Enriching Word Vectors with Subword Information
Review of the Paper

Summary:
Continuous word representation getting trained on large corpora is common but many of them ignore morphology of the word. In this paper, we are going to work on that limitation. This model is fast, allowing to train on large corpora quickly and can compute word representations that did not appear on the training data.This has actually taken continuous skip-gram model which takes into account subword information and we actually train this model on 9 different languages showing different morphologies using subword model we can actually go more in-depth into morphology and while going for optimization we need to use stochastic gradient descent on the negative log likelihood after implementation and evaluating results through by using human similarity judgement we can actually get to know about null vectors those are sisg and sisg -(Subword Information Skip Gram).

Few of the things that I found interesting in the paper are scoring function “s”.Through by model it shares the representations across words like character n-grams, and makes us to learn more about rare words.

As our model explores subwords, we also calculate valid representations for out-of-vocabulary words then we can calculate by doing sum of n-gram vectors which is useful.

English Rare Word’s dataset(RW), approach outperforms baselines while it does not on the English WS353 dataset.

Datasets that were used are wikipedia dumps in nine languages and evaluation types that were used word analogy tasks and word similarity tasks.

Word2vec actually takes for training data as contextor text so the obtained word embeddings occurs in similar contexts.
Glove is more into words co-occurrences Word embeddings of Glove depends on the probabilities of 2 words appear together.
FastText basically takes subword into account it actually solves major problems that Word2Vec cannot be solved and the process is almost similar to Word2Vec.

The loss function for FastText is binary logistic loss

$$\log\left(1+e^{-s(w_t,\,w_c)}\right)+\sum_{n\in\mathcal{N}_{t,c}}\log\left(1+e^{s(w_t,\,n)}\right)$$

Loss function for FastText is very much similar to word2vec only change in here is score function gets added.

The datasets that are used for training are wikipedia dumbs in nine different languages and the evaluation type's that were used are word similarity, word analogy, comparison to the state-of-the-art methods, analysis of the effect of the size of training data and the size of the character n-grams.