Efficient Estimation of Word Representations in Vector Space

This paper gives an overview of two new models for computing continuous vector representations of words, which can provide state-of-the-art performance at measuring syntactic and semantic word similarities. These high dimensional vectors can be used to represent connections between sets of words, and have many advantages in the field of NLP compared to previous integerized word representations. These word vectors have been historically much less efficient to work with, yet, as computing power and machine learning methods have gotten exponentially better over the last few years, we can now create word vectors with 50-100 dimensions from corpora with billions of word tokens.

In this summary, I will mostly skip over an explanation of the specific structures of these new model architectures, and instead focus on the interpretation of their results, as these will be more useful at my level of understanding. Comparing the quality of word vectors can be done intuitively by checking example words and their most similar words, to see if they could be interchanged and remain coherent in their context. Pairs of words with the same relationship can also be formed and can be used to create a type of similarity between words, like comparing base words to their superlatives (e.g. *small*, *smallest*) or finding equivalence between gendered words (e.g. *man* is to *king* as *woman* is to *queen*). Higher dimensional words vectors can even be trained to identify very subtle semantic relationships between words, such as a city and the country containing it (e.g. Italy is to Rome as Ireland is to Dublin). Identifying these sorts of semantic relationships can be incredibly useful in areas such as machine translation, information retrieval, and question answering systems.

Word vectors trained on even larger data sets with larger dimensionality will perform significantly better until a certain point, where adding more dimensions or more data will have diminishing returns. Giving the algorithm more examples of particular relationships makes it even easier to train high quality word vectors using very simple model architectures, when compared to neural networks. It is expected that these high quality word vectors will become a staple building block for many future NLP applications.

Significance

High quality word vectors trained on model architectures have been a big step forward in the field of natural language processing and can be used in a variety of word generation and classification techniques.

Limitations

These word vectors are unfortunately still not perfect, with the learned relationships in table 8 being about 60% accurate - and due to the diminishing returns of adding more dimensionality or training on bigger datasets, it seems we will need new methods of storing words or creating word vectors if we want to improve on these results in a meaningful way.

Neural Word Embedding as Implicit Matrix Factorization

Levy et al. introduces this paper by analysing the Skip-gram model with negative sampling (SGNS) used for word embeddings, a starting point for the development of NLP when developing contextual relationships. Next, a theoretical framework for the SGNS algorithm is provided and it aims to optimise the model's objective - connecting words with their associated context - by using implicit factorisation methods on a pointwise-mutual information (PMI) matrix. In order to compare word embeddings, optimisation is assigned to an 'objective' function which matches a pair of all words with all corresponding context words. Another PMI matrix is formed from the information extracted from the two previous matrices, in which each entry corresponds to the strength of the relationship between a word and its context.

A positive PMI is defined as a PMI in which all negative values are transformed to zero while the positive entries remain the same - this reduces the computational complexity, making processing more efficient. A shifted PPMI is also introduced to represent words allowing for improved results on two-word similarity tasks.

An alternative to this matrix factorisation method is given - *singular value decomposition* (SVD) - which deals better with dense and low-dimensional vectors. Lastly, a comparison is made between SGNS and the aforementioned matrix-based algorithms. In order to optimise the objective, shifted PPMI gives the closest results to the *optimal solution* of the function. When evaluating the methods on various NLP-related tasks, the two methods perform quite differently. SGNS performs well on word similarity tasks, especially when dealing with high values of k, which represents the number of negative samples, in contrast to shifted PPMI and SVD, which work better with lower values. For word analogy tasks, SGNS outperforms the rest, as it performs a *weighted* matrix factorization, giving it an edge over SVD in this regard.

Significance

This paper gives a comprehensive overview of word embedding systems, comparing multiple methods fairly in order to provide information on how best to use each of them.

Limitations

The empirical results given by this paper seemed a bit limited, being only trained on the Wikipedia corpus, leading me to question its ability to make generalised conclusions.

A Question

How is Shifted PPMI actually shifted and what advantage does that give it over standard PPMI?