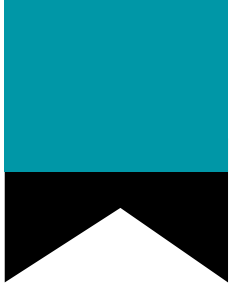
Learning to represent text in intelligent vector space



Goals and Approaches

- Representation of sentences as dense vectors using different archaic and modern word representations
- Comparing the representations on two standard nlp classification tasks
- Explore Benefits of trained embeddings, especially domain specific embeddings by showing results on a task.

Experiments performed

- Bag of Words
- Mean of word2vec
- tf-idf weighted mean of word2vec
- Doc2vec
- Weighted co-occurrence matrix with svd

Sentiment Analysis

- Trained following classifiers on Imdb dataset
- Multilayer Perceptron
- 1D CNN
- SVM classifier

Observations

- Bag of words model is still a state-of-art representation for IMDB dataset as most of the reviews sentiment doesn’t depend on the order of words and is highly dependent on frequency of words used
- Pre-trained vectors of Google Word2Vec when just took their mean as representation they reasonably worked well with MLP and CNN but failed with svm which means svm couldn’t fit in the existing mean space.

Observations

- Though due to resources constraint we just considered very less vocab size and build the co-occurrence matrix they gave greater 50%(baseline) accuracies on reducing their dimension with application of SVD they gave more or less same accuracies with all classifiers.
- Task-specific embeddings as expected gave best accuracies as the embeddings are constructed so as they can be best differentiated as different classes,they are better than the pre-trained google word2vec because google word2vec are trained with diverse context and not specific to the sentiment task.

Observations

- LSTM network gave the best accuracy as it would capture the sequence of words and relatively maps words to a better space so that they can even tackle complicated test sentences.
- Topic Classification task got reasonable accuracies with BOW representation and One-hot encoding representation because topic classification doesn’t generally need sequence of words instead it needs frequent words as features.

Observations

- Topic Classification got best accuracy with LSTM network with embedding layer which again represents LSTM fills the gap of word ordering which other networks doesn’t fill.
- As expected with Tf-Idf weighted mean of vectors gave relatively better accuracies with all classifiers as each word is given the importance it should be given instead of giving equal priority to all words.

Problems faced

- Concatenating one-hot vectors and then applying svd to get better vector-space didn’t work out as the system’s crashing even with half of average sentence length.
- Couldn’t consider complete vocabulary size as it’s taking too much memory and time.
- There aren’t google word2vec for all the words so ignored words which doesn’t have word vectors.

Accuracy Table

	BOW(%)	Mean of google word2vec	Tf-Idf weighted mean of google word2vec	Mean of Co-occurrence with SVD
Support Vector Machines	82.596	55.444	80.456	64.56
MLP	82.908	80.944	80.36	65.284
CNN	84.96	81.592	80.384	66.7

Accuracy Table with task-specific embeddings:

Sentiment Task	MLP with embedding	1D CNN with embedding	LSTM with embedding
	81.34	83.84	85.7

Reuters Newswire Topic Classification Task:

Sentiment Task	BOW(%)	One-hot Encoding
MLP	81.34	83.84
CNN	79.51	77.82

Reuters Newswire Topic Classification

Task with Task specific embeddings

Sentiment Task	MLP with embedding	1D CNN with embedding	LSTM with embedding
	67.01%	69.9	97.83%