Paper Review Report –
1.Universal Sentence Encoder
2.Deep Contextualized word representations
3.Distributed Representations of Words and Phrases and their Compositionality

NLP - CSE 628 - HW4 – Gourab Bhattacharyya - 170048888

Section 1 Introduction:

Most of the current word embedding models are learned by solving some low-level task. We are interested in learning word embeddings and then learn sentence embeddings by solving some high-level task and see if such sentence embeddings help in achieving better performance on tasks such as for TREC (question type classification or IMDB sentiment classification.

Sentence embeddings would be learned on Google 1 Billion words dataset. Dataset is publicly available. Dataset size is 1.7 GB. This problem is hard in terms of availability of large dataset and or labelled dataset. Till now we have explored and tried training two models for learning the embeddings: DAN and RNN. We will continue with our experiment and generate high quality sentence embedding to use for downstream tasks.

Section 2 Summary of each paper:

1 Universal Sentence Encoder

Task definition:

Encoding sentences into embedding vectors that specifically target transfer learning to other NLP tasks which are efficient and result in accurate performance on diverse transfer tasks.

Comparisons are made with baselines that use word level transfer learning via pretrained word embeddings as well as baselines do not use any transfer learning.

Main objective of the paper:

Present 2 models for producing sentence embeddings that demonstrate good transfer to a number of other NLP tasks. One makes use of the transformer architecture, while the other is formulated as a deep averaging network (DAN). Both models are implemented in TensorFlow.

Perform experiments with varying amounts of transfer task training data to illustrate the relationship between transfer task performance and training set size. To show that our sentence embeddings can be used to obtain surprisingly good task performance with remarkably little task specific training data.

Discuss modeling trade-offs regarding memory requirements as well as compute time on CPU and GPU. Resource consumption comparisons are made for sentences of varying lengths.

Main motivation for the objective:

Challenge with availability of limited amounts of training data available for data hungry deep learning methods and many NLP tasks as well as high cost of annotating supervised training data.

Key idea(s):

This paper introduces the model architecture for two encoding models. One based on the transformer architecture targets high accuracy at the cost of greater model complexity and resource consumption. The other targets efficient inference with slightly reduced accuracy.

Transformer: This encoding model constructs sentence embeddings using the encoding sub-graph of the transformer architecture. Use attention to compute context aware representations of words in a sentence that take into account both the ordering and identity of all the other words.

This encoder achieves the best overall transfer task performance. However, this comes at the cost of compute time and memory usage.

Deep Averaging Network (DAN):

This encoding model makes use of a deep averaging network (DAN) whereby input embeddings for words and bi-grams are first averaged together and then passed through a feedforward deep neural network (DNN) to produce sentence embeddings.

This make use of multi-task learning whereby a single DAN encoder is used to supply sentence embeddings for multiple downstream tasks.

Technical descriptions of the key ideas:

The output of the transformer and DAN sentence encoders are provided to a task specific DNN. For the pairwise semantic similarity task, we first compute the cosine similarity of the two sentence embeddings and then use across to convert the cosine similarity into an angular distance.

$$\text{sim}(\mathbf{u}, \mathbf{v}) = \left(1 - \arccos\left(\frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}||\ ||\mathbf{v}||}\right)/\pi\right)$$

Task performance varies with varying amount of training data available for the task both with and without the use of transfer learning.

Results:

Transfer learning from the transformer-based sentence encoder usually performs as good or better than transfer learning from the DAN encoder.

The best performance on most tasks is obtained by models that make use of both sentence and word level transfer which showed in Table 2.

For smaller quantities of data, sentence level transfer learning can achieve surprisingly good task performance with increasing data size, models that do not make use of transfer learning approach the performance of the other models. This observation is captured in Table 3 in paper.

Give your opinion on the idea:

Transfer learning is more critical when less training data is available. This leads to performance improvements on many tasks.

The transformer model time complexity is O(n2) in sentence length, while the DAN model is O(n). Compute time for transformer increases noticeably as sentence length increases. In contrast, the compute time for the DAN model stays nearly constant as sentence length is increased.

The transformer model space complexity also scales quadratically, O(n2), in sentence length, while the DAN model space complexity is constant in the length of the sentence.

Conclusion:

Both the encoding models, transformer and DAN based provide sentence level embeddings that demonstrate strong transfer performance on a number of NLP tasks. The sentence level embeddings surpass the performance of transfer learning using word level embeddings alone. Models that make use of sentence and word level transfer achieve the best overall performance.

The encoding models make different trade-offs regarding accuracy and model complexity that should be considered when choosing the best model for a particular application.

2 Deep contextualized word representations

Task definition:

To model complex characteristic of word use by means of syntax and semantics and to use these across various linguistic context, we have a new type of word representation called deep contextualized word representation. Here vectors are learned using deep bidirectional language model (biLM), which is pretrained on a large text corpus. These representations can be added to existing NLP applications and can improve their state of the art.

Main objective of the paper:

Introduce a new type of deep contextualized word representation that directly addresses challenges of using vectors for complex characteristics and use vectors across linguistic contexts.

Offer new word representation derived from a bidirectional LSTM that is trained with a coupled language model (LM) objective. This is called ELMo (Embeddings from Language Models) representations.

Main motivation for the objective:

Overcome single context word vectors independent representation for each word.

Recent work using context2vec representation using LSTM to encode the context around a pivot word, has the limitation of the size of parallel corpora.

To effectively learn models for downstream tasks.

Key idea(s):

Unlike the most widely used word embeddings, ELMo word representations are functions of the entire input sentence, which were computed on top of two-layer biLMs with character convolutions as a linear function of the internal network states. This setup allows us to do semi-supervised learning, where the biLM is pretrained at a large scale and easily incorporated into a wide range of existing neural NLP architectures.

Technical descriptions of the key ideas:

Bidirectional language models: A forward language model computes the probability of the sequence by modeling the probability of tokens.

A biLM combines both a forward and backward LM. Our formulation jointly maximizes the log likelihood of the forward and backward directions:

$$\sum_{k=1}^{N} \Big(\log p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \Big)$$

ELMo: This is a task specific combination of the intermediate layer representations in the biLM.

More generally, we compute a task specific weighting of all biLM layers:

$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}$$

If the activations of each biLM layer have a different distribution, in some cases it also helped to apply layer normalization to each biLM layer before weighting.

Pre-trained bidirectional language model:

To balance overall language model perplexity with model size and computational requirements for downstream tasks while maintaining a purely character-based input representation, we halved all embedding and hidden dimensions from the single best model CNN-BIG-LSTM.

Fine tuning the biLM on domain specific data leads to significant drops in perplexity and an increase in downstream task performance.

Evaluation: NLP architectures:

The Stanford Question Answering Dataset (SQuAD): After adding ELMo to the baseline model, test set F1 improved by 4.7% from 81.1% to 85.8%, a 24.9% relative error reduction over the baseline, and improving the overall single model state-of-the-art by 1.4%.

Textual entailment: This is the task of determining whether a "hypothesis" is true, given a "premise". Overall, adding ELMo to the ESIM model improves accuracy by an average of 0.7% across five random seeds. A five-member ensemble pushes the overall accuracy to 89.3%.

Semantic role labeling: Task of answering "Who did what to whom". Adding ELMo to a re-implementation the single model test set F1 jumped 3.2% from 81.4% to 84.6%.

Analysis:

Using this downstream task improves performance over previous work that uses just the top layer, regardless of whether they are produced from a biLM or MT encoder, and that ELMo representations provide the best overall performance.

Different types of contextual information captured in biLMs using two intrinsic evaluations to show that syntactic information is better represented at lower layers while semantic information is captured at higher layers, consistent with MT encoders.

We find that including ELMo at the output of the biRNN in task-specific architectures improves overall results for some tasks.

Adding ELMo to a model increases the sample efficiency considerably, both in terms of number of parameter updates to reach state-of-the-art performance and the overall training set size.

Conclusion:

In this paper a new approach has been introduced for learning high-quality deep context-dependent representations from biLMs. Significant improvement has been seen when applying ELMo to a broad range of NLP tasks.

3 Distributed Representations of Words and Phrases and their Compositionality

Task definition:

Skip-gram model is used to for learning high-quality distributed vector representations that capture a large number of precise syntactic and semantic word relationships. Several extensions of Skip-gram model presented to increase both the quality of the vectors and training speed by means of subsampling of frequent words and using negative sampling.

Main objective of the paper:

Build up on skip-gram model for training better quality of word vectors and increasing the speed during training time.

Main motivation for the objective:

Limitation of word representation is their indifference to word order and not able to present idiomatic phrases. To overcome this, the methods in this paper find phrases in the text and learn good vector representations for millions of phrases.

Key idea(s):

Use vectors to represent the whole phrases makes the Skip-gram model considerably more expressive.

First, we identify a large number of phrases using a data-driven approach, and then we treat the phrases as individual tokens during the training.

Finally, we describe another interesting property of the Skip-gram model. We found that simple vector addition can often produce meaningful results. This compositionality suggests that a non-obvious degree of language understanding can be obtained by using basic mathematical operations on the word vector representations.

Technical descriptions of the key ideas:

Skip-gram Model: One of the benefits of this is that it does not involve dense matrix multiplication as we other model like Neural Networks Language

Modeling (NNLM) and other neural network-based models.

By doing the sampling of frequent words there is significant speedup of 2x - 10x.

We use a variant of Noise Contrastive Estimation (also known as Negative sampling) to train the model.

We work towards the phrase-based model by combining words together and then treat them as individual tokens.

In skip-gram we want to maximize the log probability:

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

And we calculate the softmax by using:

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

Hierarchical Softmax: A computationally efficient approximation of the full softmax is the hierarchical softmax. This uses a binary tree representation of the output layer with the W words as its leaves and, for each node, explicitly represents the relative probabilities of its child nodes. These defines a random walk that assigns probabilities to words.

$$p(w|w_I) = \prod_{j=1}^{L(w)-1}\sigma\left([\![n(w,j+1)=\mathrm{ch}(n(w,j))]\!]\cdot {v'_{n(w,j)}}^{\top} v_{w_I}\right)$$

Where $\sigma(x) = 1/(1+\exp(-x))$

Negative Sampling: NCE can be shown to approximately maximize the log probability of the softmax. The Skip-gram model is only concerned with learning high quality vector representations, so we are free to simplify NCE as long as vector representations retain their quality.

$$\log\sigma({v'_{w_O}}^{\top} v_{w_I}) + \sum_{i=1}^{k}\mathbb{E}_{w_i\sim P_n(w)}\left[\log\sigma(-{v'_{w_i}}^{\top} v_{w_I})\right]$$

Which is used to replace every $\log P(w_O|w_I)$ term in the Skip-gram objective

Subsampling of the frequent words: To encounter the imbalance between the rare and frequent words, we used simple subsampling approach: where 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 $f(w_i)$ is the frequency of word w_i and t is a chosen threshold, typically around 10^{-5}.

Results:

While testing on phrase-based training corpus from news data with vector dimensionality 300 and context size 5, we saw that without subsampling, the negative sampling with k = 5 and k = 15 achieved better accuracy than Hierarchical softmax. But when we down sampled using subsampling (10^{-5}) then surprisingly the HS-Huffman became the best performing method with accuracy of about 47%.

With increased data set of size 33 billion words, dimensionality of 1000, and the entire sentence for the context hierarchical softmax reached an accuracy of 72% which suggests that the large amount of the training data is crucial.

We also inspect manually the nearest neighbors of infrequent phrases using various models and it seems that the best representations of phrases are learned by a model with the hierarchical softmax and subsampling.

Conclusion:

Show how to train the distributed representation of words and phrases with skip-gram model and demonstrates that these representations exhibit linear structure that makes precise analogical reasoning possible.

Word vector can be somewhat meaningfully combined using just simple vector addition. Another approach for learning representations of phrases presented in this paper is to simply represent phrases with single tokens. Combination of these two gives a powerful yet simple way of how to represent longer pieces of text, while having minimal computational complexity.

Section 3 Discussion section:

One of the paper (Universal Sentence Encoder) that I read, gave me the idea of the different types of models available (Transformer and DAN) which provides word + sentence label embeddings on small training data. This gave us the idea of using DAN and other model RNN to use with large dataset and product high quality sentence embeddings.

Another paper (Deep contextualized word representations) gave me observations, experiment results and results of applying one of the popular embedding model ELMo (Embeddings from Language Models) and use of this in different downstream NLP architectures and also use of embedding at different locations of the NLP tasks.

Third paper (Distributed Representations of Words and Phrases and their Compositionality) gave me idea about how to build better quality embedding and increase speed during training time by means of using techniques (sub-sampling and negative sampling) described and which we can use to use to generate our high-quality sentence embeddings.