted:

- I have running in production a model generating scores off 50 features
- I have new data coming in (50 features) which are shown to be equally predictive

What do I do?

1. Combine all features (100 features) and rebuild the model
2. Build a model using only the new 50 features and do an ensemble
3. ???

2. Understanding Word2Vec

Bag of words/Topic models/word2vec all aim to convert: word(s) to numbers ==> usefulness to a machine.

- Term frequency: word counts, can be normalised (TFIDF)
- Topic model: vector represents distribution of words, i.e. association to a particular topic (supervised or unsupervised)
- Word2Vec: some arbitary vector in some vector space???

2. Understanding Word2Vec

Vectors allow you to measure things:

- How close vectors are (how similar)
- Are vectors orthogonal to each other (dissimilar)
- Can do "arithmetic"!

Examples from the original paper:

```
vec(King) - vec(man) + vec(woman) = vec(Queen)
```

...and many other examples.

2. Understanding Word2Vec

Differences with other approaches:

- Context!

Word2Vec considers context of a word in its construction. The 2 approaches in "converting" the unsupervised problem to a supervised one:

- skip-gram: `Pr(context|target word)`
- continuous bag of words (CBOW): `Pr(target word|context)`

3. Building Word2Vec from Scratch (Building Training Set)

Skip-gram/CBOW

Training set construction:

1. Pick window size (odd number)
2. Extract all tokens based on this chosen window size
3. Remove the middle word in each window; this becomes your target word, other words are your context

3. Building Word2Vec from Scratch (Building Training Set)

Skip-gram (window size 3)

The cat sat on the mat

window size of 3:

- the cat sat
- cat sat on
- sat on the
- on the mat

3. Building Word2Vec from Scratch (Building Training Set)

Skip-gram (window size 3)

The cat sat on the mat

window size of 3:

```
context: the sat,  target: cat
context: cat on,   target: sat
context: sat them, target: on
context: on mat,   target: the
```

Now we can perform some supervised learning!

3. Building Word2Vec from Scratch

With this frame work we can create a (shallow) neural network

$$Pr(context|targetword)$$

3. From Word2Vec to FastText

1. Replace single words with "phrases" (ngrams)
2. We have a bag of ngrams
3. Repeat word2vec. . .

. . . if supervised

4. Add layer for output

4. CNN

Architecture:

1. Add convolution filter (1 dimension)
2. Pooling? (yes/no)
3. Add dense layer to output