Assignment 2: Distributional Word Embeddings

Filip Stefaniuk
filipste@student.matnat.uio.no

October 4, 2018

Introduction

This assignment had four different parts, each is described in detail in separate section of this report. To solve the problems I have implemented a number of scripts:

- `similar_wv.py` - finds nearest neighbours of the list of given words
- `eval_wv_intinsic.py` - evaluates models on intrinsic evaluation datasets
- `train_word2vec.py` - trains word embeddings model
- `train_model.py` - trains neural network model for calssifying texts
- `eval_on_test.py` - evaluates model on test data

Every script has interface with support of `--help` command, that shortly explains script and other command line arguments. Most of the results are saved in json files. The code snippets to preprocess the data and process the results are in notebooks. Scripts, notebooks, results and latex source for this report are available in github repository.

1 Basic operations with word embeddings

For this part of the assignment, I have used script `play_with_model.py` which I have slightly modified, to allow me specifiy the range of indexes of the nearest words.

To produce input file, I copied the first sentence from Chen and Manning paper, and used nltk to lemmatize, tag and remove stop words from that sentence. Results are saved in the file `data/words_auto.txt`. Unfortunately not everything was done correctly, namely one word was badly lemmatized and one had wrong POS tag. I manually corrected the mistakes and saved it as `data/words.txt`.

Finally I run the script using 2 different word models *Wikipedia* and *Gigaword*. I saved the results in `results/similar_words/`. Some of the words are presented in the tables below:

Top Words: classify_VERB

	Gigaword		Wikipedia	
11	redacted_ADJ	0.451433	reclassification_NOUN	0.502967
12	iv2_NOUN	0.446738	classified_ADJ	0.498774
13	unredacted_ADJ	0.441288	group_VERB	0.498190
14	nonpublic_ADJ	0.440909	sub-type_NOUN	0.489708
15	declassify_VERB	0.435394	label_VERB	0.484267

Figure 1: 11th to 15th nearest neighbours for word 'classify', with corresponding scores.

Top Words: current_ADJ

	Gigaword		Wikipedia	
11	latest_ADJ	0.405347	2008-2011_NUM	0.443892
12	overall_ADJ	0.402181	2011-2013_NUM	0.439801
13	original_ADJ	0.401413	past_ADJ	0.437877
14	outgoing_ADJ	0.400801	change_VERB	0.433897
15	existing_ADJ	0.397309	2014-2017_NUM	0.427117

Figure 2: 11th to 15th nearest neighbours for word 'current', with corresponding scores.

It looks like *Wikipedia* model captures the meaning of 'classify' quite well with the similar words like 'group' or 'label'. Moreover, becouse the words presented above are not the closest neighbours it is also interesting to see that the other meaning of 'classify' was captured with the words such as 'nonpublic' and 'classified'. With word 'current' another phenomena occures, while in *Gigaword* we can see words with quite similar meanining, in *Wikipedia* we observe the dates! This is not wrong, since when talking about *current* events, we often think about last days or years, but this example illustrates that with anomalies like this one it is difficult to train models, that will stay valid for a longer period of time.

2 Intrinsic evaluation of pre-trained word embeddings

I have downloaded both **SimLex999** and **Google Analogies** datasets, and I evaluated on them three different models of word embeddings. I have used builtin gensim methods `evaluate_word_pairs()` and `evaluate_word_analogies()`.

2.1 SimLex999

I fund out that, the simplest way to split **SimLex999** into sections, was to simply split dataset into separate files. Because there were no clear separators between sections I have checked the original paper and discovered, that the dataset is structured in the way that the first 111 word pairs are adjectives next 666 are nouns, and the last 222 are verbs. I split the dataset accordingly.

	40 CoNLL-17	75 Oil and Gas	82 Common Crawl
adj	0.436081	0.405065	0.570734
noun	0.410926	0.263471	0.434926
verb	0.154409	0.186691	0.157278
total	**0.361796**	**0.268299**	**0.398917**

Figure 3: Pearson correlation on SimLex999 dataset.

	40 CoNLL-17	75 Oil and Gas	82 Common Crawl
adj	0.00000	41.441441	0.00000
noun	0.00000	12.462462	0.00000
verb	0.45045	31.981982	0.45045
total	**0.10010**	**20.020020**	**0.10010**

Figure 4: Ratio of OOV words (%).

From the evaluation results we can observe that comparing verb pairs is the most difficult. It is very interesting, since the authors of the **Simlex999** mentioned in their work, that when testing word pairs with children, they also had the most difficulty with the verbs. It is also worth mentioning that even though the second model (trained on Oil and Gas data) achieved better correlation on verbs than the other two, 30% of the words were not in the vocabulary.

2.2 Google Analogies

Gensim methods for evaluation are not very flexible, and with word analogies there was no simple method to extract the information about oov tokens, that was logged but not returned as a result from function. I present the results of the evaluation below, but I want to emphesise that in case of the second model 35% of words were not in vocabulary.

	40 CoNLL-17	75 Oil and Gas	82 Common Crawl
capital-common-countries	0.776680	0.177489	0.916996
capital-world	0.722370	0.084122	0.906883
currency	0.146040	0.000000	0.206304
city-in-state	0.462910	0.022505	0.697608
family	0.863636	0.000000	0.938735
gram1-adjective-to-adverb	0.218750	0.173810	0.357863
gram2-opposite	0.280788	0.158333	0.347291
gram6-nationality-adjective	0.762351	0.479290	0.893684
Total accuracy	**0.574914**	**0.153178**	**0.740230**

Figure 5: Accuracy scores, evaluation on Google Analogies dataset.

2.3 Analysis

On both test sets the clear winner is the model trained on *Common Crawl*. There is a simple explanation for that fact, this model was trained on the largest number of data which really matters in case of word embeddings.

3 Training a word embedding model on in-domain data

I have trained two models using `train_word2vec.py`, the models were trained with skipGram algorithm, window size of 2 and vector sizes of **300** (model1) and **100** (model2). Both models were trained for 2 iterations.

3.1 Top most frequent words

I have compared the most frequent words from models trained on *SignalMedia*, *CoNLL17*, *Oil and Gas* and *Common Crawl*. The most common words in models trained on large corporas are either interpunction signs or stop words. The explanation is that the large amount of data is more difficult to process and the quality of it drops. The models achieve good results because the amount of data compensates its quality. In report I include only 10 most common but file `results/top_words.csv` contains all 20.

	SignalMedia	CoNLL17	Oil and Gas	Common Crawl
0	say_VB	</s>	lrb	the
1	year_NN	,	rrb	,
2	make_VB	the	sediment	.
3	company_NN	.	fault	and
4	also_RB	of	datum	to
5	get_VB	and	basin	of
6	new_JJ	to	sample	a
7	time_NN	a	area	in
8	include_VB	in	study	is
9	take_VB	-	model	that

Figure 6: Most common words in models

3.2 Intrinsic Evaluation

I have evaluated both models in the same manner as the models in the previous section:

	SignalMedia1	SignalMedia2
adj	0.574247	0.496217
noun	0.401974	0.377345
verb	0.320678	0.191505
total	**0.390785**	**0.340145**

	SignalMedia1	SignalMedia2
adj	0.000000	0.000000
noun	0.600601	0.600601
verb	0.900901	0.900901
total	**0.600601**	**0.600601**

Figure 7: Pearson correlation and percentage of OOV words on SimLex999 dataset.

Interestingly correlation on nouns were as good as in *Common Crawl* model, even though this model was trained on very little data. The total correlation is not really comparable, since 90% of verbs, which loweres the result were out of vocabulary. Comparing two models trained on in-domain data it can be seen that increasing vector size improves the results.

	SignalMedia1	SignalMedia2
capital-common-countries	-	-
capital-world	-	-
currency	0.000000	0.166667
city-in-state	-	-
family	0.620915	0.565359
gram1-adjective-to-adverb	-	-
gram2-opposite	-	-
gram6-nationality-adjective	-	-
Total accuracy	**0.586420**	**0.543210**

Figure 8: Accuracy scores, evaluation on Google Analogies dataset.

It is hard to get useful information from evaluating the models on analogies dataset, since almost all the classes contained words that were not in the vocabulary of this models. The only meaningful result was from evaluating on *family* section, on which model performes worse than pre-trained embeddings. The reason for that is that the domain on which the model was trained is much different from the categories in this dataset.

4 Document classification with word embeddings

4.1 Training a classifier

I have trained a model to classifiy articles from SignalMedia dataset. After couple of experiments, I have ended up with the feed forward network with 3 hidden layers of 256, 128 and 64 neurons. I have used Adam optimizer and categorical crossentropy loss function. I have tokenized a teksts, used only 3000 most common words. I decided that using all the words would provide unnecessairy noise when trainig embeddings. When using pretrained models most of these words weren't in the dictionairy either way. Then I padded the sequences to 1000. I have build embedding manually layer, gensim function since it is not compatible with keras `pad_sequences`.

I have tested the model with all the available embeddings but in most cases results were bad. I believe this is due to the fact that articles are long, and simple average of embeddings may be very similar regardless of the source of the article. When using BOW, some website may have used specific vocabulary, when using embeddings, when two different sources write similar article with different words but the same semantical information they are very difficult to distinguish. Maybe that is the reason why model with BOW had better results that the ones with word embeddings.

The best performing models that use embeddings were the one with trained embeddings during classification task and the one trained on Signal Dataset. I have tried training different embedding model with wider window but it didn't improve the results.

	29	3	40	75	82	bow	signal1	signal3	trained
4 Traders	0.433	0.452	0.389	0.413	0.437	0.631	0.609	0.586	0.591
App.ViralNewsChart.com	0.806	0.831	0.777	0.901	0.847	0.860	0.909	0.917	0.921
EIN News	0.874	0.867	0.812	0.917	0.866	0.909	0.925	0.911	0.891
Individual.com	0.656	0.648	0.602	0.612	0.645	0.742	0.693	0.681	0.712
Latest Nigerian News.com	0.466	0.544	0.433	0.557	0.570	0.653	0.600	0.518	0.486
Mail Online UK	0.712	0.661	0.666	0.595	0.699	0.696	0.716	0.702	0.766
MyInforms	0.761	0.759	0.714	0.739	0.730	0.819	0.816	0.813	0.819
NewsR.in	0.841	0.856	0.794	0.874	0.853	0.923	0.910	0.897	0.901
Reuters	0.500	0.506	0.305	0.466	0.512	0.688	0.597	0.487	0.642
Yahoo! News Australia	0.544	0.449	0.390	0.395	0.463	0.706	0.571	0.569	0.650
Average	0.659	0.657	0.588	0.647	0.662	0.763	0.735	0.708	0.738

Figure 9: F1 score for classifier trained with different word embeddings

4.2 Evaluation of the best model

For my best model I decided to train embeddings during the classification task, but use pretrained embeddings as initial values. That significantly improved results:

	f1	precision	recall
4 Traders	0.588487	0.648935	0.542500
App.ViralNewsChart.com	0.932550	0.897927	0.970438
EIN News	0.896069	0.908833	0.885790
Individual.com	0.741059	0.778867	0.707182
Latest Nigerian News.com	0.618342	0.672765	0.621064
Mail Online UK	0.773009	0.838145	0.721915
MyInforms	0.815975	0.782300	0.858885
NewsR.in	0.897730	0.915478	0.882883
Reuters	0.656083	0.673590	0.639740
Yahoo! News Australia	0.671265	0.685490	0.667910
Average	0.759057	0.780233	0.749831

	f1	precision	recall
4 Traders	0.016296	0.029568	0.045598
App.ViralNewsChart.com	0.008531	0.021226	0.008027
EIN News	0.022064	0.053669	0.009938
Individual.com	0.001001	0.013711	0.012847
Latest Nigerian News.com	0.022002	0.107470	0.160776
Mail Online UK	0.016987	0.033911	0.056718
MyInforms	0.019676	0.027051	0.070654
NewsR.in	0.008634	0.041638	0.023180
Reuters	0.012440	0.010543	0.019074
Yahoo! News Australia	0.014743	0.048674	0.070863
Average	0.006451	0.017531	0.021414

Figure 10: average results from 3 runs of the best model.

4.3 Conclusion

I believe that in this task using embeddings didn't help at all. It was difficult to even achieve results similar to those that I got using BOWs. The main reason may be that taking a simple average of embeddings is not good enough representation of the document. Better results may have been achieved with taking weighted average or using neural network encoder.