**1. Summary**

This paper introduces Paragraph Vector. It is an unsupervised algorithm learns vector representations for variable-length pieces of texts (e.g., sentences and documents), and it can predict surrounding words in sampled contexts.

In the field of text classification and clustering, fixed-length vector representation for texts provides significant input for many machine learning algorithms. Bag-of-words (BOW), as a commonly-used vector representation, cannot preserve (1) word ordering and (2) word semantics. To address these disadvantages, some neural network models represent a word with a vector concatenated or averaged with other vectors in a context. and use the resulting vector to predict the following word in the given context. The objective of this word vector model is to maximize the average log (conditional) probability of a word given its surrounding context words. And the prediction task is done with a multiclass classifier such as softmax taking as parameters output words’ unnormalized log probabilities (computed using concatenation/average of word vectors from matrix W). In practice, hierarchical softmax is utilized to speedup stochastic gradient descent. In this model, semantically similar words have similar vector representations in the resultant vector space, which preserves some information about word semantics.

To extend the neural network model from word-level to phrase-level or sentence-level representations, techniques such as weighted average of words and parse tree of a sentence are explored, while being limited by issues like losing word ordering or needing parsing. In this paper, Paragraph Vector is able to construct vector representations of input sentences of variable length without abovementioned limitations. There are 2 such models: one is based on distributed memory, and the other one is distributed bag of words (without word ordering).

Paragraph Vector using distributed memory (PV-DM) modifies the original word vector model by including matrix D of paragraph vectors as another input source of hidden layer (in addition to word matrix W). As the authors claims, a paragraph vector, as features for the corresponding paragraph, represents the “missing information” from the current context and can work as a memory of the “topic” of the paragraph. Note that a paragraph vector is only shared across contexts from same paragraph while a word vector is shared across different paragraphs. There are 2 keys stages: the first is training for parameters (including matrices W and D) on *already-seen* parameters, and the second (“inference”) is adding columns and gradient descending on paragraph matrix D for *new* paragraphs. An important advantage is that paragraph vectors are learned from *unlabeled* data and can work for tasking without enough labeled data. Fortunately, this model takes into account word order at least in a small context without creating very high-dimensional vector representations that might hurt the ability to generalize.

Alternatively, Paragraph Vector without word ordering (PV-DBOW) ignores the input context words but forces itself to predict words randomly sampled from the paragraph in the output. Similar to Skip-gram model, it is conceptually simple and space efficient (need not to store word vectors).

To conduct experiments, each paragraph vector is a *combination* of two vectors: one learned by the standard paragraph vector with PV-DM and one learned by the paragraph vector with PV-DBOW. Sentiment analysis (on Stanford Treebank (1 sentence) and IMDB (more than 1 sentences) sentiment analysis dataset) and information retrieval (identifying paragraphs as results of the same query) tasks shows that Paragraph Vector model performs well than state-of-the-art models. In addition, it is observed from experiments that (1) PV-DM is usually better than PV-DBOW, (2) concatenation in PV-DM is often better than sum, (3) window size needs cross validation, and (4) Paragraph Vector can be done in parallel at test time to be less expensive.