GloVe: Global Vectors for Word Representation

1 Citation

Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.

`http://anthology.aclweb.org/D/D14/D14-1162.pdf`

2 Abstract

We create a new word vector model that uses global word co-occurence statistics AND context windows to create word vectors that get state of the art on the analogy task.

3 Introduction

Latent semantic analysis (LSA) uses global co-ocurrence statistics and the Skipgram uses local context windows. How can we combine the two?

4 Related Work

LSA and other methods are based on factoring a large matrix into a product of low-rank matrices. Local context windows aim to predict a context (i.e. nearby words) based on a word.

5 The GloVe Model

Our model is called Global Vectors (GloVe).

In order to define this model, we begin with the assumption that we have two sets of word vectors: regular vectors and context vectors (the latter are represented with a tilde, like $\tilde{w}$). We'll consider properties that we'd like these vectors to have and represent these with math. Eventually, we will have enough properties to get an exact mathematical equation that we can use to determine the word vectors.

Let X_{ij} be the number of times word j appears in the context of word i. Then let $X_i = \sum_j X_{ij}$. Let $P_{ij} = X_{ij}/X_i$. Now, suppose word i is "ice", j is "steam", and k is "solid". In this case, we expect $P_{ik}/P_{jk} > 1$. So, we can encode this property as follows (for some function F).

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

How do we pick F? Well, to enable vector arithmetic, we can make a function of vector differences: $F(w_i - w_j, \tilde{w}_k)$. Next, we need to figure how the vector difference will interact with context vector $\tilde{w}_k$ - dot product is the cleanest way to do this: $F((w_i - w_j)^T \tilde{w}_k)$. Next, we observe that the distinction between word vectors and context word vectors is arbitrary, so we should make it possible to exchange the two without messing up our equation. This gives:

$$F((w_i - w_j)^T \tilde{w}_k) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)} = \frac{P_{ik}}{P_{jk}}$$

One choice of F that satisfies the above is $F(x) = \exp(x)$. This gives:

$$w_i^T \tilde{w}_k = \log P_{ik} = \log X_{ik} - \log X_i$$

Notice that X_i is independent of k, so we can capture it with bias terms:

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log X_{ik}$$

One issue with the above model is that all co-occurences are weighted equally. To fix this, we define a weighting function $f(X_{ij})$ and get cost function:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$$

where we have (for $x_{max} = 100$ and $\alpha = 3/4$).

$$f(x) = \begin{cases} (x/x_{max})^\alpha & x < x_{max} \\ 1 & otherwise \end{cases}$$

Interestingly, the Skipgram model can also be formulated using the cost function that we have described above.

Naively, our cost function seems to take $O(|V|^2)$ time to compute, where $|V|$ is vocabulary size, which is extremely slow. However, its compute time is actually proportional to the number of nonzero elements in X. If you assume co-occurences follow a power law distribution, then the compute time is more like $O(|C|)$, where $|C|$ is the corpus size. This makes it similar to the compute intensiveness of Skipgram.

6 Experiments

We evaluate on the word analogy dataset. Basically, the ground truth is: word a is to b as c is to d. We compute $w_b - w_a + w_c$ and find the closest (by cosine similarity) word vector. If the vector belongs to word d, then our model answers the anology correctly. We also evaluate on word similarity tasks and named entity recognition (we use a given set of features + word vectors for a 5-word window and feed them into a conditional random field).

We train on multiple corpora that come from Wikipedia, Gigaword, and Common Crawl. We set $|V| = 400,000$, $x_{max} = 100$, $\alpha = 3/4$, and a context window of 10 to the left and 10 to the right. We optimize with Adagrad with a learning rate of 0.05, We get word embedding matrices W and $\tilde{W}$ and we use $W + \tilde{W}$ as our final word vectors. Our vectors have length 300. It takes 85 minutes to populate X (single thread) and 14 minutes to do optimization (32 cores).

We set state of the art on the analogy task, beating word2vec (i.e. Skipgram) with a corpus half as large. We also beat other models on Named Entity Recognition, which shows that GloVe vectors are useful features. GloVe trains faster than word2vec.

7 Conclusion

We strike a balance between global co-occurence counts and local context windows, which allows to create GloVe word vectors that beat other word vectors on word analogies and also serve as useful features in Named Entity Recognition.