Natural Language Processing

Prodromos Kampouridis MTN2203
University of Piraeus
National Center for Scientific Research "DEMOKRITOS"
Athens, Greece

30 April 2023

Professor

Efstathios Stamatatos,

Dept. of Information and Communication Systems Engineering,

University of the Aegean

ABSTRACT

This project was assigned as part of the coursework for the Natural Language Processing course in the Master's program in Artificial Intelligence at NCSR "Demokritos" and the University of Piraeus. The assignment focuses on the concepts of pre-trained word embeddings, the implementation of text classification approaches and the evaluation of performances of various Recurrent Neural Network and Long Short-Term Memory models. The behavior and performance of these concepts are analyzed, and their results are interpreted and visually presented.

KEYWORDS

Word embeddings; word2vec; GloVe; Multinomial Naïve Bayes; Support Vector Machines; TF-IDF; word 1-grams; character 3-grams; RNN; LSTM;

A. WORD EMBEDDINGS

The first part of the assignment involves using pre-trained word embeddings such as word2vec and GloVe to find the 10 most similar words, according to the given words.

1. The 10 words most similar to the requested words, according to *word2vec*, are listed in this table below.

Word	Result
car	'vehicle', 'cars', 'SUV', 'minivan', 'truck', 'Car', 'Ford_Focus', 'Honda_Civic', 'Jeep', 'pickup_truck'
jaguar	'jaguars', 'Macho_B', 'panther', 'lynx', 'rhino', 'lizard', 'tapir', 'tiger', 'leopard', 'Florida_panther'
Jaguar	'Land_Rover', 'Aston_Martin', 'Mercedes', 'Porsche', 'BMW', 'Bentley_Arnage', 'XF_sedan', 'Audi', 'Jaguar_XF', 'XJ_saloon'
facebook	'Facebook', 'FaceBook', 'twitter', 'myspace', 'Twitter', 'twitter_facebook', 'Facebook.com', 'myspace_facebook', 'facebook_twitter', 'linkedin'

Table 1: Top 10 most similar words according to word2vec

The 10 most similar words found according to GloVe are listed in this table, respectively.

Word	Result
car	'cars', 'vehicle', 'truck', 'driver', 'driving', 'vehicles', 'motorcycle', 'automobile', 'parked', 'drivers'
jaguar	'rover', 'bmw', 'mercedes', 'sepecat', 'mustang', 'lexus', 'volvo', 'cosworth', 'xk', 'maserati'
Jaguar	Word not found in vocabulary
facebook	'twitter', 'myspace', 'youtube', 'blog', 'linkedin', 'google', 'website', 'web', 'blogs', 'networking'

Table 2: Top 10 most similar words according to GloVe

The table below contains the common words between the 2 models, for each word of the requested words.

Word	Result	Count
car	'cars', 'vehicle', 'truck'	3
jaguar	No common words	0
Jaguar	No common words	0
facebook	'twitter', 'linkedin', 'myspace'	3

Table 3: The common words between Word2vec and GloVe results

For the word 'car' both models return car related words and the three common words between the two sets are '*cars', 'truck'* and '*vehicle'*. Likewise, for the word 'facebook both models have three words in common and more specific '*myspace', 'linkedin'* and 'twitter'. The word 'Jaguar' was not found in the vocabulary according to Glove since the pre-trained word vectors from glove-wiki-gigaword-300 are all in lowercase. Therefore, it was clear than no common words could exist between the results. On the other hand, for the word 'jaguar' even though twenty results were obtained, no common words appeared.

2. The 10 most similar words to 3 words of our choice, found using *word2vec,* are listed in this table below.

Word	Result
mustang	'wild_mustang', 'mustangs', 'wild_mustangs', 'quarterhorse', 'Paso_Fino', 'horses', 'burros', 'burro', 'horse', 'appaloosa'
Mustang	'Camaro', 'Corvette', 'Ford_Mustang', 'Mustang_GT', 'SVT_Cobra', 'Camaros', 'Shelby_GT###s', 'Dodge_Challenger', 'Chevy', 'Chevy_Camaro'
bill	'legislation', 'bills', 'amendment', 'HB####', 'omnibus_bill', 'appropriations_bill', 'Legislation', 'Tax_Extenders_Act', 'Senate', 'proposed_constitutional_amendment'
bob	'bobs', 'bangs', 'backcombed', 'flicky', 'wavy_locks', 'sleek_bob', 'pixie_crop', 'auburn_locks', 'maxi_skirts', 'sideswept'

Table 4: Top 10 most similar words according to word2vec

The 10 most similar words found using *GloVe* are listed below.

Word	Result
mustang	'camaro', 'p-51', 'thunderbird', 'mustangs', 'gt', 'roadster', 'jaguar', 'f-150', 'corvette', 'fastback'
Mustang	Word not found in vocabulary
bill	'legislation', 'bills', 'passed', 'measure', 'senate', 'amendment', 'provision', 'proposal', 'amendments', 'enacted'
bob	'tom', 'billy', 'jim', 'robert', 'steve', 'mike', 'dole', 'wilson', 'thompson', 'bobby'

Table 5: Top 10 most similar words according to GloVe

The table below contains the common words between the 2 models for each word of the requested words.

Word	Result	Count
mustang	'mustangs'	1
Mustang	No common words	0
bill	'bills', 'amendment', 'legislation'	3
bob	No common words	0

Table 6: The common words between Word2vec and GloVe results

For the word ‘mustang’ both models return similar words related to horses and cars but only ‘mustangs’ is common between the two sets of similar words. Moreover, for ‘Mustang’, the word2vec model returns similar car models while GloVe does not recognize the word, hence there are no common words. For the ‘bill’ word, both models return similar words related to legislation and lawmaking, with ‘amendment’, ‘legislation’ and ‘bill’s being common. Last but not least, both models return different similar words for ‘bob’ but with no common words.

3. The table below lists the 10 words most similar to ‘student’ as determined by word2vec and according to all requested constraints.

Word	Result
student	'students', 'Student', 'teacher', 'stu_dent', 'faculty', 'school', 'undergraduate', 'university', 'undergraduates', 'semester'
student (Excluding university-related words)	'sixth_grader', 'seventh_grader', '8th_grader', 'eighth_grader', 'grader', 'kindergartner', 'kindergartener', 'Kindergartner', 'teen', 'middle_schooler'
student (Excluding elementary-middle-high school related words)	'doctoral_student', 'Graduate_Student', 'Student', 'doctoral_candidate', 'prof', 'Undergraduate_Student', 'researcher', "CSOM_'##", 'pharmacologist', 'Undergraduate'

Table 7: Top 10 most similar words to ‘student’ according to word2vec

The 10 most similar words found using *GloVe* respectively are listed below.

Word	Result
student	'students', 'teacher', 'graduate', 'school', 'college', 'undergraduate', 'faculty', 'university', 'academic', 'campus'
student (Excluding university-related words)	'15-year', '16-year', '17-year', '14-year', '13-year-old', '14-year-old', '9-year', '16-year-old', '15-year-old', '12-year-old'

student (Excluding elementary-middle-high school related words)	'22-year-old', 'premed', '23-year-old', 'twenty-year-old', 'afshari', 'demonstrator', 'farzaneh', 'suada', '21-year-old', '25-year'

Table 8: Top 10 most similar words to 'student' according to GloVe

4. The 2 most similar words found using word2vec and their similarity scores are listed in this table below.

Analogy	Result
king - man + woman	queen: 0.7118, monarch: 0.6190
France - Paris + Tokyo	Japan: 0.8168, Japanese: 0.6481
trees - apples + grapes	oak_trees: 0.6750, vines: 0.6702
swimming - walking + walked	swam: 0.6926, swim: 0.6725
doctor - father + mother	nurse: 0.7128, doctors: 0.6593

Table 9: Top 2 most similar words and the similarity scores according to word2vec

The results according to *GloVe* are listed below, respectively.

Analogy	Result
king - man + woman	queen: 0.6713, princess: 0.5433
France - Paris + Tokyo	One of the words in (France, Paris, Tokyo) not found in vocabulary.
trees - apples + grapes	vines: 0.5909, tree: 0.5843
swimming - walking + walked	swam: 0.4978, swimmers: 0.4852
doctor - father + mother	nurse: 0.6570, doctors: 0.6172

Table 10: Top 2 most similar words and the similarity scores according to GloVe

The results indicate that the analogies found by the two models are similar but not identical. For the first analogy, both models return 'queen' as the most similar word, but the second most similar word is different. For the second analogy, word2vec model returns 'Japan' and 'Japanese' as the most similar. Additionally, the GloVe model was unable to find results for the because one of the words was not found in the vocabulary. Moreover, for the third one, both models return 'vines' as one of the most similar as well as for the fourth both models return swam, respectively. Lastly, for the fifth analogy both models return 'nurse' as one of the most similar words.

5. Repeating the previous task with 3 new analogies of our choice, we can obtain the following results.

Analogy	Result
computer_programmer - man + woman	homemaker: 0.5627, housewife: 0.5105
eyes - see + hear	ears: 0.6380, ear: 0.5290
groom - man + woman	bride: 0.6800, Mohammad_Rassool_cousin: 0.6375

Table 11: Top 2 most similar words and the similarity scores according to word2vec

Analogy	Result
computer_programmer - man + woman	One of the words in (computer_programmer, man, woman) not found in vocabulary.
eyes - see + hear	ears: 0.5937, heard: 0.5364
groom - man + woman	bride: 0.6347, wedding: 0.4721

Table 12: Top 2 most similar words and the similarity scores according to GloVe

For the first analogy, word2vec returns 'homemaker' and 'housewife' as the 2 most similar words. This result reflects societal gender stereotypes and biases that associate women with householding. On the other hand, Glove is unable to find results for this set of words because one of the words was not found in vocabulary. Furthermore, for the second, both models suggest related concepts showing that the models are capable of capturing semantic relationships between words. On the other hand, for the last analogy word2vec suggests the word 'Mohammad_Rassool_cousin' which is not semantically meaningful, but Glove suggests 'bride' and 'wedding' which are suitable and semantically meaningful answers.

B. TRADITIONAL TEXT CLASSIFICATION

The second part of the assignment requires the implementation of 4 text classification approaches and the evaluation of their performance in terms of accuracy, dimensionality, and time cost.

1. Firstly, we import all the necessary libraries and the AG News Classification Dataset using pandas library. At this starting point, we convert the text in the 'Title' and 'Description' columns of the train and test DataFrames to lowercase letters. Next, we concatenate these two columns of the train and test into the new column called 'text'. This is done by adding the two columns together with a space in between. Then, we create 4 text classification pipelines, according to the pronunciation. The pipelines differ in the type of text representation used, specifically word 1-grams or char 3-grams, and the type of classifier used such as, Naïve Bayes or Support Vector Machines. Consequently, we train each of the four pipelines on the training data and evaluate their performance on the test data. For each pipeline, we record the time it takes to train the model, make prediction on the test set, calculate the accuracy of the predictions and get the dimensionality of the text representation. Lastly, we create and display a DataFrame with the results.

2. The next table contains the evaluation of accuracy, and the calculation of Dimensionality and Time cost of the previous four text classification models.

	NB (word 1-grams)	NB (char 3-grams)	SVM (word 1-grams)	SVM (char 3-grams)
Accuracy	0.902237	0.868684	0.919605	0.912105
Dimensionality	64999	31074	64999	31074
Time cost	4.826115	26.336993	10.888158	41.910980

Table 13: The Accuracy, Dimensionality and Time cost of the 4 requested text classification models

The above results, show that all four models are highly accurate, with SVM models outperforming the other. Furthermore, using char 3-grams can be more efficient in terms of memory usage since their dimensionality is lower than that of word 1-grams. In terms of time cost, Naïve Bayes model using word 1-grams is the fastest with the SVM using word 1-grams being the second.

3. At first, we identify the indices of the test samples that are misclassified by all the four pipelines. Then, we notice that the total number of misclassified texts is 341. In sequence, if the misclassified indices list is not empty, we print an example of a misclassified text, specifically the first one found. Alternatively, we print a message indicating that no texts were misclassified by all four models.

The example of a misclassified text by the models is the following: *"rivals try to turn tables on charles schwab by michael liedtke san francisco (ap) -- with its low prices and iconoclastic attitude, discount stock broker charles schwab corp. (sch) represented an annoying stone in wall street's wing-tipped shoes for decades..."*

Thenceforth, we count the number of the misclassified texts per category, and we list the results below.

Category	Count
World	112
Sports	9
Business	135
Sci/Tech	85

Table 14: The total number of misclassified texts per category

Lastly, to complete this task we find that the most common pair of correct category and wrong prediction, which is ('[Business, Sci/Tech]) and it was found 381 times.

Γ. TEXT CLASSIFICATION WITH RNNS

Firstly, we modify the given code to create some reusable functions for each subquestion. The first cell contains the modified data preprocessing, model training and model evaluation code. Furthermore, we move the functionality of data preprocessing under a function named *preprocess_data* that receives the *max_words* variable as a parameter so that we can later reuse the same code to produce data loaders and vocabulary based for different *max_words* value for task 3. Moreover, we modify the TrainModel function to calculate the average epoch time cost in seconds and return it. Thereafter, we alter the Model class to make it customizable. This allows to use this class to create all alternative models of the first task, by adjusting the *rnn_type*, *rnn_layers* and bidirectional parameters, and also load pre-trained embeddings and choose to keep them freezed or not (*pretrained_embeddings* and *freeze_embeddings* parameters for questions 4 and 5. In addition, we define the *run_experiments* method which trains and evaluates all models in an *experimental_configs* list and we use it for the first question. It also supports loading of pre-trained embeddings, freezed or not, thus we will also use it for questions 4 and 5 too. As a final point, we define the *print_eval_table* function that prints a table showcasing the results of all models, as described in the first task.

1. Initially, we create a list of the experimental configurations that describe the 6 requested models. Additionally, we create, train and evaluate these models by using the *run_experiments* method giving the *experimental_configs* as input.

The table below contains the evaluation of accuracy and the calculation of parameters and Time cost of the requested models.

	1RNN	1Bi-RNN	2Bi-RNN	1LSTM	1Bi-LSTM	2Bi-LSTM
Accuracy	0.871974	0.861053	0.864474	0.877105	0.884079	0.889868
Parameters	2.13628e+06	2.14716e+06	2.172e+06	2.16816e+06	2.21091e+06	2.31024e+06
Time cost	4.74409	4.9758	5.34347	5.14104	5.87746	6.86684

Table 15: The Accuracy, Parameters and Time cost of the 6 requested models

Although the differences in Accuracy between the models is small, the models with more parameters tend to perform better. 1RNN is an exception and performs better than the more complex RNNs which could be the result of a better random initialization of this model. Nonetheless, when comparing LSTMs with RNNs, all LSTMs perform better than RNNs, even if an RNN model has more parameters. e.g., 1LSTM performs better than 2Bi-RNN even though the latter has more parameters. Moreover, Time cost increases with the complexity (i.e., number of parameters) of the model. Models with 2 layers (vs. 1) are slower, with 2Bi-LSTM being the slowest model. Time cost was calculated on GPU runtimes and even larger differences were noticed on CPU runtimes. Last but not least, using 2 layers instead of 1, slightly improves the performance of both Bi-RNN and Bi-LSTM.

2. We notice that the total number of misclassified texts by all the models is 414. Additionally, the example of a misclassified text by all the models is: *"Greenspan: Debt, home prices not dangerous The record level of debt carried by American households and soaring home prices do not appear to represent serious threats to the US economy, Federal Reserve Chairman Alan Greenspan said Tuesday."*

Next, we count the number of the misclassified texts per category, and we list the results below.

Category	**Count**
World	122
Sports	23
Business	155
Sci/Tech	114

Table 16: The total number of misclassified texts per category

Finally, to complete this task we find that the most common pair of correct category and wrong prediction, which is ('[Business, Sci/Tech]) and it was found 1280 times.

3. Next, we create new datasets, data loaders and vocab by calling the preprocess_data with MAX_WORDS = 50. Consequently, we run again all the 6 models by using the *run_experiment* function, which will now respect the new modification.

	1RNN	1Bi-RNN	2Bi-RNN	1LSTM	1Bi-LSTM	2Bi-LSTM
Accuracy	0.349079	0.511842	0.469211	0.881579	0.887368	0.892368
Parameters	2.13628e+06	2.14716e+06	2.172e+06	2.16816e+06	2.21091e+06	2.31024e+06
Time cost	5.08226	5.36384	6.1992	5.80503	6.77106	7.1414

Table 17: The Accuracy, Parameters and Time cost of the requested models for task 3

RNN models perform significant worse when using MAX_WORDS=50 instead of 25. This is probably happening due to the vanishing gradient problem of RNNs. The longer the sequence that the RNN has to encode, the more difficult it is to capture information from the first

timesteps of the RNN. This problem is solved in LSTMs which manage to "remember" important information from earlier timesteps. Indeed, we notice that the LSTM models are not affected by vanishing gradient and perform similarly and slightly better than the experiments with MAX_WORDS=25. Regarding the complexity of the models, it remains the same regardless of the MAX_WORDS values, since its value only affect the number of timesteps/calculations and not the structure of the neural network.

4. Now, we use torchtext to load the GloVe embeddings. First, we create a vocabulary for this experiment, by finding the common words between the vocab that we constructed previously from the dataset, and the vocabulary of the GloVe embeddings. This way, we only load the embeddings that we need for training and evaluation of the model. Using these common words, we create a new torchtext vocab that also contains <PAD> and <UNK> tokens at the first two positions. We then create the embedding matrix by taking the embedding of each word in the new vocab and create a matrix whose rows are these embeddings in the same order as in the vocabulary. We set the first row to zeros (for the token) and the second vector to a random vector (for the token).

To load the embedding matrix, we modified the model so that the embedding layer is created with the same dimensions as the pre-trained embeddings matrix to load. We explicitly set *padding_idx=0* to prevent the <PAD> embedding from updating during training and we replace the randomly initialized embedding layer weight, with the pre-trained embeddings. Specifiacally,*self.embeding_layer=nn.Embedding(num_embeddings=pretrained_embeddigs.shape[0], embedding_dim=pretrained_embeddings.shape[1], padding_idx=0)* and *self.embedding_layer.weight.data.copy_(pretrained_embeddings)*. Now, we obtain the results below.

	1RNN	1Bi-RNN	2Bi-RNN	1LSTM	1Bi-LSTM	2Bi-LSTM
Accuracy	0.884605	0.861579	0.871711	0.907895	0.908158	0.907368
Parameters	2.05458e+06	2.06546e+06	2.0903e+06	2.08646e+06	2.12921e+06	2.22854e+06
Time cost	4.41377	4.78526	5.15109	5.07838	5.77404	6.71245

Table 18: The Accuracy, Parameters and Time cost of the requested models for task 4

We observe that all models perform better when using pre-trained embeddings vs random embedding initialization. The best model with pre-trained embeddings is 1Bi-LSTM with accuracy 0.908158. The best model with random embeddings initialization (task 1) was 2Bi-LSTM with accuracy 0.889868. That is, the usage of pre-trained embeddings lead to significant improvements. The complexity of the models is similar to that of the models in first task, since we just changed the way we initialize the weights of the embeddings, which are still parameters to be optimized during training. The number of parameters in the models with pre-trained embeddings is slightly smaller that in the models of task 1 only due to the slightly smaller vocabulary size, since not all words of our original vocabulary has a corresponding GloVe embedding.

5. In this task, the pre-trained embeddings should be modified during model training (freeze embeddings). To achieve this, we modify the code so that gradient calculation is not required for the embedding layer, as follows *self.embedding_layer.weight.requires_grad = not freeze_embeddings*. (i.e. when we want to freeze the embeddings this translates to *self.embedding_layer.weight.requires_grad = False)*. During the creation of the optimizer, this layer will be ignored because its *requires_grad* variable is False, and the layer will not be

optimized during training, *optimizer = torch.optim.Adam([param for param in classifier.parameters() if param.requires_grad == True], lr = LEARNING_RATE).*

The following table contains the evaluation of accuracy, and the calculation of parameters and Time cost of the requested models, respectively.

	1RNN	1Bi-RNN	2Bi-RNN	1LSTM	1Bi-LSTM	2Bi-LSTM
Accuracy	0.854079	0.867763	0.861842	0.899342	0.896053	0.899868
Parameters	10884	21764	46596	42756	85508	184836
Time cost	4.47029	4.75267	4.88219	4.81324	5.08863	5.50467

Table 19: The Accuracy, Parameters and Time cost of the requested models for task 5

With freezed pre-trained embeddings, the models perform slightly worse than with non freezed pre-trained embeddings. Despite this, we notice that the complexity of the models is much smaller, since the parameters of the models are now only the RNN/LSTM parameters and the linear layer parameters. In the previous models, the majority of the parameters originated from the embedding layer, which is now freezed during training.

6. To complete the third and last part of this assignment, we modify the preprocessing process to respect the new dataset format, which contains a *"review"* column for the text to classify, and a *"sentiment"* column for the label (*"positive"* or *"negative"*). Furthermore, we modify the model into a binary classification model, which has an output dimension of 1 and uses a sigmoid activation function instead of softmax after the linear layer. The model will predict a number between 0 and 1 and the predictions closer to 0 will be interpreted as *"negative"* and closer to 1 as *"positive"*. Moreover, we change the loss to Binary Cross Entropy loss, *BCELoss* to comply with the new model output. The table below contains all the requested results.

	1RNN	1Bi-RNN	2Bi-RNN	1LSTM	1Bi-LSTM	2Bi-LSTM
Accuracy	0.7137	0.7155	0.7102	0.7131	0.717	0.7207
Parameters	2.91719e+06	2.92788e+06	2.95271e+06	2.94906e+06	2.99162e+06	3.09095e+06
Time cost	6.5928	6.66773	6.76621	6.64112	6.92877	7.40679

Table 19: The Accuracy, Parameters and Time cost of the requested models for the last task

In this new dataset, again more complex models tend to perform slightly better. An exception is the 2Bi-RNN model which performs worse than less complex RNNs. The most complex model, 2Bi-LSTM, achieves the best accuracy. Regarding Time cost, the more complex a model, the slower is its average epoch runtime, although the differences in GPU execution is small. The slowest model is again the most complex model, 2Bi-LSTM. Lastly, using 2 layers instead of 1, results in lower accuracy in the Bi-RNN models, but improves accuracy of the Bi-LSTM models and leads to the best model 2Bi-LSTM.