

Deep Learning Approaches to Aspect Based Sentiment Analysis

Project report submitted in partial fulfillment of the requirement for

POST-GRADUATE DIPLOMA IN
STATISTICAL METHODS AND ANALYTICS



submitted by

Abhishek Bhatia and Dristanta Nirola

DST-19/20-003 and DST-19/20-010

July, 2020

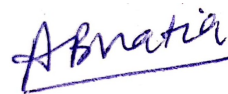
Certificate

This is to certify that Abhishek Bhatia and Dristanta Nirola have done the project under my supervision (from 03/02/2020 to 27/07/2020). This is an original project report based on work carried out by them in partial fulfillment of the requirement for the Post-Graduate Diploma in Statistical Methods and Analytics programme of the Indian Statistical Institute, North-East Centre, Tezpur, Assam.

Dr. Malay Bhattacharyya

Acknowledgements

We would like to express profound gratitude to our guide Dr. Malay Bhattacharyya for all his moral support, suggestions and constructive criticism which has contributed immensely to the evolution of new ideas on the project. Moreover, we are grateful for the cooperation and constant support from Dr. Sanjit Maitra. His continuous encouragement has helped us complete the project on time. This would not have been possible without our friends, and family, the simple joy of whose company lifted our spirits in such a tough time of global crisis.



Abhishek Bhatia and Dristanta Nirola

TABLE OF CONTENTS

Certificate	i
Acknowledgements	
1 Introduction	1
2 Methodology	4
2.1 Problem	4
2.2 Description of Data	4
2.3 Data analysis and techniques	5
2.3.1 Deep learning and neural networks	5
2.3.2 Convolution Neural Networks	6
2.3.3 Recurrent Neural Networks	6
2.3.4 Long Short Term Memory networks	7
2.3.5 Word embeddings	8
2.3.6 Loss function optimization	8
2.4 Aspect Extraction (Subtask1)	9
2.4.1 Problem statement	9
2.4.2 System Description	10
2.4.3 Preprocessing	10
2.4.4 Preparing the input sequences	11
2.4.5 Preparing word embeddings	11
2.4.6 Building the architecture	11
2.5 Aspect Polarity Detection (Subtask 2)	12
2.5.1 System Description	12
2.5.2 Problem Statement	13
2.5.3 Preprocessing	13
2.5.4 Preparing the input sequences	14
2.5.5 Preparing word embeddings	14
2.5.6 Building the architecture	14

3	Results and Discussions	16
3.1	Performance measures	16
3.2	Aspect Extraction (Subtask 1)	17
3.3	Aspect Polarity Detection (Subtask 2)	18
4	Conclusions	20
4.1	Summary of Conclusions	20
4.2	Future scope	21

Chapter 1

Introduction

The recent boom of blogs, review sites, and especially social networking sites has led to the creation of a large pool of information. This text-based user-generated opinionated content is an invaluable source of information for businesses, enabling them to generate actionable insights into the preferences of customers. Automatically analyzing the large volume of data generated from customer feedback would help businesses tailor products and services to various market segments. The research interest in sentiment analysis has taken off tremendously over the last two decades. This field presents a plethora of challenges to the researchers in the Natural Language Processing (NLP) community [1]. The mostly addressed tasks include spam detection, part-of-speech (POS) tagging and named entity recognition (NER), while the still harder tasks include Question-Answering (QA), paraphrasing, text summarization and developing chat dialogue systems. The areas in which good progress is being made are sentiment analysis, coreference resolution, word sense disambiguation (WSD), parsing, machine translation (MT) and information extraction (IE).

Sentiment analysis or opinion mining is a text analysis method to detect polarity (e.g. a positive or negative opinion) within the text. Sentiment analysis has been called a “suitcase research problem” [2], as it contains a suitcase of other NLP challenges (atleast 15) which need to be addressed separately as subtasks of the problem. Although in most research, it is primarily treated as a simple classification task which can’t effectively tackle the basic premise of sentiment analysis i.e. the idea of conveying emotions and opinions through natural language. The text can be analyzed at various levels i.e. document level to examine the overall polarity of the document and the sentence level in which the document is broken down into individual sentences. Much of the current approaches limit their focus to these two levels i.e. detecting the overall polarity of the text [3, 4, 5].

However, in unstructured product reviews, people tend to express multiple opinions about various features of the product. Depending upon their likes and dislikes some of the opinions expressed about specific features may be positive and some may be negative. Here the text is examined at the third level i.e. the aspect level where the polarity of the specific domain-related entities (e.g., laptops, restaurants) and their aspects/attributes (e.g., battery, screen;

food, service) are analyzed. Such an approach, is formally called **Aspect Based Sentiment Analysis (ABSA)**, wherein a document is assumed to be composed of a range of opinion units targeted towards multiple entities in the text [6]. Entity here could be tangible referring to a product, a person, an organization, etc. or could be intangible referring to a topic, say service in a restaurant. These entities could be further made up of components which in itself have a set of associated attributes.

The components and attributes when considered as a whole are referred to as aspects of the entity. Aspect terms can be a word or a phrase i.e. single word or multi-word aspects. Further, aspects can be categorized into *implicit* and *explicit* [7]. Explicit aspect terms are words or phrases that explicitly convey the target entity in the sentence. Consider the following review text:

The food was delicious but the service was horrible

Here, the customer is expressing a positive sentiment about “food” and a negative sentiment about “service”. An aspect based sentiment classifier would first extract the relevant aspect terms i.e. food, service and classify the sentiment expressed towards each of these aspects i.e. positive, negative. Simply labelling the overall polarity of the sentence (positive or negative) ignores much of the valuable information about the various targets present in the text. It is also reasonable to assume that the longer the text gets, more likely is the possibility of finding a range of opinions expressed towards multiple target entities. Note here, the words “food” and “service” are explicitly mentioned in the sentence. Whereas in the following review text a positive opinion is expressed about food quality and price through an implicit aspect clue (IAC) [8]. In this project, we only deal with explicit aspects.

The cheeseburger is sooo tasty and extremely cheap as well!!

The first directed efforts to analyze the sentiment of text at the aspect level were made by Hu et al.[9] wherein they performed a subjectivity analysis on a sub-sentence level. Since then, this field has matured significantly within sentiment analysis and has four major sub-tasks (ST); **ST1**: aspect term extraction, **ST2**: aspect term polarity, **ST3**: aspect category detection, and **ST4**: aspect category polarity [10]. Here it is important to point out that the terms *sentiment analysis* and *opinion mining*, although used interchangeably have different origination. The former has its roots in tracking market sentiment [11] while the latter primarily has linkages to the web domain, more specifically product reviews [12]. And because aspect based sentiment analysis is primarily used in the product reviews domain, opinion mining might be a better suited name instead.

There is another importance difference between the coarse grained analysis and a finer aspect level analysis. This relates to the relatively higher degree of inter-annotator disagreement when annotating texts at the aspect level rather than the document or the sentence level [13]. This poses problems in establishing the bounds of human level performance which

we can expect from a machine on an aspect based sentiment task. Besides this makes the task of developing a large annotated dataset suited for ABSA a tad bit harder due to a high possibility of diverging human annotator opinions.

Initially, most of the approaches to ABSA relied on hand-crafted rules based on linguistic techniques such as dependency parsing and POS tagging [9, 14, 15, 16]. More recently, deep learning approaches are being used more and more [17, 18, 19, 20]. This forms the basic premise of our project, where we focus on the tasks of extracting the aspect terms (**ST1**) and then detecting the polarity associated with the term (**ST2**). We achieve this through adapting deep learning architectures based on Convolutional Neural Networks (Section 2.3.4) and Long Short Term Memory (Section 2.3.4) networks from the literature on our task. We briefly explain the underlying theory of the networks and focus on the relevance of these architectures to ABSA.

Chapter 2

Methodology

2.1 Problem

Aspect Based Sentiment Analysis (ABSA) is a branch of sentiment analysis, comprising mainly four broad subtasks (ST); ST1: aspect term extraction, ST2: aspect term polarity, ST3: aspect category detection, and ST4: aspect category polarity. In this report, we focus on the 1st subtask (ST1) (Section 2.4.1) of extracting the aspect terms and the 2nd subtask (ST2) (Section 2.4.1) of determining the polarity expressed towards each aspect term in a review text.

2.2 Description of Data

We use the SemEval 2014 dataset [10] consisting of user reviews on the restaurant domain. The dataset contains over 3000 English sentences each along with annotations for aspect terms (Subtask 1), aspect term polarity (Subtask 2), aspect categories (Subtask 3) and aspect-category specific polarity (Subtask 4). The dataset is adapted from [21], which only provided annotations for the aspect category. The sentences in the datasets are annotated using XML tags (Figure 2.1).

```
<sentence id="1458">
<text>Our agreed favorite is the orrechiote with sausage and chicken (usually the waiters are kind enough to split the dish in half so you get to sample both meats).</text>
<aspectTerms>
  <aspectTerm term="orrechiote with sausage and chicken" polarity="positive" from="27" to="62"/>
  <aspectTerm term="waiters" polarity="positive" from="76" to="83"/>
  <aspectTerm term="meats" polarity="neutral" from="152" to="157"/>
  <aspectTerm term="dish" polarity="neutral" from="113" to="117"/>
</aspectTerms>
<aspectCategories>
  <aspectCategory category="food" polarity="positive"/>
  <aspectCategory category="service" polarity="positive"/>
</aspectCategories>
</sentence>
```

Figure 2.1: An example of XML annotated review text for the restaurant domain

Key information about the dataset

Table 2.1 shows the key information extracted from a preliminary exploration of the restaurant dataset.

i) Training Dataset

Total no. of Reviews	3699
No. of unique Aspects	1295
Top 5 Aspect Terms	['Food', 'Service', 'Place', 'Price', 'Menu']
No. of unique reviews	2021
Minimum Review length	1
Maximum Review length	69
Average Review length	15.1
Review Polarity (Positive, Neutral, Negative)	(2164, 637 ,807)

ii) Test Dataset

Total no. of Reviews	1134
No. of unique Aspects	555
Top 5 Aspect Terms	['Food', 'Service', 'Menu', 'Price', 'staff']
No. of unique reviews	606
Minimum Review length	4
Maximum Review length	55
Average Review length	14
Review Polarity (Positive, Neutral, Negative)	(728, 196, 196)

Table 2.1: Details of the restaurant dataset

2.3 Data analysis and techniques

Here we provide the necessary background on the deep learning techniques used in the project. This will serve as a primer for the methodology adopted for the two subtasks namely, aspect extraction (ST1) and aspect polarity detection (ST2).

2.3.1 Deep learning and neural networks

The concept of artificial neurons is not a new idea and its origin could be traced back to the 1940s [22]. Neural networks are composed of individual computation units called neurons, which in a forward pass, take as input the computations from previous unit(s), or a layer and subsequently perform mathematical operations to produce an output. This output is called the activation of the neuron and may serve as input for another neuron or a layer of neurons. This mathematical operation is usually a weighted linear combination plus a bias term with subsequent non-linearity applied to it. Most common non-linear activation functions are the ReLu, softmax and tanh. A stacking of multiple layers results in a deep neural network. The gradient based learning is carried out through backpropagation algorithms [23], which iteratively update the weights of the network in a backward pass with a aim to minimize the loss function. This process of forward and backward passes is repeated for multiple iterations

(epochs).

But ‘vanilla’ neural networks are not too useful for NLP tasks, primarily because these (i) require a fixed input size, and (ii) do not account for the sequential nature of text data. The first problem is overcome by assuming a bag of words model (BoW), in which an averaging of the individual word vectors leads to a fixed size representation of text. But then this fails to account for the order of the words in which they occur in the data. So, in such a model, the two conflicting statements 1 and 2 are exactly identical:

1. The pasta was nice, but the dessert was awful
2. The dessert was nice, but the pasta was awful

2.3.2 Convolution Neural Networks

Convolutional neural networks (CNN) were originally proposed by LeCun et al. (1995)[24] to overcome the problems posed by a fully connected neural network in processing unstructured data like images and text. Although initially the use of CNNs was limited to images, the various advantages offered like weight parameter sharing resulting in significantly less number of parameters thus preventing overfitting, the translational invariance through the use of a max-pool layer, and preserving local dependencies helped drive the use of CNNs in NLP tasks.

A convolutional layer primarily works by sliding a fixed-size window or filter or kernel (weights) over an input array composed of say pixels of an image or words in case of textual data. Sliding of the same filter over the different parts of the input is what enables it to use the same weights, hence reduced parameter set and also learn local dependencies really well. It is important to note here that the convolution performed in images is a 2D convolution and in case of text data is a 1D convolution. The subsequent pooling operation extracts the most important features and lowers the dimensionality of the convolved output. A max pooling operation, hence makes the model robust to minor variations or noise present in the data. CNNs in NLP would work pretty well if only short term (local) dependencies are required to achieve the task. But often in text data, we have long range dependencies such as in a multi sentence review text and CNNs might not be able to “remember” the context to perform well.

2.3.3 Recurrent Neural Networks

Recurrent Neural Networks (RNN), are an adaptation of the simpler feed forward neural networks (FFNN), offer a solution to various problems in text data all at once. RNNs i) can work with input sequences of arbitrary lengths, ii) share parameters across the sequence, and iii) can learn longer range dependencies. RNNs learn long range dependencies through incorporating “memory” in the system [25]. Contrasting with a FFNN, which can only pass information one way through the network, RNNs have loops in network allowing information

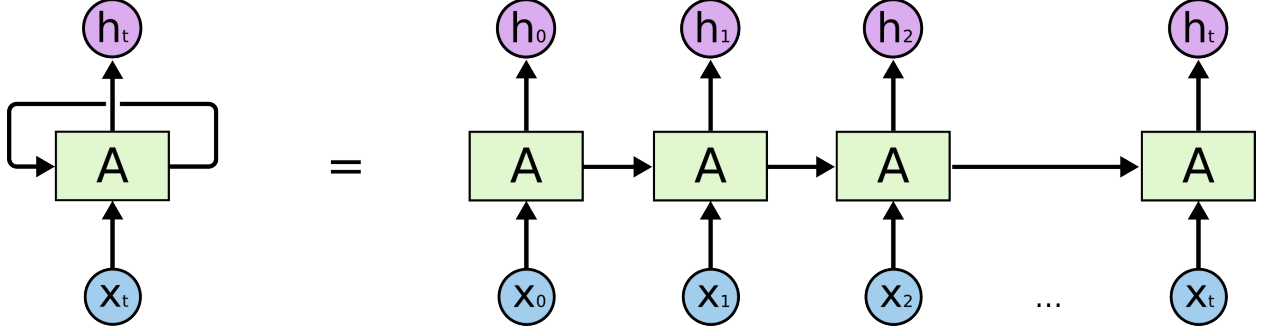


Figure 2.2: Diagrammatic representation of a RNN

to persist (See Figure 2.2). This chain-like structure is a natural choice to process a sequence data like text composed of words. This is done by splitting the output of the hidden layer, where one part of information is passed as usual to the next cell, while the other is "re-used" and combined with some additional information.

Although, in theory RNNs can indeed learn such long range dependencies in the data but some practical considerations make them unstable. Specifically, this problem is caused by the repeated multiplication of terms in a long sequence, which when the gradients are small lead to a compounding effect of exponential decrease i.e. vanishing gradients (leading to slow learning) and when the gradients are large lead to exponentially large gradients i.e. exploding gradients (leading to divergence) [26]. It must be noted here that, similar to FFNN, RNNs use backpropagation to calculate the gradients but have a fancier term for it called *backpropagation through time*.

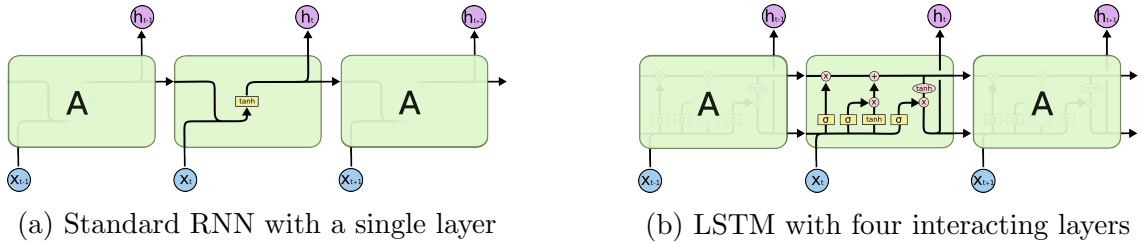


Figure 2.3: Comparison of a standard RNN with an LSTM cell [27]

2.3.4 Long Short Term Memory networks

Long Short Term Memory networks (LSTM) are a modification over the simple RNN (Figure 2.3) enabling them to perform a gradient based learning to incorporate long-term dependencies [28]. LSTMs enable the retention of long range memory through the use of *gates* and also an additional *context* [29]. These gates in each LSTM cell learn to selectively allow or restrict the passage of information similar to logic gates by setting the output value to 1 or 0. Input gate controls how much information to incorporate from the previous hidden layer, output gate regulates the flow of information to the next hidden layer and a forget gate which

computes the relevance of the previous hidden state and the current input to calculate the current cell state or the “long term memory”. At time t , The LSTM receives a new input vector x_t (including the bias term), as well as a vector of its output at the previous timestep, h_{t-1} :

$$a^t = \tanh(W_c x^t + U_c h^{t-1}) = \tanh(\hat{a}^t) \quad (2.1)$$

$$i^t = \sigma(W_i x^t + U_i h^{t-1}) = \sigma(\hat{i}^t) \quad (2.2)$$

$$f^t = \sigma(W_f x^t + U_f h^{t-1}) = \sigma(\hat{f}^t) \quad (2.3)$$

$$o^t = \sigma(W_o x^t + U_o h^{t-1}) = \sigma(\hat{o}^t) \quad (2.4)$$

2.3.5 Word embeddings

Generally, in NLP tasks we treat each word as a feature of the data. The use of one-hot encoding for each word results in a very high dimensional sparse representation of textual data. Such high dimensional sparse representation do not add much syntactic or semantic meaning to the representation of the data. The first problem primarily caused by treating the document as a bag-of-words (BoW) as noted earlier could be partly overcome by using bi-grams, tri-grams or higher n-grams. But still the semantic meaning is lost and large sparse representations lead to poor performance of deep learning systems [29]. The solution to this was proposed by Bengio et al. (2003) [30] by introducing a word embedding model which encoded words as dense vectors of much lower dimension in the feature space. This was subsequently followed by several other models, notably word2vec [31] and GloVe [32]. Typically, the learning of these word vectors is framed as a self-supervised problem achieved through the use of mainly two techniques: continuous bag-of-words (CBOW) and the skip-gram model. CBOW model uses the surrounding words as context to predict the word in the middle, while the skip-gram model uses the word to predict the context. The way such a model is trained determines how well the syntactic and semantic information is captured. Important here is to mention that, the embeddings can be trained at word level and also at the character level e.g. ELMo [33]. But the main point being, learning such dense vector representations reduce much of the need for hand-crafted feature engineering based on linguistic rules.

2.3.6 Loss function optimization

A supervised learning problem is solved by optimizing a suitable loss function \mathcal{L} which is a function of predicted value \mathcal{Y} and the true label \mathcal{Y} . The loss function measures how well the predicted value matches with the true observation. This loss function must be differentiable as variants of gradient descent algorithm like the ADAM optimizer [34] require to calculate the derivative of the loss function. For both our subtasks, we optimize the categorical cross-entropy loss function which is based on the principle of maximum likelihood and is the most common loss function for multi-class classification tasks. Here x (sentence or token) is an

example belonging to the training set \mathcal{X} , C is the number of label categories, $P_c(x)$ is the predicted probability of x belonging to class c , and $P_c^g(x)$ is the true label i.e. indicator variable of class c

$$\text{loss} = - \sum_{x \in \mathcal{X}} \sum_{c=1}^C P_c^g(x) \cdot \log(P_c(x)) \quad (2.5)$$

2.4 Aspect Extraction (Subtask1)

2.4.1 Problem statement

Given a set of sentences with pre-identified entities (e.g., restaurants), identify the aspect terms present in the sentence and return a list containing all the distinct aspect terms.

For example:

“I liked the service and the staff, but not the food” \longrightarrow {service, staff, food}

Here, we pose the aspect extraction problem as a sequence labelling task, where for each word in a sentence, our task is to classify it into three categories using the Begin-Inside-Out labelling scheme; $\{B, I, O\}$. B indicates beginning of the word, I indicates the continuing word (in case of multi-word aspect) and O indicates the non-aspect words.

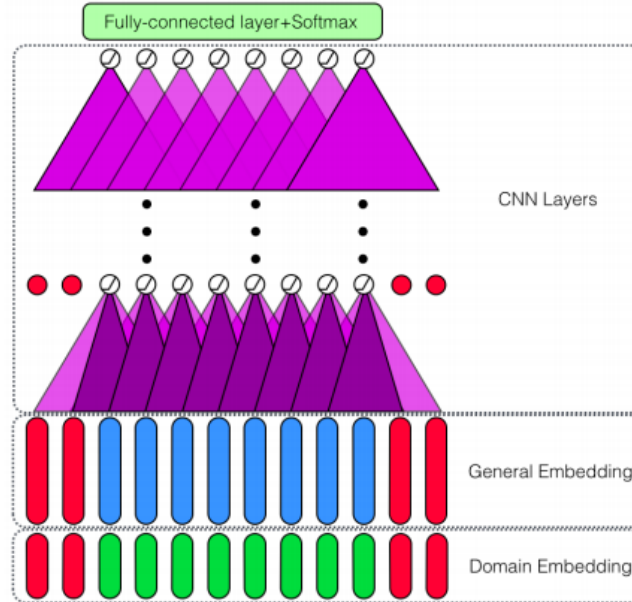


Figure 2.4: Architecture of the CNN aspect extraction system [19]

2.4.2 System Description

We implement the deep learning architecture presented in Xu et al.’s (2018) paper *Double Embeddings and CNN-based Sequence Labeling for Aspect Extraction* [19]. The aspect extraction system uses 300-dimensional Word2vec general embeddings concatenated with 100-dimensional domain-specific embeddings (restaurant and laptop) to obtain a 400-dimensional word vector representation. The deep learning system consists of four convolution layers followed by a fully connected dense layer which outputs softmax probabilities (See Figure 2.4).

2.4.3 Preprocessing

Consider the following example review text from the restaurant domain dataset (Review id (rid) = 50):

The fried rice is amazing here.

1. We tokenize each review text by splitting on whitespace, and additionally treat contractions like 'hv, 'nt and punctuation as a separate token.
2. We then obtain the lemmatized form (lemma) of the token.
3. Then we obtain the universal part-of-speech (upos) and language specific part-of-speech (xpos).
4. Finally, as described in Section 2.4.1, we tag each token as $\mathcal{Y} \in \{\text{B}, \text{I}, \text{O}\}$.

The final table obtained after the preprocessing step is shown below:

rid	tid	token	lemma	xpos	upos	label
50	1	The	the	DET	DT	O
50	2	fried	fry	VERB	VBN	B
50	3	rice	rice	NOUN	NN	I
50	4	is	be	AUX	VBZ	O
50	5	amazing	amazing	ADJ	JJ	O
50	6	here	here	ADV	RB	O
50	7	.	.	PUNCT	.	O

2.4.4 Preparing the input sequences

In this step we convert the text data into a numerical sequence for suitable ingestion into the CNN model as follows:

1. We assign a unique integer id to each token in the vocabulary and convert the tokenized sentences to a sequence of integers. This is an important step and serves as input for the embedding layer we put in our model.
2. Finally, we pad ('post') the sequences with the reserved padding token <PAD>: 0, up to a maximum length of 83.

2.4.5 Preparing word embeddings

For each word, we obtain the 300 dimensional GoogleNews¹ word embedding w^g and concatenate it with the 100 dimensional in-domain embedding² w^d (restaurant and laptop). We make use of gensim [35] to load the word2vec formatted GoogleNews binary file and the facebook fasttext in-domain embeddings file. Additionally, we experiment by concatenating separately the one-hot encoded upos (46 dimensional) and xpos (17 dimensional) part-of-speech tags w^{pos} . Hence we obtain the final word vector as $w = w^g \oplus w^d \oplus w^{pos}$. Here, $w \in \mathbb{R}^{446}$ if we use the upos tags, and $w \in \mathbb{R}^{417}$ if we use the xpos tags.

2.4.6 Building the architecture

We use Keras [36] to code up the deep learning architecture. We start by adding an embedding layer which takes as input a sequence of integer encoded tokens (as described in Section 2.3.2) and outputs a d-dimensional representation of the token. We derive this output representation by specifying the embedding matrices created in Section 2.4.5 and keep it non-trainable, as on a small training set, it will result in worse performance. The first convolution layer has 128 filters with two kernel sizes $k = 3$, and $k = 5$, followed by three convolution layers each with 256 filters and kernel size $k = 5$ with "same" padding. As described earlier the sliding window kernel helps the model learn local dependency relations in the surrounding context of the words. We set the dropout rate in the first layer = 0.55 to regularize the network in order to prevent overfitting. Although, we note in Section 2.3.2, that a max pooling operation "extracts" the most important information while denoising the input, the authors report worse performance with a MAXPOOL layer. They argue that in such a task of sequence tagging, max pooling operation has tendency to lose individual word position.

For training we take a batch size of 128, and fix the learning rate of the Adam optimizer to be 0.0001, as in our observation, a higher learning led to highly fluctuating gradients. This

¹<https://code.google.com/archive/p/word2vec/>

²<https://howardhsu.github.io/dataset/>

is followed by a fully connected 3 neuron dense layer with softmax activation for classifying the labels according to the IOB scheme. Figure 2.5 shows the simple plot of the architecture obtained from Keras.

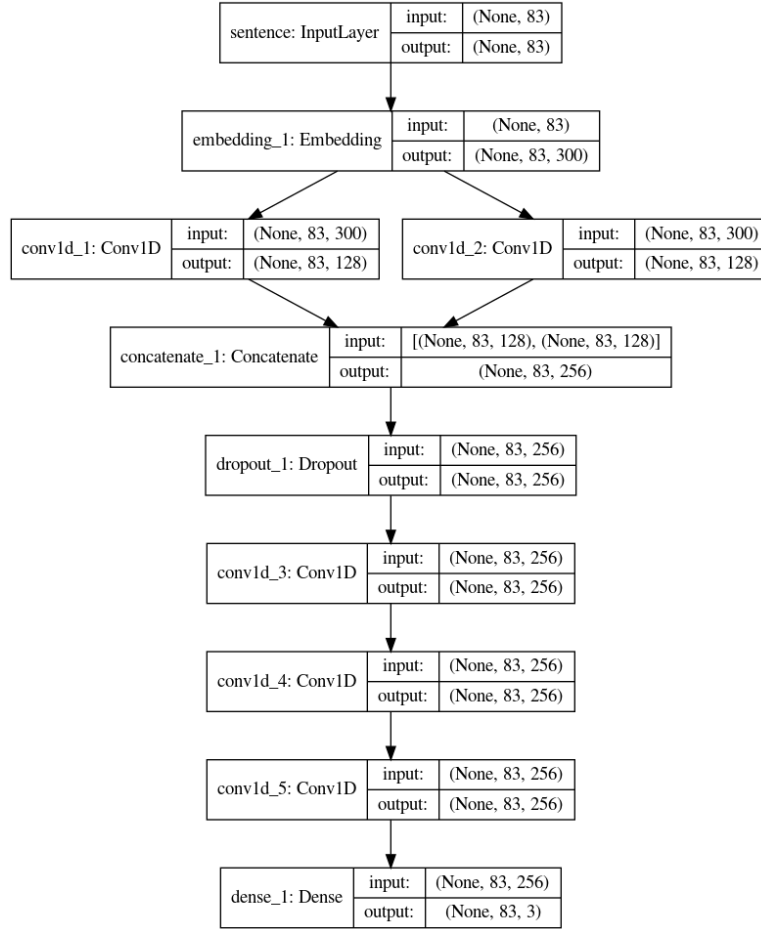


Figure 2.5: Keras plot of the CNN architecture

2.5 Aspect Polarity Detection (Subtask 2)

2.5.1 System Description

We implement the deep learning architecture presented in Tang et al.'s (2015) paper *Effective LSTMs for Target-Dependent Sentiment Classification* [37]. We choose the simple LSTM and the target dependent LSTM (TD-LSTM) architecture mentioned in the paper. As a modification to the simple LSTM, TD-LSTM is composed of two LSTMs, the left $LSTM_L$, which processes the text from left to right and the right $LSTM_R$, which processes the text from right to left up till the target i.e. the aspect term (See Figure 2.6). Finally, the two hidden states are concatenated and output softmax probabilities are obtained to classify the

sentiment.

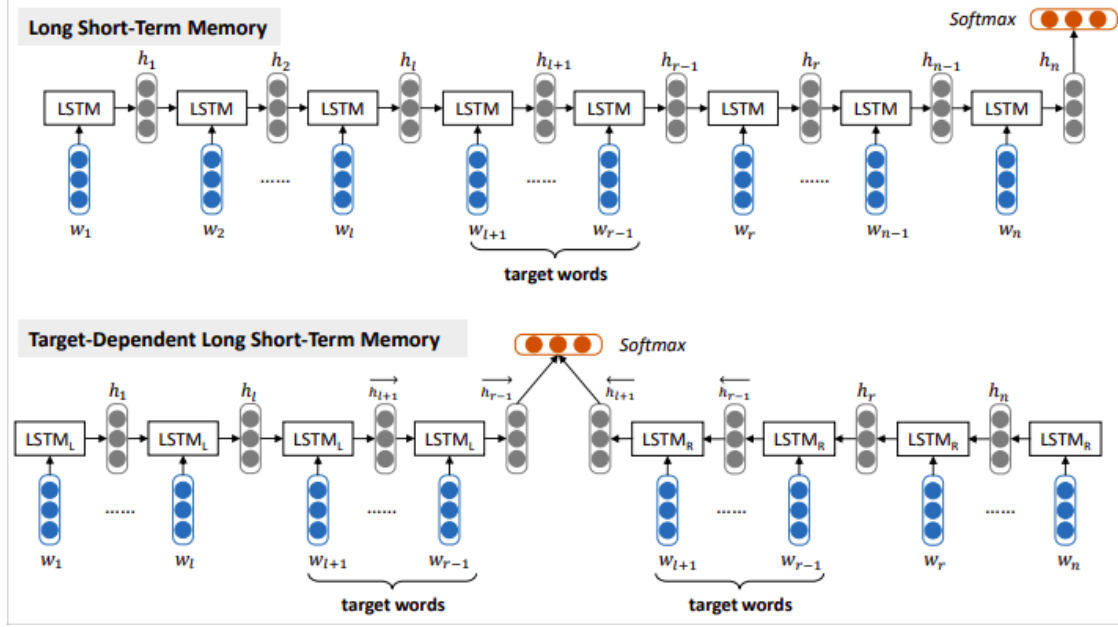


Figure 2.6: Architecture of the LSTM aspect polarity detection system [37]

2.5.2 Problem Statement

For a given set of aspect terms within a sentence, determine whether the polarity of each aspect term is positive (1), negative (-1) or neutral (0).

For example:

“I loved their fajitas” $\rightarrow \{fajitas : positive\}$

“I hated their fajitas, but their salads were great” $\rightarrow \{fajitas : negative, salads : positive\}$

“The fajitas are their first plate” $\rightarrow \{fajitas : neutral\}$

2.5.3 Preprocessing

We mostly follow the same preprocessing procedure as for the aspect extraction subtask (Section 2.4.3), except make the following changes:

1. Punctuations are removed altogether
2. Contractions are handled separately by decontracting them. For e.g. 'hv \rightarrow have
3. We skip the lemmatization step and simply use the lowercased form of the token
4. Finally, each sentence (not token) is assigned a polarity label as $\mathcal{Y} \in \{-1, 0, 1\}$.

2.5.4 Preparing the input sequences

In this step we convert the text data into a numerical sequence for suitable ingestion into the LSTM models as follows:

1. For the simple LSTM, we only require one sequence as our training feature \mathcal{X} , so we keep the preprocessed sentence *as it is*.
2. For the TD-LSTM, we create two sentences, one to the left of the target and other to the right of the target, both containing the aspect term.
3. This is followed by a simple tokenization process on the created sentences by splitting on a whitespace
4. Then, similar to Section ?? we assign a unique integer id to each token in the vocabulary and convert the tokenized sentences to a sequence of integers for input to the embedding layer.
5. Finally, we pad ('post') the sequences with the reserved padding token $\langle \text{PAD} \rangle$: 0, upto a maximum length of 80 for both the simple LSTM and TD-LSTM.

2.5.5 Preparing word embeddings

We follow the same procedure for generating the word embeddings as for the aspect extraction subtask (Section 2.4.5) albeit with the following changes:

1. Instead of the GoogleNews embedding, we use glOve³ embeddings available in various dimensions (50d, 100d, 200d, 300d).
2. We similarly concatenate the in-domain embeddings with the general embeddings to obtain the double embedding. This has not been done in the original LSTM paper as it was published in 2015, while Xu et al.'s double embedding based system came later in 2018.
3. Instead of concatenating a one representation of 46d upos or 17d xpos tags with the embeddings, we choose a smaller set of 6 part-of-speech tags comprising of {noun, verb, adjective, adverb, preposition, conjunction}.

2.5.6 Building the architecture

For this task too, we use Keras to code the architecture. The simple LSTM is created by adding an embedding layer similar to the aspect extraction task, to output the sequence of word vectors. This is followed by a many-to-one LSTM layer takes the sequence of these word vectors and outputs a hidden representation in the last cell. We fix the dimensionality

³<https://nlp.stanford.edu/projects/glove/>

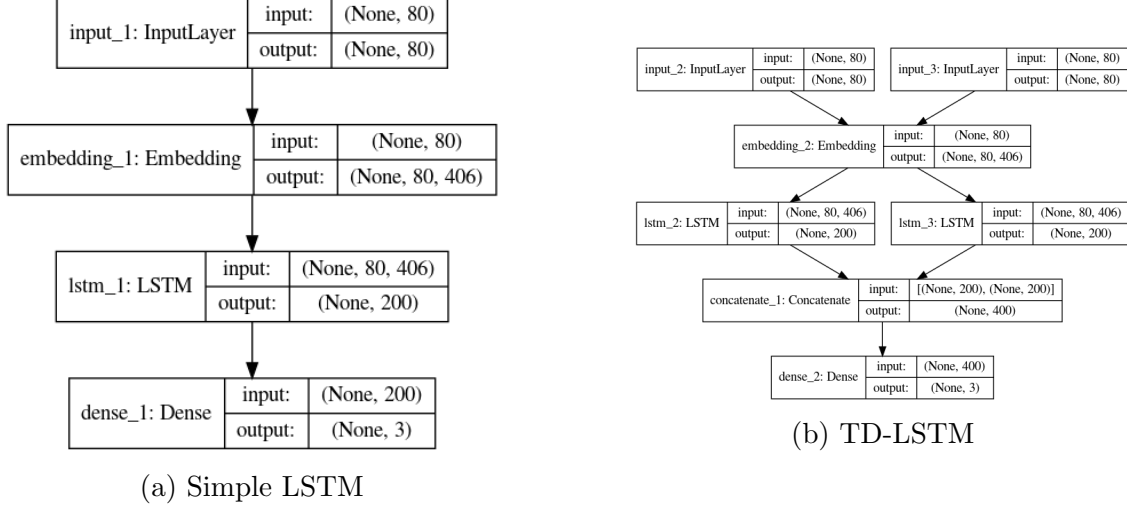


Figure 2.7: Keras plot of the implemented LSTM architectures

of this output as 200 (units). This is followed by a fully connected 3 neuron dense layer with softmax activation for classifying the polarity.

In case of TD-LSTM, we process each of the left and right integer encoded sequences through its own embedding layer followed by identical layers $LSTM_L$ and $LSTM_R$, each with 200 units. The final hidden representations from both the LSTMs are concatenated to form a 400d tensor. As before, this is followed by a fully connected dense layer with softmax activation to obtain multi-class probabilities. An important detail to note is that the authors specify the right $LSTM_R$ to process the sequence from right to left i.e. from the end of the sentence to the aspect term. We achieve this in Keras by specifying *go_backwards = True* in the right $LSTM_R$. The authors argue that this strategy helps to take into account the semantic meaning of the aspect term relevant for sentiment classification, as the final hidden representation in both the LSTMs are outputted from the head of the LSTM module which takes as input the aspect term.

In both the cases we optimize the categorical cross entropy loss function with ADAM optimizer (learning rate = 0.01) and batch size 128. The Keras plot of the architectures are shown in the Figure 2.7.

Chapter 3

Results and Discussions

3.1 Performance measures

We report the performance of the deep learning systems using the following evaluation metrics for each class in the multi-class classification tasks:

1. **Precision** - This tells us how precise the classifier's prediction are and is defined as the proportion of correctly predicted positive class out of all the positive examples

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (3.1)$$

2. **Recall** - This is defined as the proportion of correctly classified positive examples out of all the positive predicted examples

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (3.2)$$

3. **F1-score** - This metric maintains a balance between the precision and recall for the classifier and is simply the harmonic mean of the precision and the recall

$$\text{F1-score} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \quad (3.3)$$

Along with reporting these classification metrics for each class, we also report the macro and the weighted average of the metrics, which are defined as:

- **macro**: the averaged score is obtained without considering the proportion of each label in the dataset
- **weighted**: a weighted average score is obtained by considering the proportion of each label in the dataset

3.2 Aspect Extraction (Subtask 1)

In this task, we experimented with the concatenation of xpos and upos tags with the double embeddings used in Xu et al.’s system. We report performance gains (f1-score) in both the cases of concatenation of xpos and upos tags, with slightly higher metrics in the latter. Table 3.1 compares the metrics for the three systems. Here, we note that the original paper used the SemEval 2016 Restaurant data, which is different from the SemEval 2014 Restaurant domain data that we have used.

DL System	Precision	Recall	F1-score
DE-CNN	0.85	0.76	0.79
DE-CNN + xpos	0.86	0.76	0.80
DE-CNN + upos	0.88	0.79	0.82

Table 3.1: Comparison of performance metrics (macro) for the three AE DL systems

Table 3.2 shows the performance of the best model (DE-CNN + upos) in classifying the individual labels **B**, **I** and **O**. The performance of the model is best in classifying the non-aspect term label **O**, which is reasonable since the majority of the tokens are not aspect terms. It also performs reasonably well in classifying the beginning of the aspect term label **B**, hence single word aspects are handled well by the system. But it performs significantly worse in labelling multi-word aspects, as evident from the low performance on **I** label, which denotes the inside i.e. continuation of the aspect term.

This is easier to see if we relax the labelling scheme to only include binary labels indicating aspect term as **I**, and non-aspect terms as **O**. Higher overall performance metrics (See Table 3.3) in this scheme gives credence to the conclusion that the model performs better in extracting single word aspect terms and performs poorly in multi-word aspect terms.

	Precision	Recall	F1-score	Support
B	0.82	0.85	0.83	1132
I	0.87	0.54	0.67	571
O	0.95	0.97	0.96	11049
macro avg	0.88	0.79	0.82	12752
weighted avg	0.93	0.94	0.94	12752

Table 3.2: DE-CNN + upos performance for IOB labels

	Precision	Recall	F1-score	Support
I	0.90	0.81	0.85	1703
O	0.95	0.97	0.96	11049
macro avg	0.93	0.89	0.91	12752
weighted avg	0.94	0.95	0.95	12752

Table 3.3: DE-CNN + upos performance for IO labels

3.3 Aspect Polarity Detection (Subtask 2)

In this task, we borrowed the technique of concatenating the general embedding (glove 300d) with the in-domain restaurant embeddings (fasttext 100d) from the previous task for training the simple LSTM and the TD-LSTM. In addition to this, we experimented with the concatenating the 6 dimensional one-hot representation of the pos tags. TD-LSTM outperforms simple LSTM in all the metrics, which is reasonable to expect as the simple LSTM has no input information about the position of the aspect term need to classify the directed sentiment. The concatenation of the 100d restaurant embeddings and the pos tags both lead to minor improvements as well in some performance metrics (See Table 3.4; shown in red are the reported scores in the original paper).

DL System	Precision	Recall	F1-score
LSTM	0.64	0.62	0.62
DE-LSTM	0.66	0.64	0.63
DE-LSTM + pos	0.67	0.63	0.65
TD-LSTM	0.67	0.65	0.65
DE-TD-LSTM	0.68	0.65	0.65
DE-TD-LSTM + pos	0.70	0.64	0.66

Table 3.4: Comparison of performance metrics (macro) for the APD DL systems

Table 3.5 shows the performance of the best model (f1-score) in classifying the polarity labels positive (1), negative (-1) and neutral (0). The model has both high precision and recall for the positive sentiment, followed by the negative sentiment which has significantly less scores. The worst performance is noted in classifying the neutral label, a low recall implies model is not keen on predicting the neutral class, and a low precision implies that when it does, it is highly inaccurate. Here, it must be noted that the original implementation

in the paper was evaluated for a different dataset (twitter [38]), hence we have not included the scores for comparison.

	Precision	Recall	F1-score	Support
Positive	0.83	0.93	0.88	728
Neutral	0.61	0.36	0.45	196
Negative	0.68	0.65	0.66	196
macro avg	0.70	0.64	0.66	1120
weighted avg	0.77	0.78	0.77	1120

Table 3.5: DE-TD-LSTM + pos performance for polarity classification

Chapter 4

Conclusions

4.1 Summary of Conclusions

In this project, we have made an attempt to implement the deep learning architectures in the literature in the context of aspect based sentiment analysis. These deep learning approaches have replaced the need for manual feature engineering to some extent as deep learning allows machines to learn suitable hidden representations of the data [39]. This has enabled deep learning approaches to consistently outperform both the traditional machine learning algorithms and the purely linguistic approaches. We have demonstrated how minimal feature engineering like the use of double embeddings, and the incorporation of linguistic information (part-of-speech tags) can enhance performance of the deep learning systems. Deep learning systems are undeniably enticing for the researchers in the NLP community due to their ability to learn idiosyncrasies of the natural language which may be harder to extract through rule-based linguistic approaches. But all of this comes at a cost:

- **Lack of interpretability of DL systems:** this poses problem in conducting a thorough error analysis essential to improving the system.
- **Reliable labelled data:** Further, for such a task the deep learning systems are supervised requiring a labelled dataset, and hence are only as good as the data they are trained on. Issues relating to inter-annotator disagreement pose problems in the reliability of annotations and generating adequate labelled data, which is especially important in the context of sentiment analysis. Although, some unsupervised techniques do exist [40, 41], but these require a much larger corpus than their supervised counterparts.
- **Domain dependence:** In this project, part of our focus was to achieve higher performance metrics on data belonging to a particular domain i.e. restaurants. Training DL systems require huge computational resources, time and arguably some arbitrary decisions in the process of tuning the hyperparameters. While linguistic insights will almost certainly transfer well to a different domain, for e.g. laptops, there is no guaran-

tee that a DL system will perform equally or even achieve a decent level of performance in the other domain.

Although it must be pointed here that purely linguistic approaches are not without their flaws as well. A sophisticated linguistic approach can lead to much complexity due to the various sub components associated with it like tagging the word with parts-of-speech, parsing the dependency tree, choosing seed words in an opinion lexicon [42], and a complex algorithm involving hand-crafted linguistic rules.

4.2 Future scope

Moving forward, we would like to perform an in-depth error analysis of the deep learning systems to better understand the kind of mistakes the system makes. For e.g. errors could percolate down from the pos tagger used to tag each word with the part-of-speech tags. Further, we would like to explore ways in which the theory of linguistics could be leveraged in cutting edge deep learning systems to achieve high performance natural language systems. Notable examples of such approaches combining rule based linguistics with deep learning are the Poria et al.'s [17] CNN based extraction + Linguistic Patterns (LP), and Ray et al.'s [43] 7-layer CNN with rule based linguistics and clustering methods for aspect category classification. Another exciting development which is at the cutting edge of NLP using DL, are the transformer neural networks. They are based on a novel architecture that are built from ground up to handle long-range dependencies encountered in textual with much more ease than standard RNN or even LSTMs. First introduced in the paper *Attention Is All You Need* in 2017 [44], these networks are based on the core idea of the attention mechanism. Another interesting aspect of these transformer networks, besides outperforming RNNs on many standard NLP tasks with a lower computational cost, is that it is possible to visualize what parts of a sentence the network deems important or *attends* to when the processing a single token¹. Such a mechanism allows to gain insights in the way information flows through the network thus adding interpretability to the system. This is an exciting area of machine learning known as interpretable machine learning [45]. We conclude this report by quoting the following lines from the book *Interpretable Machine Learning* by Christopher Molnar [46]:

The goal of science is to gain knowledge, but many problems are solved with big datasets and black box machine learning models. The model itself becomes the source of knowledge instead of the data. Interpretability makes it possible to extract this additional knowledge captured by the model.

¹<https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html>

Bibliography

- [1] B. Liu, “Sentiment analysis and opinion mining,” *Synthesis lectures on human language technologies*, vol. 5, no. 1, pp. 1–167, 2012.
- [2] E. Cambria, S. Poria, A. Gelbukh, and M. Thelwall, “Sentiment analysis is a big suitcase,” *IEEE Intelligent Systems*, vol. 32, no. 6, pp. 74–80, 2017.
- [3] B. Pang and L. Lee, “Opinion mining and sentiment analysis,” *Foundations and trends in information retrieval*, vol. 2, no. 1-2, pp. 1–135, 2008.
- [4] T. Wilson, J. Wiebe, and P. Hoffmann, “Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis,” *Computational linguistics*, vol. 35, no. 3, pp. 399–433, 2009.
- [5] C. Zhang, D. Zeng, J. Li, F.-Y. Wang, and W. Zuo, “Sentiment analysis of chinese documents: From sentence to document level,” *Journal of the American Society for Information Science and Technology*, vol. 60, no. 12, pp. 2474–2487, 2009.
- [6] L. Zhang and B. Liu, “Aspect and entity extraction for opinion mining,” in *Data mining and knowledge discovery for big data*, pp. 1–40, Springer, 2014.
- [7] H.-Y. Chen and H.-H. Chen, “Implicit polarity and implicit aspect recognition in opinion mining,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 20–25, 2016.
- [8] I. Cruz, A. F. Gelbukh, and G. Sidorov, “Implicit aspect indicator extraction for aspect based opinion mining,” *Int. J. Comput. Linguistics Appl.*, vol. 5, no. 2, pp. 135–152, 2014.
- [9] M. Hu and B. Liu, “Mining and summarizing customer reviews,” in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 168–177, 2004.
- [10] M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, and S. Manandhar, “Semeval-2014 task 4: Aspect based sentiment analysis. proc. 8th int. workshop on semantic evaluation (semeval 2014),” 2014.

- [11] G. Paltoglou and M. Thelwall, “Sensing social media: A range of approaches for sentiment analysis,” in *Cyberemotions*, pp. 97–117, Springer, 2017.
- [12] K. Dave, S. Lawrence, and D. M. Pennock, “Mining the peanut gallery: Opinion extraction and semantic classification of product reviews,” in *Proceedings of the 12th international conference on World Wide Web*, pp. 519–528, 2003.
- [13] J. Wiebe, R. Bruce, M. Bell, M. Martin, and T. Wilson, “A corpus study of evaluative and speculative language,” in *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*, 2001.
- [14] A.-M. Popescu and O. Etzioni, “Extracting product features and opinions from reviews,” in *Natural language processing and text mining*, pp. 9–28, Springer, 2007.
- [15] S. Raju, P. Pingali, and V. Varma, “An unsupervised approach to product attribute extraction,” in *European Conference on Information Retrieval*, pp. 796–800, Springer, 2009.
- [16] S. Kiritchenko, X. Zhu, C. Cherry, and S. Mohammad, “Nrc-canada-2014: Detecting aspects and sentiment in customer reviews,” in *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pp. 437–442, 2014.
- [17] S. Poria, E. Cambria, and A. Gelbukh, “Aspect extraction for opinion mining with a deep convolutional neural network,” *Knowledge-Based Systems*, vol. 108, pp. 42–49, 2016.
- [18] Y. Wang, M. Huang, X. Zhu, and L. Zhao, “Attention-based lstm for aspect-level sentiment classification,” in *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 606–615, 2016.
- [19] H. Xu, B. Liu, L. Shu, and P. S. Yu, “Double embeddings and cnn-based sequence labeling for aspect extraction,” *arXiv preprint arXiv:1805.04601*, 2018.
- [20] C. Sun, L. Huang, and X. Qiu, “Utilizing bert for aspect-based sentiment analysis via constructing auxiliary sentence,” *arXiv preprint arXiv:1903.09588*, 2019.
- [21] G. Ganu, N. Elhadad, and A. Marian, “Beyond the stars: improving rating predictions using review text content,” in *WebDB*, vol. 9, pp. 1–6, Citeseer, 2009.
- [22] D. Jurafsky and J. Martin, “Speech and language processing (3rd (draft) ed.),” 2019.
- [23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *nature*, vol. 323, no. 6088, pp. 533–536, 1986.
- [24] Y. LeCun, Y. Bengio, *et al.*, “Convolutional networks for images, speech, and time series,” *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

- [25] D. Jurafsky and J. H. Martin, “Speech and language processing: An introduction to speech recognition, computational linguistics and natural language processing,” *Upper Saddle River, NJ: Prentice Hall*, 2008.
- [26] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.
- [27] C. Olah.
- [28] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [29] Y. Goldberg, “A primer on neural network models for natural language processing,” *Journal of Artificial Intelligence Research*, vol. 57, pp. 345–420, 2016.
- [30] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, “A neural probabilistic language model,” *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.
- [31] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.
- [32] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.
- [33] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” *arXiv preprint arXiv:1802.05365*, 2018.
- [34] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [35] R. Řehůřek and P. Sojka, “Software Framework for Topic Modelling with Large Corpora,” in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, (Valletta, Malta), pp. 45–50, ELRA, May 2010. <http://is.muni.cz/publication/884893/en>.
- [36] F. Chollet *et al.*, “Keras.” <https://keras.io>, 2015.
- [37] D. Tang, B. Qin, X. Feng, and T. Liu, “Effective lstms for target-dependent sentiment classification,” *arXiv preprint arXiv:1512.01100*, 2015.
- [38] 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 (volume 2: Short papers)*, pp. 49–54, 2014.

- [39] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [40] A. Garcia-Pablos, M. Cuadros, and G. Rigau, “V3: Unsupervised aspect based sentiment analysis for semeval2015 task 12,” in *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 714–718, 2015.
- [41] A. García-Pablos, M. Cuadros, and G. Rigau, “W2vlda: almost unsupervised system for aspect based sentiment analysis,” *Expert Systems with Applications*, vol. 91, pp. 127–137, 2018.
- [42] D. Jovanoski, V. Pachovski, and P. Nakov, “On the impact of seed words on sentiment polarity lexicon induction,” in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp. 1557–1567, 2016.
- [43] P. Ray and A. Chakrabarti, “A mixed approach of deep learning method and rule-based method to improve aspect level sentiment analysis,” *Applied Computing and Informatics*, 2019.
- [44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, pp. 5998–6008, 2017.
- [45] F. Doshi-Velez and B. Kim, “Towards a rigorous science of interpretable machine learning,” *arXiv preprint arXiv:1702.08608*, 2017.
- [46] C. Molnar, *Interpretable Machine Learning*. Lulu. com, 2020.