Learning Word Representations

Tarun Krishna
tarun.krishna@student.uva.nl

Dhruba Pujary
dhruba.pujary@student.uva.nl

May 17, 2018

Word Representations In this task we implemented 3 models of word representation, one trained for maximum likelihood [1]SG), and two latent variable models trained by variational inference(BSG and EA). We implemented **SG** with negative sampling (with 10 negative samples corresponding to each word context pair). We used a window of size k=2 i.e 4 context word for each central word w. Model were trained with an embedding dimension of size 150. One of the shortcomings of this type of word embeddings is that they are independent from the dynamic context in which a word occurs.

In situations where a word can have different meanings in different context, they are not able to distinguish between the different senses of the words. To tackle this issue, we implement **BSG** in which choice of context words is dependent on the context-specific latent representation of the central word. A very important component of this model is the inference network $q_\phi(z|c, w)$ (also known as an encoder), which approximates the posterior distribution $p_\theta(z|c, w)$ a complete bayesian setting as the name suggests. Finally the output from the encoder is $\boldsymbol{\mu}_q$ and $\log \sigma_2$ as in [1]. For the loss we just get re-parameterized sample from the inference model, stick that into the decoder and build the loss (negative ELBO). We use a window size k=2 in this as well.

EmbedAlign model maps to a context-specific latent representation similar to BSG but uses a foreign language(L2) to understand the meaning, or the context of a word in a language(L1). The input to the encoder are word embeddings of bi-directional LSTM(Bi-LSTM) which captures the dependencies between words in a sentence. The output is μ_i and σ_i^2 of each word(w_i) in the latent space(z_i). The decoder takes the z_i and outputs the categorical distribution over the vocabulary of the two languages. The encoder consist of consist of one Bi-LSTM and two feed forward neural network(FFNN) for each μ and σ^2 with softmax and softplus activation respectively. The input to Bi-LSTM is a embedded sentence of dimension(dim) 128 which return the embedding dim=100,that is feed into the two FFNN, each of input dim=100 and output dim=150. The decoder consist of two FFNN with softmax activation, input dim=150 and output dimension equivalant to the vocabulary size of each language.

Training details We trained SG and BSG on complete hansards data each model running for 20 epochs. We again trained all 3 models on hansards and europarl but for 10,000 sentences and 3 training epochs only, this is because Embed-Align was computationally very expensive to train. We used Adam optimizer[2] for all the models we trained. We also removed all the stop-words and words with frequency less than 4.

Model	*cos*	*Add*	*Mult*	*KL*
SG	0.227	0.224	0.223	-
BSG	-	-	-	0.225

Table 1: GAP score for SG and BSG on hansards complete data

Evaluation Task We evaluate the models on the task of lexical substitution and report their GAP scores on a set of predicted rankings to a set of gold standard rankings [3]. We used 150 dimension embedding with context size k=2, SG and BSG(location and scale). For SG apart from *cosine* we used *Add* and *Mul* heuristics as in [4] .If we go by the results discussed in [1, 5] BSG should outperform SG but in our case it looks more or less both models give the same performances on hansards Table1. Comparison of all the models is depicted in Table2. For *europarl* BSG has a very slight improvement over SG. Well these numbers do not give too much insights about model performances. We did some qualitative analysis on some random words from LST dataset in *Appendix* .Technically BSG should outperform SG because latter is independent of the dynamic context which is taken care in BSG. For embed align AER was evaluated by considering only those words that are in the vocabulary of L2. The model trained on handsard gives aer score of 0.8695 on wa test dataset. Training on a very small [2]dataset(wa) also gives a very high alignment rate(0.85) because only the seen words are reported and most of them alignments are correct. From this we can say, embed align models capture the word alignment across two language correctly provided a small vocabulary or dataset(overfitting). We couldn't test on large dataset because of computation limitation. Having a trained model on large dataset would have

[1]SG: Skip Gram, BSG: Bayesian Skip Gram, EA: Embed-Align

[2]this was just to make sure if we are doing moving the right direction.

captured more general context of word and also a large vocabulary which would have make alignment decision tough. We couldn't report the GAP score for embed align because of memory issue with the model.[3]

hansards				
Model	*cos*	*Add*	*Mult*	*KL*
SG	0.179	0.178	0.178	-
BSG	-	-	-	0.178
EA	-	-	-	
europarl				
SG	0.187	0.182	0.182	-
BSG	-	-	-	0.189
EA	-	-	-	

Table 2: GAP score for SG BSG and EA on hansards 10,000 sentences

Lesson Learnt As compared to SG which assumes word embedding is fixed across the entire text collection while BSG encodes a word-specific prior density.Also, for each occurrence of a given word we can also obtain context-specific densities: they encode uncertainty about the sense of a word given its context and correspond to posterior distributions within our model. These contest specific densities is exploited in *Lexical Substitution Task*. The embed align model captures the context similar to BSG but by using a foreign language as support without supervision.

References

[1] Arthur Brazinskas, Serhii Havrylov, and Ivan Titov. Embedding words as distributions with a bayesian skip-gram model. *CoRR*, abs/1711.11027, 2017.

[2] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[3] Kazuaki Kishida. Property of average precision and its generalization: An examination of evaluation indicator for information retrieval experiments. 2005.

[4] Oren Melamud, Omer Levy, and Ido Dagan. A simple word embedding model for lexical substitution. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing, VS@NAACL-HLT 2015, June 5, 2015, Denver, Colorado, USA*, pages 1–7, 2015.

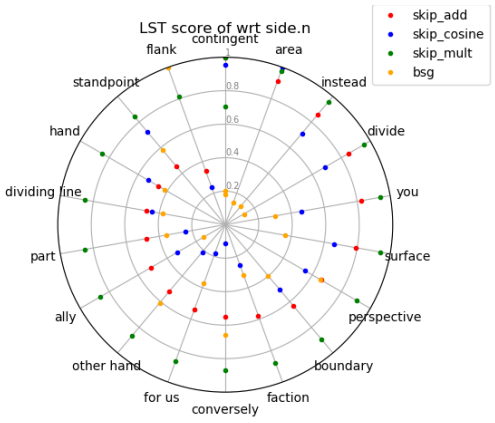
[5] Miguel Rios, Wilker Aziz, and Khalil Sima'an. Deep generative model for joint alignment and word representation. *CoRR*, abs/1802.05883, 2018.

[3]The task consumes most of the system memory and process gets killed. This last minute bug couldn't be resolved. The code needs to be optimized for large dataset which may require restructuring the networks even though they are implemented in modules. The trained models are very inefficient with memory usage.

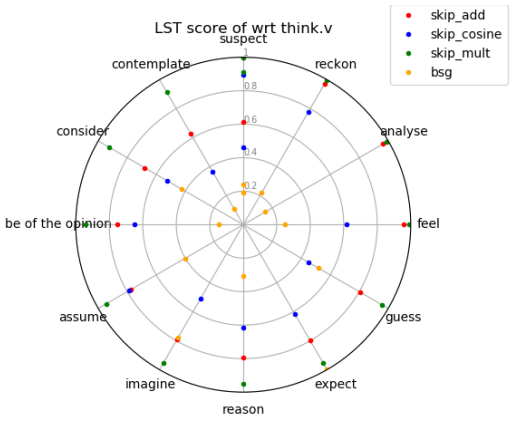
Code:
https://github.com/druv022/ULL/tree/master/Lab2

A Plots

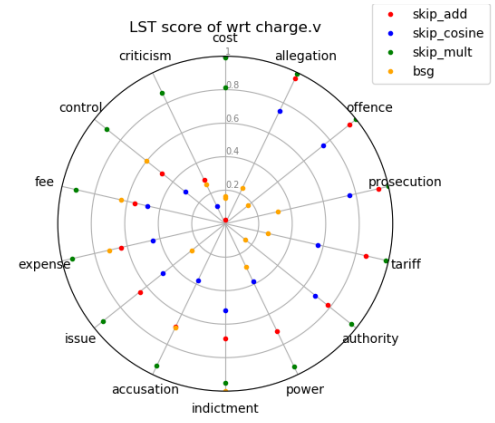
We tried to study how SG and BSG score each words for the LST. Here we didn't look at the gold standards but instead try to study how these methods ranks words depending on the context. If we take a closer look at Figure1(a) for the word *side.n* we see BSG ranks words like *divide, ally, instead, area*, while SG ranks words like *part, conversely, for us, other hand*. Likewise for word *think.n* we see words like *analyze, contemplate, be of the opinion* around the center word but SG seems to be dubious. Also one very nice example that we came across for the word *charge.n* BSG ranks *cost, offense* very close to the center word which make sense, *charge* is often used for knowing the price of items and also to charge someone with someone with misbehavior. To us this seems to be very accurate in terms of capturing context dependent meaning of the words. Also, for word *job.n* BSG gives *career* as closest but their are better alternatives for this one, on the other hand SG seem to be doubtful. For word *full.n* both of them seems to be confused. SG acts as more of point estimate that's why it fails to capture context dependent meaning.



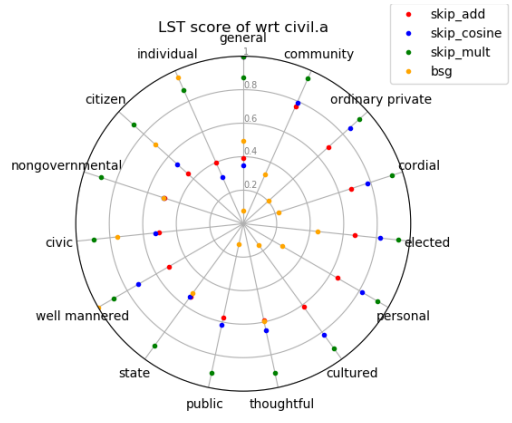
(a) side



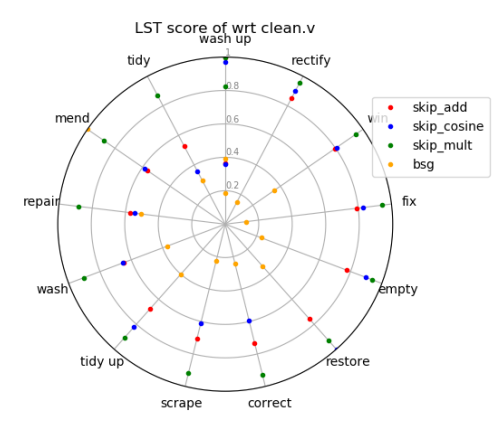
(b) think



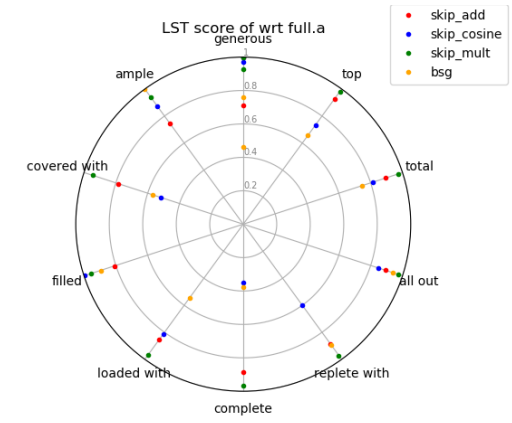
(c) charge



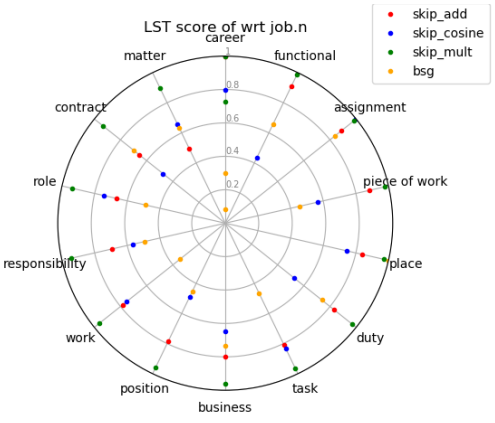
(d) civil



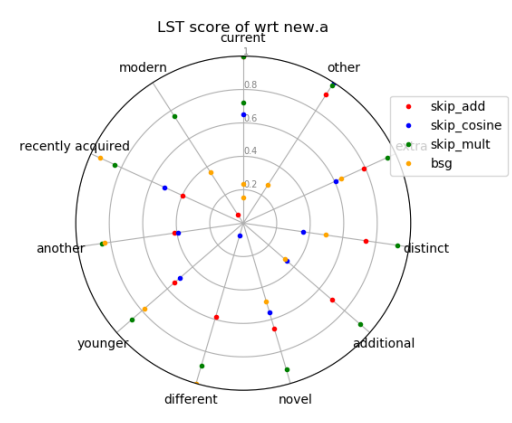
(e) clean



(f) full



(g) job



(h) new