Aspect Based Sentiment Analysis

Group Id: 39

23690372
Henh Phan Chenh

23552815
Ri Nandiya

24026638
Tien Trung Pham

***Abstract*—Sentiment analysis, a critical component of Natural Language Processing (NLP), is designed to determine the overall sentiment expressed in a given text. However, when aspect factors are considered, users can gain more refined and accurate insights from the text. This paper delves into aspect-based sentiment analysis, utilizing a Restaurant dataset that comprises 8,879 reviews of a single restaurant. These reviews are categorized into eight distinct aspects: *food*, *service*, *staff*, *price*, *ambience*, *menu*, *place*, and *miscellaneous*. To predict sentiment polarities (i.e., *positive*, *negative*, and *neutral*), we implemented and compared three different models. Our comprehensive analysis involved examining five different word embeddings and three seq2seq models. The findings highlighted the significant role of attention mechanisms in achieving higher accuracy within the restaurant dataset. Our top-performing model, an Attention-based Long Short-Term Memory Network (AT-LSTM), achieved an accuracy of 62.9% in predicting the sentiment polarities. Furthermore, the study investigated the effects of using two attention mechanisms and varying the placement of aspect knowledge. It was observed that separating reviews and aspects into different locations did not confer any substantial advantage in terms of accuracy. On the contrary, concatenating them proved to be more time-efficient and enhanced the overall accuracy. Similarly, adding an additional attention layer did not result in any noticeable improvement in performance. Thus, our research underscores the efficacy of a single attention mechanism and a unified approach to aspect placement for optimal sentiment analysis results.**

I. Introduction

Sentiment analysis, also referred to as opinion mining, has garnered significant attention in recent years as a fundamental task in natural language processing (NLP). Aspect-level sentiment analysis, a more granular task, offers comprehensive and detailed insights. In this study, we focus on aspect-level sentiment classification and observe that the sentiment polarity of a sentence is greatly influenced by both its content and aspect. This task involves determining the sentiment polarity (positive, negative, neutral) of a specific aspect within a given sentence. For instance, in the sentence ”great food but the service was dreadful!”, the sentiment polarity of the aspect ”food” is positive, while the polarity of the aspect ”service” is negative. Researchers typically approach this task by employing machine learning algorithms to build sentiment classifiers in a supervised manner. Common methods in the literature include feature-based Support Vector Machine (SVM)([1], [2]) and neural network models([3], [4]). Neural models are gaining popularity due to their ability to learn text representations directly from data without requiring manual feature engineering. They also excel in capturing semantic relationships between aspect and context words in a more scalable manner compared to feature-based SVM.

However, Neural network models are still in their early stages regarding aspect-level sentiment classification. While some approaches, such as Target-Dependent LSTM (TD-LSTM) and Target-Connection LSTM (TC-LSTM), have shown promise by incorporating target information, they primarily focus on the target and do not adequately consider aspect information, which is proven to be crucial for aspect-level classification. Another interesting study came when [6] came up with an attention mechanism. Attention has become an effective mechanism to obtain superior results, as demonstrated in the image recognition [7], machine translation [8], and sentence summarization [9]. Example paper that shows the State-of-Art on aspect-based sentiment analysis achieved are [10], [11], and [12]

In this paper, we aim to compare three different models with different aspects of location and different integration methods. Our first model focused solely on inserting aspect information on a hidden layer of the seq2seq model without any attention mechanism. Meanwhile, the second model used the full input, which consisted of the concatenation between reviews and aspects, and deployed it into the seq2seq model with an additional attention mechanism. Finally, our last model comprised of the mix between the first and the second models. Starting from putting attention mechanism only from reviews input, another attention was added after the concatenation between the aspects input and the result vector from the first attention. Based on our findings, our contributions to the field of aspect-based sentiment analysis are as follows:

- We showed that LSTM outperforms the other seq2seq model and is more suitable for aspect-based sentiment analysis.
- Our research showed that implementing attention mechanisms plays a significant role in filtering unrelated words.
- We suggested that concatenating aspect and review embeddings from the outset enhances the performance of the model, leading to better sentiment prediction.

II. Methods

In this section, we described the three different architectures that we used for aspect-based level sentiment analysis. First, the definition of the task will be discussed before describing all the approaches.

A. Task Definition

Given a sentence $s = w_1, w_2, w_3, w_4, ...w_n$ with n length words and aspect word w_i occurring in sentence s, each architecture aims to determine the sentiment polarity of sentence s towards aspect w_i. For instance, consider the sentiment expressed in the sentence "great food, but the service was dreadful!" toward two aspects: "food" evokes a positive sentiment, whereas "service" elicits a negative one. In text analysis, we convert each word into a compact, continuous, and real-valued vector known as word embedding [13]. All the word vectors are stacked in a word embedding matrix $L \in \mathbb{R}^{d \times |V|}$, where d is the dimension of the word vector and $|V|$ is the vocabulary size. The word embedding of w_i is notated as $e_i \in \mathbb{R}^{d \times 1}$, which is a column in the embedding matrix L.

B. Preprocessing

Given a restaurant dataset split into train, validation and test, we combined both the train and validation datasets to predict the test dataset. As the aim of this paper is to find out whether putting aspect information and adding additional information plays a role in increasing the accuracy or not, we decided that the same parameter should be used for comparison.

Moreover, since we would compare five different types of word embedding for ablation study, inspired by [14], rather than using the randomly initialized weight for an unknown word, we decided to use the average of each sentence to represent the connection between unknown words. However, this approach was only used for Glove embedding input [18]. Furthermore, another lemmatization was also adopted to increase the chance of each word existing in Glove-trained embedding. As for other input embeddings (Word2Vec and FastText [19]), there were no additional approaches implemented as all the unknown words would be trained by using CBOW and Skipgram methods.

C. Architectures

First Model. In the first model, we introduced an architecture that implemented the aspect information in different locations. As we aim to examine how the placement of aspect information may impact the model, in this approach, only the sentences (s) served as the input. Meanwhile, the aspect information (w_i) was combined with the final hidden vector of seq2seq model h_n, representing the entire sentence. This combined vector (w_i and h_n) is then passed through a linear layer, resulting in a vector with a length corresponding to the number of class labels, followed by a *Softmax* layer. The illustration of this architecture can be seen in **Figure 1**.

Second Model. As we believed that the seq2seq model could not uncover the crucial part for aspect-based sentiment analysis, we suggested designing an attention mechanism that has the ability to identify the key part of a sentence corresponding to a given aspect. The second model employed a seq2seq model to encode the full concatenation of reviews and aspects as the input sequence. The attention mechanism computed the significance (attention weights) of each hidden layer of the model by referencing the final hidden layer. These

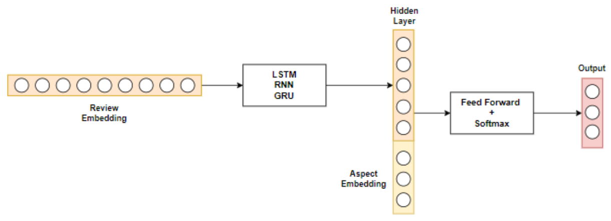

Fig. 1. Architecture of Model 1

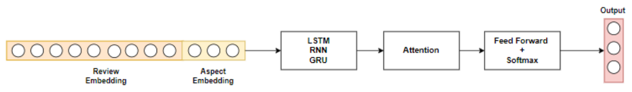

Fig. 2. Architecture of Model 2

attention weights were then used to calculate a weighted sum of the hidden layers, resulting in a context vector. This context vector, representing the input sequence with the given aspect, was then passed into a fully connected layer and then through a *Softmax* layer to produce the final output. The illustration of this model can be seen in **Figure 2**.

Third Model. The last model was the combination of the first model and the second model. However, instead of applying one attention, two attentions mechanism was implemented. Starting from the input, which was similar to the first model, an attention mechanism was put to the hidden layer of the seq2seq model from review input. After the concatenation of aspect information, we fed the concatenation vector into another attention. The idea of this approach was that we would like to know whether the attention mechanism can just simply focus on the review input and ignore the aspect knowledge in case there was no word in the text reviews that represents the aspect knowledge. For instance, aspect *miscellaneous* doesn't focus on any specific point of the reviews. In the review "Hostess was extremely accommodating when we arrived an hour early for our reservation", the polarity for *miscellaneous* is *neutral*. Meanwhile, the review "I like the smaller portion size for dinner" has polarity *negative*. From those two examples, we can see that there are times when there is less semantic relation between the sentences in the reviews and the aspects. Therefore, we put the attention mechanism first for the reviews to capture the most important words that affect the polarities and additional attention after concatenation with the aspect to filter the necessary aspect information. The figure of the third model can be seen in **Figure 3**.

We employed three primary seq2seq models and one Natural Language Processing technique in this research: Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU). Additionally, we integrated

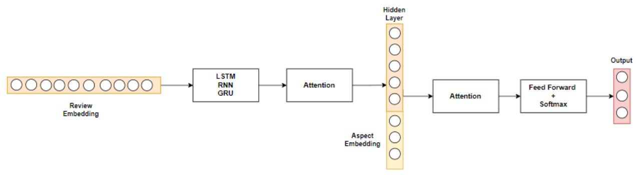

Fig. 3. Architecture of Model 3

the attention technique into the implementation of the second and third models.

- **RNN**. Recurrent Neural Networks is a neural network architecture designed primarily for identifying patterns in sequential data [15]. This data can include handwriting, genomes, text, or numerical time series, which are frequently generated in industrial contexts. Recurrent Neural Networks are distinct from Feedforward Neural Networks, also called Multi-Layer Perceptrons (MLPs), in their information processing. Unlike Feedforward Networks, which transmit data in a single direction without loops, RNNs incorporate cycles that loop information back into the network. This allows RNNs to account for previous inputs $(X_0 : t-1)$ in addition to the current input (X_t), enhancing their functionality beyond that of Feedforward Networks. The formula of RNN can be seen as follows:

$$H_t = \phi_h(X_t W_{xh} + H_{t-1} W_{hh} + b_h) \tag{1}$$

$$O_t = \phi_o(H_t W_{ho} + b_o) \tag{2}$$

  where hidden state and the input at time step t is represented by $H_t \in \mathbb{R}^{n \times h}$ and $X_t \in \mathbb{R}^{n \times d}$ where n is the number of samples, d is the number of inputs of each sample, and h is the number of hidden units. $W_{xh} \in \mathbb{R}^{d \times h}$ is a weight matrix, while hidden-state-to-hidden-state matrix is denoted by $W_{hh} \in \mathbb{R}^{h \times h}$. Lastly, all this information is passed through an activation function ϕ, which is usually a logistic sigmoid or tanh function, to prepare the gradients for usage in backpropagation.
- **LSTM**. The Recurrent Neural Network (RNN) extends the capabilities of the standard feed-forward neural network. Yet, conventional RNNs encounter obstacles like the vanishing or exploding gradient problem. To combat these limitations, the Long Short-Term Memory network (LSTM) was devised, demonstrating notable advancements [16]. Within the LSTM structure, there exist three gates which are *forget gate*, *Input gate*, *Output gate* alongside a cell memory state. The Formula of each cell in LSTM can be computed as follows:

$$X = \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} \tag{3}$$

$$f_t = \sigma(W_f \cdot X + b_f) \tag{4}$$

$$i_t = \sigma(W_i \cdot X + b_i) \tag{5}$$

$$o_t = \sigma(W_o \cdot X + b_o) \tag{6}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c \cdot X + b_c) \tag{7}$$

$$h_t = o_t \odot \tanh(c_t) \tag{8}$$

  where W_i, W_f, $W_o \in \mathbb{R}^{d \times 2d}$ are the weighted matrices and b_i, b_f, $b_o \in \mathbb{R}^d$ are biases of LSTM to be learned during training, parameterizing the transformations of the input, forget, and output gates respectively. σ is the *Sigmoid* function and $\odot$ stands for element-wise multiplication.
- **GRU**. The aim of GRU is to enable each recurrent unit to capture dependencies across various time scales dynamically [17]. Like the LSTM unit, the GRU incorporates gating units that regulate information flow within the unit, albeit without individual memory cells. At time step t, the activation h_j^t of the GRU is computed as a linear combination of the previous activation h_j^{t-1} and the candidate activation $\tilde{h}_j^t$.interpolation between the previous activation h_j^{t-1} which can be formulated as bellow:

$$h_j^t = (1 - z_j^t) h_j^{t-1} + z_j^t \tilde{h}_j^t, \quad (5) \tag{9}$$

  The update gate z_j^t determines the extent to which the unit adjusts its activation or content. It is calculated as follows:3.5

$$z_j^t = \sigma(W_z x_t + U_z h_{t-1})_j. \tag{10}$$

  This procedure of taking a linear sum between the existing state and the newly computed state is similar to the LSTM unit. The GRU, however, does not have any mechanism to control the degree to which its state is exposed; it exposes the whole state each time. The candidate activation $\tilde{h}_j^t$ is then computed similarly to that of the traditional recurrent unit **Equation 1**. Finally, the rest gates can be formulated as follows:

$$r_j^t = \sigma(W_r x_t + U_r h_{t-1})_j. \tag{11}$$

- **Attention**. The attention mechanism allows the model to focus on different parts of the input sequence [6]. It computes a context vector as a weighted sum of encoder hidden states, where the weights are derived from the alignment scores between the encoder hidden states and the current decoder state.
  The alignment score is calculated as follows:

$$e_{t,i} = \text{score}(s_t, h_i) \tag{12}$$

  The attention weights are obtained using a softmax function:

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{k=1}^{T} \exp(e_{t,k})} \tag{13}$$

  The context vector is then computed as:

$$c_t = \sum_{i=1}^{T} \alpha_{t,i} h_i \tag{14}$$

By integrating the attention mechanism, the RNN, LSTM and GRU models can dynamically focus on relevant parts of the input sequence, improving their ability to handle long-range dependencies.

III. Experiments

A. Dataset Description

We conduct experiments using the Restaurant dataset, which comprises customer reviews. Each review includes a set of aspects along with their respective polarities. Our objective is to determine the polarity of an aspect within a sentence. The dataset statistics are outlined in **Table 1**.

Aspect	Positive	Neutral	Negative
	Train/Val/Test	Train/Val/Test	Train/Val/Test
Fo.	754/88/96	1298/180/158	255/22/37
Pl.	125/25/12	430/42/47	139/21/22
St.	332/34/46	129/12/10	922/119/113
Mc.	227/33/26	531/73/88	196/23/22
Se.	174/25/30	128/20/14	329/39/34
Pr.	72/8/10	136/17/16	114/20/12
Me.	64/7/12	372/41/60	39/3/4
Am.	181/21/13	53/3/19	90/12/19
Total	**1702/241/245**	**3077/388/412**	**2084/259/263**

TABLE I
Aspect distribution in the dataset

B. Experiment Setup

We utilize all three models for polarity classification in our experiments. We initialize word vectors using Glove [18], Word2Vec, and FastText [19]. Training the word embeddings involves both Skipgrams and Cbow methods for Word2Vec and FastText. For Glove, we employ a pre-trained word embedding from Wikipedia 2014 with a vector dimension of 50, matching the lengths of vectors for Word2Vec and FastText. Each input sentence is represented as a $40x50$ matrix, where the maximum sentence length is 40 words, with 50 vectors representing each word. In the first and third models, we focused on the reviews for input and combined the aspect knowledge in the hidden layer. In contrast, the second method utilizes the concatenation of the aspect information and reviews as the whole input from the beginning. The length of attention weights matches the input length for both model 2 and model 3. We trained all models for 1000 epochs with learning rates set at 0.001. Initially, we compare the best accuracy among all word embeddings by using a base sequence model, LSTM, followed by comparing the best word embeddings with different sequence models. The detail of our experimental setup can be seen in **Table II**.

IV. Results

As we tried different word embeddings as the inputs, Word2Vec with the Skipgram method showed better accuracy than the others in the first and second models. However, FastText with the Skipgram method achieved the highest accuracy in the third model. The accuracy comparison for different word embeddings can be seen in **Table III**, **Table IV** and **Table V**.

Params	Model 1	Model 2	Model 3
Hidd.Layer1	50	50	50
Hidd.Layer2	-	-	100
Optimizer	Adam		
Epoch	1000		
Loss.func	Cross Entropy		

TABLE II
Model Parameters

Input Embeddings	**Accuracy**
LSTM, Glove	53.16
LSTM, W2V.Cb	43.61
LSTM, W2V.Sk	**57.71**
LSTM, Ft.Cb	52.60
LSTM, Ft.Sk 1	55.49

TABLE III
Accuracy Results on Different Embeddings, First Method

We then furthered the research by comparing each model with its respected best word embedding inputs with other seq2seq models. The accuracy comparison for different seq2seq models on each model can be seen in **Table VI**, **Table VII** and **Table VIII**.

The result from comparing all models' performance showed that the second model with LSTM and Word2Vec by Skipgram method embedding outperformed the other models. The training and loss accuracy of this model can be seen in **Figure 4**.

Our most effective model allocated greater attention weight to specific words that represent aspects of knowledge. For instance, in **Figure 5**, when presented with a review sentence and the *Staff* aspect, the model concentrated more on the words "*Server*", "*Helpful*", and "*Suggested*", while giving less attention to words like "*Food*", "*Little*", and "*Difficulty*".

Another example of our best model's performance is illustrated in **Figure 6**, where it allocated attention weights evenly across a review with the *Service* aspect. The model's attention mechanism identified the most significant words representing negative sentiment related to this aspect. Specifically, it assigned more weight to the words "*Wait*" and "*Hour*" compared

Input Embeddings	**Accuracy**
AT-LSTM, Glove	62.60
AT-LSTM, W2V.Cb	60.60
AT-LSTM, W2V.Sk	**63.60**
AT-LSTM, Ft.Cb	60.16
AT-LSTM, Ft.Sk 1	62.38

TABLE IV
Accuracy Results on Different Embeddings, Second Method

Input Embeddings	**Accuracy**
2AT-LSTM, Glove	61.49
2AT-LSTM, W2V.Cb	55.05
2AT-LSTM, W2V.Sk	61.60
2AT-LSTM, Ft.Cb	55.05
2AT-LSTM, Ft.Sk	**62.82**

TABLE V
Accuracy Results on Different Embeddings, Third Method

Input Embeddings	Accuracy
LSTM, W2V.Sk	57.71
RNN, W2V.Sk	51.50
GRU, W2V.Sk	53.50

TABLE VI
ACCURACY RESULTS ON DIFFERENT SEQ2SEQ MODELS, FIRST METHOD

Input Embeddings	Accuracy
AT-LSTM, W2V.Sk	**63.82**
AT-RNN, W2V.Sk	62.60
AT-GRU, W2V.Sk	62.38

TABLE VII
ACCURACY RESULTS ON DIFFERENT SEQ2SEQ MODELS, SECOND METHOD

to "*Tried*" and "*Get*".

V. CONCLUSION

We developed three distinct architectures to predict review polarity using aspect knowledge. Experimental results indicated that attention mechanisms significantly enhanced performance. A single attention mechanism was sufficient to capture keywords related to certain aspects of the reviews. Moreover, separating reviews and aspects into different locations did not provide any significant advantage. Instead, concatenating them improved accuracy to 62.60% and was more time-efficient for the model to be computed. Additionally, among the seq2seq models, the LSTM model outperformed others. Furthermore, the Skipgram method was found to be the most effective for word embedding input.

VI. TEAM CONTRIBUTION

The contribution of each member :

- **23690372**: Second Model Code, Third Model Code, Visualization.(45%)
- **23552815**: First Model Code, Preprocessing Code, Reports.(45%)
- **24026638**: Visualization.(10%)

Input Embeddings	Accuracy
2AT-LSTM, Ft.Sk	**62.82**
2AT-RNN, Ft.Sk	60.71
2AT-GRU, Ft.Sk	60.71

TABLE VIII
ACCURACY RESULTS ON DIFFERENT SEQ2SEQ MODELS, SECOND METHOD

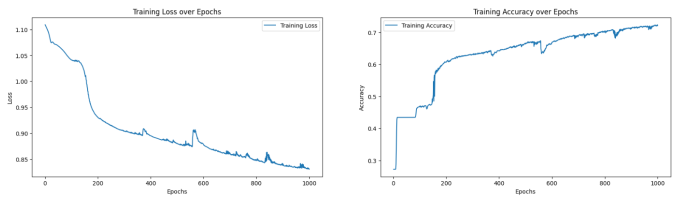

Fig. 4. Training Loss and Accuracy, AT-LSTM Word2Vec Skipgram

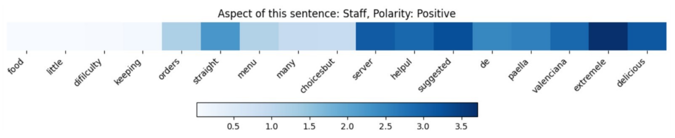

Fig. 5. Attention Visualization with Aspect Staff

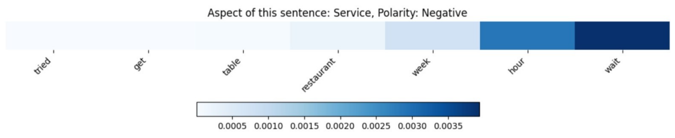

Fig. 6. Attention Visualization with Aspect Service

REFERENCES

[1] S. Kiritchenko, X. Zhu, C. Cherry, and S. Mohammad. " Nrccanada-2014: Detecting aspects and sentiment in customer reviews," In Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval), 2014, pages 437–442.

[2] J. Wagner, P. Arora, S. Cortes, U. Barman, D. Bogdanova, J. Foster, and Lamia Tounsi. "Dcu: Aspectbased polarity classification for semeval task 4," In Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval), 2014, pages 223–229.

[3] L. Dong, F. Wei, C. Tan, D. Tang, M. Zhou, and K. Xu. "Adaptive recursive neural network for target-dependent twitter sentiment classification," In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, 2014, pp.49–54.

[4] H. Lakkaraju, R. Socher, and C. Manning. "Aspect specific sentiment analysis using hierarchical deep learning," In NIPS Workshop on Deep Learning and Representation Learning. 2014.

[5] D. Tang, B. Qin, X. Feng, and T. Liu. "Target-dependent sentiment classification with long short term memory," arXiv preprint arXiv:1512.01100. 2015.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. "Attention is all you need," in NIPS, 2017.

[7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. "An image is worth 16x16 words: Transformers for image recognition at scale," arXiv preprint arXiv:2010.11929, 2020.

[8] D. Bahdanau, K. Cho, and Y. Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473. 2014.

[9] A. M. Rush, S. Chopra, and J. Weston. "A neural attention model for abstractive sentence summarization," arXiv preprint arXiv:1509.00685. 2015.

[10] F. Fan, Y. Feng, and D. Zhao. "Multi-grained attention network for aspect-level sentiment classification," In Proceedings of the 2018 conference on empirical methods in natural language processing, pp.3433–3442. 2018.

[11] D. Bahdanau, K. Cho, and Y. Bengio. "Neural machine translation by jointly learning to align and translate," CoRR, abs/1409.0473, 2014.

[12] Y. Kim, C. Denton, L. Hoang, and Alexander M. Rush. "Structured attention networks," In International Conference on Learning Representations. 2017.

[13] T. Mikolov, K. Chen, G. S. Corrado, J. Dean. "Efficient Estimation of Word Representations in Vector Space," in Proceedings of Workshop at ICLR, 2013.

[14] J. Coates, D. Bollegala. "Frustratingly Easy Meta-Embedding – Computing Meta-Embeddings by Averaging Source Word Embeddings," in Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol 2, pp.194–198. 2018.

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning internal representations by error propagation. Technical report," California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[16] S. Hochreiter and J. U. Schmidhuber. "Long short-term memory," Neural computation, 9(8):1735–1780, 1997.

[17] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259, 2014.

[18] J. Pennington, R. Socher, and C. D. Manning. ”Glove: Global vectors for word representation,” in Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP), 2014, 12:1532–1543.

[19] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. ”Enriching Word Vectors with Subword Information,” Transactions of the Association for Computational Linguistics, Vol 5, pp.135–146. 2017.