Word Vectors from Small Corpora

Eric Zelikman

Overview

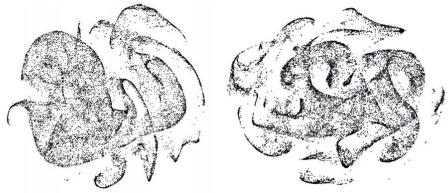
- Words define the meanings of sentences
- Sentence vectors can be well-approximated given word vectors (Zelikman 2018)
- We construct word vectors to minimize their distances to sentence vectors that contain them
- Allows for learning of word vectors from small datasets

Algorithm

- Randomly sort vocabulary, with selection probability being frequency in dataset
- For each word:
  - Initiate word randomly (normalized Gaussian)
  - Calculate sentence vectors of all the sentences where the word appears, with only initiated words
  - Bring the word closer to the normalized average of the vectors
- Proceed by selecting random words or sentences and adjusting the word vectors in each to be as close as possible to their corresponding sentence vectors (Parallelized)

Visualizations

tSNE visualization of a low-dimensional word embedding at different perplexities (10, 30, 60, and 100 respectively)



tSNE visualization of a high dimensional embedding

tSNE visualization of a high dimensional embedding when initiated randomly and then adjusted by word

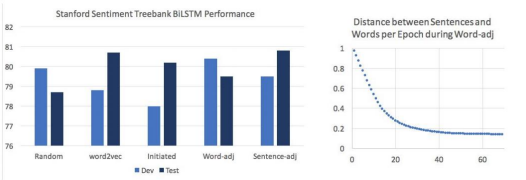
Prior Work

- There are a variety of word embedding algorithms (GloVe, word2vec, fastText, etc.)
- Broadly, words in similar contexts are assigned similar vectors
- Algorithm used to calculate sentence vectors (Zelikman 2018):

Data: $trainingSet$, a large document or transfer learning training set
Result: Returns Mahalanobis metric of training set and average of training word vectors
$vectorList = \{getVec(w_i)$ for word w_i in $trainingSet\}$;
$globalAverage = avg(vectorList)$;
$metric = MahalanobisMetric(cov(globalCovariance))$;
return $globalAverage, metric$
Algorithm 1: Training algorithm. Finds the distribution of the document's word vector cloud

Data: $sentence$: a list of words; $globalAverage$ and $metric$ from training algorithm; $globalOnly$: a boolean for whether to use the global average or the sentence's average word vector.
Result: Weights the words by their contextual importances
$vecs = [getVec(w_i)$ for word w_i in $sentence]$;
$averageVec = globalAverage$ if $globalOnly$ else $normalize(avg(vecs))$;
$distanceList = [metric.distance(vec, averageVec)$ for vec in $vecs]$;
$distanceList[a] = avg(distanceList)$;
$weights = sigmoid(distanceList)$;
return $sum(weights_i * vecs_i$ for i from 0 to $len(vecs))$;
Algorithm 2: Sentence embedding algorithm. Returns the sentence vector in a trained context

Performance



Best BiLSTM dev model for SST classification, comparing randomly initiated vectors, word2vec (WikiNews), initiated, and then adjusted word vectors

Evaluation

- Comparison of biLSTM performance on Stanford Sentiment Treebank (SST, binary) given different word vectors as inputs
- Typical distance between words and their sentences
- Mahalanobis distance
  - Accounts for dimensional correlation and variance
  - Discourages convergence to a point
  - Measures how well-differentiated sentences are
  - Essentially constant for words

Datasets

- Reddit 2009 corpus for small dataset evaluation
- Stanford Sentiment Treebank (SST) for performance analysis

Qualitative

Sample Clusters (reddit, ~2000 sentences, k-means by dist):

starts	solution	complaint	imprison	*SST Cluster:*
becuase	beneficial	gcc	inbuilt	punchier
origins	puppies	vud	burglar	crisper
postings	hubble	dispatch	traditionally	modernizes
ever	valid	conf	exhausted	hankies
interested	proved	requiring	undeserving	skidding
intact	efforts	rebuild	industrialized	repugnant
steel	worth	upstream	tradition	feathers
				decision

Conclusion and Future Directions

- This can be useful in analyzing and understanding the evolutions of small communities
- Presumably, the latter part of this algorithm can be treated as a transfer learning approach
- More broadly, what does this imply about the way in which meanings are learned?
- Clearly, performing per-word adjustments is not ideal
- The sentence formula is differentiable as a function of the value of a word: Can this be transformed into a more explicit gradient descent algorithm?
- How can the fact that it is computationally cheap to put maximally unrelated sentence be resolved? Should it be?