Distributed Representations of Words and Phrases and their Compositionality

Agenda

- Skip gram model
- Hierarchical Softmax
- Negative Sampling
- Subsampling of Frequent Words
- Empirical Results
- Learning Phrases
- Conclusion

Skip Gram Model

- Objective :

  “find word representations that are useful for predicting the surrounding words in a sentence”

- C: size of the training context (which can be a function of the center word wt). Larger the c more accurate is the output

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

Hierarchical Softmax

- Softmax function:

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)}$$

- Hierarchical Softmax function:

  log2 (W) nodes

Negative Sampling

- Objective Each Training sample only modify a small percentage of the weights and not all of them.
- “negative” word is one for which we want the network to output a 0.
- still update the weights for our “positive” word
- Selecting 5-20 words - smaller dataset
- 2-5 words - larger dataset

Subsampling Of Frequent Words

- The vector representations of frequent words do not change significantly after training on several million examples.
- Each word wi in the training set is discarded with probability computed by the formula :

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

- where f(wi) is the frequency of word wi and t is a chosen threshold, typically around 10^−5 .

Empirical Results

- Finding analogies such as “Germany” : “Berlin” :: “France” : ?, which are solved by finding a vector x such that vec(x) closest to

  vec(“Berlin”) - vec(“Germany”) + vec(“France”)

- The task has two categories:

  -syntactic analogies (such as “quick” : “quickly” :: “slow” : “slowly”)
  -semantic analogies, such as the country to capital city relationship.

Empirical Results

Method	Time [min]	Syntactic [%]	Semantic [%]	Total accuracy [%]
NEG-5	38	63	54	59
NEG-15	97	63	58	**61**
HS-Huffman	41	53	40	47
NCE-5	38	60	45	53
The following results use 10^{-5} subsampling				
NEG-5	14	61	58	60
NEG-15	36	61	61	**61**
HS-Huffman	21	52	59	55

Learning Phrases

- To learn vector representation for phrases, find words that appear frequently together, and infrequently in other contexts.
- For example, “New York Times” and “Toronto Maple Leafs” are replaced by unique tokens in the training data, while a bigram “this is” will remain unchanged.

$$\text{score}(w_i, w_j) = \frac{\text{count}(w_i w_j) - \delta}{\text{count}(w_i) \times \text{count}(w_j)}.$$

- δ is used as a discounting coefficient and prevents too many phrases consisting of very infrequent words to be formed.
- The bigrams with score above the chosen threshold are selected as phrases

Conclusion

- The subsampling method enhances performance by 2X-10X factor.
- Negative sampling algorithm, which is an extremely simple training method that learns accurate representations specially for frequent words.
- Subsampling of the frequent words results in both faster training and significantly better representations of uncommon words.
- Learning phrases with skip-gram model is being used in many applications and show very positive results.