DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2019

Identifying Action related Dialogue Acts in Meetings

Research into identifying Action Items, Decisions, and Ideas from multi-party meeting transcripts

SUHAS SHESHADRI

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Identifying Action related Dialogue Acts in Meetings

Research into identifying Action Items, Decisions, and Ideas from multi-party meeting transcripts

SUHAS SHESHADRI
suhass@kth.se

Master in Computer Science
Date: September 30, 2019
Supervisor: Johan Boye
Examiner: Jussi Karlgren
School of Electrical Engineering and Computer Science
Host Company: Seavus Stockholm AB
Company Supervisor: Reijo Silander
Swedish title: Identifiering av handlingsrelaterade dialoghandlingar i möten

Abstract

This thesis aims to determine the dialogue acts such as action items, decisions and ideas put forth in multi-party meetings or dialogues. Conversational meetings can generally contain several such dialogue acts and pinpointing them might not be a straight-forward operation. At times, it may be as simple as *"The business team will come up with a plan"* or as ambiguous as *"Let us go with his design"*. Due to such regularly occurring instances where the context of the sentence plays a crucial factor and due to the small quantity of available data, it is challenging to apply traditional neural models on them. Natural language processing, of late, has evidenced many innovations and advances, one of them being the pre-trained language models. These off-the-shelf available language models are trained on huge amounts of data and can be applied to an array of natural language processing tasks, with almost none to little training (fine-tuning). This thesis investigates the viability of one such language model called "Bidirectional Encoder Representations from Transformers (BERT)" in the task of identifying dialogue acts. Lately, BERT based models have proved their worth by excelling in several natural language processing tasks and scoring state-of-the-art performances. Although, they have not yet been applied to the area of meetings and dialogue acts.

Results show that BERT based models have outperformed the current state-of-the-art models yet again and by a definitive margin. This exhibits the versatility of BERT and corroborates its recent success in the field. It also opens new doors, which were earlier not possible, in the area of multi-party meetings and more generally, other fields of natural language processing. Another intriguing aspect of language models like BERT is the absence of dependency between the pre-trained data and the fine-tuned data. This thesis creates a steppingstone for further research on conversational meetings and dialogues employing pre-trained language models.

Sammanfattning

Den här uppsatsen fokuserar på att bestämma dialoghandlingar så som handlingar, beslut och idéer som presenteras i flerpartsmöten eller dialoger. Konversationsmöten kan i allmänhet innehålla flera sådana dialoghandlingar och det kan vara utmanande att identifiera dem. Ibland kan det vara så enkelt som *Företagsteamet kommer med en plan"* eller så tvetydigt som *Låt oss köra på hans design"*. På grund av sådana regelbundet förekommande fall där kontexten av meningen är en avgörande faktor och på grund av den lilla mängden tillgängliga data, är det utmanande att tillämpa traditionella neurala nätverks modeller på dem. På senare tid har naturlig språkbehandling sett många innovationer och framsteg, en av dem är de förtränade språkmodellerna. Dessa lättillgängliga språkmodeller är tränade på enorma mängder data och kan tillämpas på en mängd naturliga språkbehandlingssuppgifter, med nästan ingen till liten träning (finjustering). Den här uppsatsen undersöker möjligheten hos en sådan språkmodell som kallas Bidirectional Encoder Representations from Transformers (BERT)"i uppgiften att identifiera dialoghandlingar. På senare tid har BERT-baserade modeller visat sitt värde genom att utmärka sig i flera naturliga språkbehandlingssuppgifter och med prestanda i forskningsfronten. Däremot har de inte tillämpats på området för möten och dialoghandlingar.

Resultaten visar att BERT-baserade modeller har överträffat de nuvarande moderna modellerna igen och med en signifikant marginal. Detta visar BERTs mångsidighet och bekräftar att den senaste framgången i området. Det öppnar också nu dörrar, som tidigare inte var möjliga, inom området för flerpartsmöten och mer allmänt andra områden för naturligt språkbehandling. En annan spännande aspekt av språkmodeller som BERT är frånvaron av beroende mellan förtränad data och finjusterad data. Den här uppsatsen skapar ett steg för ytterligare forskning om analys av möten och dialoger med förtränade språkmodeller.

Acknowledgement

This thesis would not be complete without the help and support of many people, and a big heartfelt **Thank You** for everyone. I sincerely want to thank each and everyone who have been a part of my thesis and my life during the last two years. I might have forgotten to name some of you here for which I apologize in advance.

Thanks to Seavus for providing me the opportunity to work on this challenging project, Mr. Reijo Silander, the Project Manager and Head of R&D for the regular guidance, be it technical or moral, as well as all the other employees who were great company to work and interact with. I would like to appreciate the great work environment and the wonderful learning experience provided.

I would like to thank my supervisor, Mr. Johan Boye, associate professor in the Speech Technology group at the School of Electrical Engineering and Computer Science at KTH, for all the assistance provided during this period. You have been a great influence for my interest in NLP from the first day of your lecture. The interactive sessions we had have been eye opening and provided me with immense knowledge and directions in which I could drive this thesis forward. The attention to detail, warm work environment paired with no urgency was like icing on the cake, just made this work even more enjoyable.

I want to thank my examiner Mr. Jussi Karlgren, computational linguist and adjunct professor in language technology at KTH, for the thought provoking sessions and guidance provided during the course of this thesis.

I am greatful to all my family and friends, who have always been there for me, in more possible ways than I could ever wish for. Sanjana, Sudeep, Abhishek, Bandhavi, Risha, Avinash, Pradeepa, Ishta, Karthik, Srivani, Tejaswi, Susmitha, Dhanush, Pratibha, Rithika, Emilio, John and many others whom I could not mention, thank you one and all from the bottom of my heart.

Last but most importantly, a huge Thank You to my wonderful Parents, **Mr. Sheshadri** and **Mrs. Seethalakshmi** for all the love, support and encouragement, without whom I would not be where I am today.

Contents

1 Introduction **1**

1.0.1 Task 1: Decision Detection 3

1.0.2 Task 2: Dialogue Act Detection 3

1.1 Research Question 4

1.2 Report Structure 4

2 Background **5**

2.1 Basic Concepts 5

2.1.1 Similarity Measure 5

2.1.2 Frequency Measure 6

2.1.3 Tokenization, POS tag and NER 6

2.1.4 Text preprocessing 7

2.1.5 Dialogue Act 8

2.1.6 NLP open source libraries 9

2.2 Learning: Supervised and Unsupervised 10

2.3 SVM 11

2.4 Tree Based Learning 12

2.4.1 Decision Tree 12

2.4.2 LGBM 13

2.5 Natural Language Processing 15

2.6 Language Representation 15

2.6.1 Recurrent Neural Networks 18

2.6.2 BERT 20

2.6.3 Pre-Training 22

2.6.4 Fine-Tuning 25

2.7 Text Summarization: Extractive and Abstractive 25

2.8 Evaluation 26

2.9 Previous Work 28

3 Methods **30**
3.1 Dataset . . . 30
3.2 Data Extraction and Description . . . 31
3.3 Preprocessing . . . 34
3.4 Classification . . . 35
3.4.1 LGBM . . . 35
3.4.2 BERT . . . 36
3.5 Decision Detection . . . 39
3.6 Dialogue Act Detection . . . 39
3.7 Post-processing . . . 40
3.8 Summarizing . . . 41

4 Results **42**
4.1 Top Performing Model . . . 42
4.2 Decision Detection . . . 43
4.3 Dialogue Acts Detection . . . 44
4.4 Action Report . . . 45

5 Discussion **47**
5.1 General Discussion . . . 47
5.2 Language Model Fine-tuning . . . 48
5.3 Text pre-processing and post-processing . . . 48
5.4 Practical Usefulness . . . 49
5.5 Social and Ethical Impact . . . 49

6 Conclusions **50**
6.1 Conclusions . . . 50
6.2 Future Work . . . 51

Bibliography **52**

A SpaCy NER Types and Descriptions **58**

B Supplementary Results **59**
B.1 Test 1: Varying Learning Rate . . . 59
B.2 Test 2: Varying Epochs . . . 59
B.3 Test 3: Parts of Speech . . . 59
B.4 Test 4: Named Entity Recognition . . . 60
B.5 Test 5: Meeting by Meeting Training . . . 60
B.6 Test 6: Markings for each meeting . . . 60

B.7 Test 7: Speaker Info 61
B.8 Test 8: Ensemble Model 61

Chapter 1

Introduction

Meetings are a general way to exchange information, share knowledge, discuss problems and come up with solutions. There are around 11 million business meetings [29] taking place in the US alone during a typical day. The authors state that meetings dominate worker's and manager's time and also reveal that they generally end up being expensive, unproductive and dissatisfying [46]. Yet, meetings are essential and their durations are steadily increasing over the previous decades. More than half of the time spent in these meetings are of no use, while costing over $37 billion annually in the US alone [26].

Minutes of Meetings (MoM), or informally called Notes are on-the-spot written records of a meeting. Usually, the agenda, attendees, actions agreed upon, decisions made, or other important points worth noting down make up the contents of MoM[1]. This routine and time-consuming job is a prime candidate for automatization, with the help of machine learning. Such automatically produced meeting reports can provide value to people by providing quick access of past meeting contents. This has resulted in research being done to efficiently automate the task of producing reports from a meeting transcript.

A system which generates this for a business will add value not just by providing instant reports but also by reducing the cost and time by overcoming the manual and tedious work of reading through a meeting transcript. These reports provide a synopsis of the project, along with the structure and flow of the project over the course of time. These documents also provide an opportunity for employees who could not attend the meeting, a chance to catch up. They can be analyzed to gauge the skills, knowledge and deficiency of the

[1] https://en.wikipedia.org/wiki/Minutes

team. A central aspect to our organizational life is to review decisions [35] [45]. For example, when a new person is introduced to a project, reviewing all the major decisions taken during the project would be a good point of understanding the current situation. This is generally not straight-forward unless all the previous meetings have been annotated or explicit decisions have been noted. Even providing a keyword-based searching platform does not make it easy to retrieve information about the decision points as shown in a study [36].

Not to forget, analyzing meeting transcripts is not straight-forward as they may contain disfluencies, repetitions, spelling mistakes and half spoken words, or irrelevant sentences among others. Also, the conversation is often between two or more people. Figure 1.1 shows a snippet of a meeting transcript where these challenges are evident.

```
A 1535.10 so I will do something with button A_.
A 1550.01 So maybe button A_ is the power button, okay.
D 1550.55 Mm-hmm.
B 1551.87 Mm-hmm.
A 1552.64 Um and then it needs to be able to send the signal out
to the device itself which is the receiver here.
A 1557.86 Um and I think that's about it in terms of my design
um.
A 1562.87 It's fairly general, um and I guess the purpose of
this is also not to restrict you in in the way you're thinking,
like um voice recognition, right,
A 1562.87 um,
A 1562.87 if it's something which is important then we just add
more power rather than having a thing that we don't have enough
power.
B 1567.66 Mm-hmm.
A 1578.28 So it's not really a constraint in that sense,
A 1578.28 but I mean these are functionally, you know, the base,
what the technology has to do.
B 1578.57 'Kay.
D 1579.71 Mm-hmm.
```

Figure 1.1: A snippet of meeting transcript after extracting to text from. The format of each sentence is the speaker id followed by the start time of the sentence followed by the actual utterance of the sentence.

Pairing the above information with the latest advances in NLP, this thesis presents method to identify the action items, decisions, or ideas from such transcripts. Let us first start by understanding what these terms actually mean. Decision is not the same as action, for example, *deciding* to become a millionaire is absolutely different from actually performing *actions* in pursuit of it. Just deciding on something does not make it happen. In management terms, *Action items* are actions, events or tasks that must take place. An *idea* is just

a thought or a suggestion which can possibly become a decision or even pave way for an action.

Figure 1.2 shows one such fragment of desired report containing Action related Dialogue Acts. This can later be fed into a summarizer to condense and retain only the important parts of such a report.

```
So I will do something with button A.
So maybe button A is the power button, okay.
Then it needs to be able to send the signal out to the device
itself which is the receiver here.
I think that's about it in terms of my design.
If it's something which is important then we just add more power
rather than having a thing that we don't have enough power.
So it's not really a constraint in that sense,
But I mean these are functionally, you know, the base, what the
technology has to do.
```

Figure 1.2: An action report for one of the meeting transcripts

To achieve this, the thesis was branched into two sub-tasks.

1.0.1 Task 1: Decision Detection

The first effort of the project is to research and design methods to identify decisions from a meeting transcript. Decisions may not be straight-forward as in "We are deciding to deploy this" or "I have decided to do this project". Many a times, for a topic discussed at the beginning of the meeting might reach a conclusion at a different point of time in the meeting. There may be other topics discussed in between or even contain discussions which might not be related to the agenda of the meeting at all. Navigating such situations is not easy even with the current state-of-the-art methods, which will be explained further in the next chapter.

1.0.2 Task 2: Dialogue Act Detection

Trying to identify only decisions seemed too specific and hence the resolution to take a step back and look at meetings from a wider point of view. Thus resulted the second effort of this project; to identify not just the decisions, but also the action items or ideas proposed during a meeting.

1.1 Research Question

The research question that has been explored is:

With recent technological advances in NLP, is it possible to identify Action related Dialogue Acts from meeting transcripts utilizing pre-trained language models to achieve better performances in comparison with present state-of-the-art methods?

1.2 Report Structure

This thesis is organized as follows. The background is presented in the Chapter 2. Basic concepts of Natural Language Processing tasks are also introduced along with general information regarding language models and how they can be used in transfer learning. Further, it also comprises of the previous work, which consists of work done on decision and dialogue act detection. In Chapter 3, the methods used to implement this thesis is covered. It includes the dataset corpus in consideration, models for decision detection and dialogue act detection with their configurations. Chapter 4 covers the results for all methods and configurations used. The successive Chapters 5 and 6 will consists of discussions and conclusions. Following these will be the references and appendices.

The thesis refers to several interesting blogs for further detailed knowledge, although the information from these blogs are not necessary to understand this thesis.

Chapter 2

Background

2.1 Basic Concepts

2.1.1 Similarity Measure

The similarity measure is the measure of how much alike (or dissimilar) two data objects are. In a data science context, it is the distance between objects considering the features as dimensions. There are several popular similarity measures such as Euclidean distance[1], Manhattan distance[2], Cosine similarity[3], Jaccard distance[4].

Cosine similarity is the most common measure used in NLP, which measures the *cosine* of the angle between two non-zero vectors, independent of their magnitudes. The cosine similarity between x and y is derived from the Euclidean dot product as:

$$CosSim(x, y) = \frac{x \cdot y}{||x|| \cdot ||y||} \qquad ;\epsilon[-1, 1] \tag{2.1}$$

The cosine distance is just:

$$CosDist(x, y) = 1 - CosSim(x, y) \qquad ; x, y > 0 \tag{2.2}$$

[1] https://en.wikipedia.org/wiki/Euclidean_distance
[2] https://en.wikipedia.org/wiki/Taxicab_geometry
[3] https://en.wikipedia.org/wiki/Cosine_similarity
[4] https://en.wikipedia.org/wiki/Jaccard_index

2.1.2 Frequency Measure

A frequency measure widely used in Information Retrieval, Summarization, text mining is tf-idf [47], which stands for term frequency - inverse document frequency. The intention of tf-idf is to represent the relevance of a term in a given document.

The intuition here is that if a word occurs several times in a document, it should have a higher relevance as it should be more meaningful than other words which appear occasionally (tf). On the other hand, if a word crops up profusely in multiple documents, it might be due to the fact that this word is just a frequent word; and not because it was relevant or meaningful (idf) [27].

Given a word w which appears in a document d from a set of documents D, the frequency measure (*tfidf*) can be represented as:

$$tfidf(w,d) = tf(w,d) \cdot idf(w) = \frac{N(w,d)}{N(d)} \cdot \left(1 + ln\Big(\frac{|D|}{|D*|}\Big)\right) \quad (2.3)$$

where,

- N(w,d): number of times w occurs in d
- N(d): number of words in d
- D: number of documents in the dataset
- D*: number of documents where w occurs at least once

The higher the tf-idf score, the higher the appearance of the term in a document but not in many other documents in the set. This term provides the most information about that specific document.

2.1.3 Tokenization, POS tag and NER

In natural language processing, tokenization is one of the essential elements to process text. Tokenization is the process of breaking up text to obtain smaller yet meaningful units from a given text. These units are referred to as tokens which can be words, phrases, symbols or other meaningful elements depending on the requirement. Based on the context, a suitable delimiter should

be preferred. For example, the dot is usually a sentence delimiter, but it is not the case when it is part of an acronym.

Parts-of-Speech (POS) and Named Entity Recognition (NER) are usually essential for solving more complex problems. POS tagging is the process of classifying a word to an appropriate POS tag (e.g., nouns, pronouns, verbs)[5], based on its definition and the context. Sometimes, this is not very straightforward as a particular word may have different Parts-of-Speech depending on the context it is used in.

Named Entity Recognition (NER) is a sub-task of information extraction which can be tackled in different ways. It aims to identify the named entities for a set of pre-defined categories (like name, location, organization, date)[6]. There are several libraries that have implemented state-of-the-art models with performance comparable to human standard.

2.1.4 Text preprocessing

In most of the cases, one can observe that the source data is not generally clean. Data derived from different sources may have different characteristics and that makes Text Preprocessing as one of the most important prerequisites in the machine learning pipeline. Text data from a novel would be completely different from that of Twitter. For example, consider the tweet, "The flight 2 lga i'm on was Cancelled Flighted". It can be seen that the tweet is not grammatically correct and also contains spelling errors. Hence, depending on the source and its quality, the text data would need to be treated accordingly. Some of the common preprocessing techniques are:

- **Stopword** removal (eg. a, an, the, and) from the source data, as these words do not provide any useful information.
- **Stemming** is a process where words are reduced to a root word by removing morphological and inflectional endings through dropping unnecessary characters, generally a suffix. For example, runs and running

[5]Most common parts of speech: noun, verb, adjective, adverb, pronoun, preposition, conjunction, interjection, article

[6]See appendix A for the complete list of NER types and correspondent descriptions used in this project

will both become run. The resulting words may not always be a real word (studies will become studi).

- **Lemmatization** is an alternative approach to stemming to removing inflection. Utilizing the part of speech and a lexical database, the word-forms are reduced to linguistically valid lemmas which are nothing but the dictionary form of the word.

- **Spelling correction** might be worth considering for conversational texts as there can be misspelled words or truncated words or even abruptly interrupted words.

- **Word embedding or Text vectors** are modern ways of representing words or text as vectors. Their aim is to redefine the high dimensional word features into low dimensional feature vectors. Put simply, it represents words as a vector of numbers (of something like hundreds to thousands dimensions) where related words are placed closer, for a given vocabulary of words.

- **Disfluencies** (eg. Hmm, mm, ah) are quite common in conversational text. They do not impart any meaning in the context of the conversation. Although, sometimes, they can be used as confirmation replies during a conversation (eg. mm-hmm).

- **Contractions** are shortened words that are written with an apostrophe (eg. "haven't" for "have not"). To standardize the source data, it makes sense to expand these contractions.

- **Capitalization** can reflect the beginning of sentences, proper nouns, abbreviations. For simplicity, the most common approach is to reduce everything to lower case. At the same time, it is also important to keep in mind that some words, like “US” to “us”, can result in different meanings when reduced. It may be redundant to to reduce in case NER is applied, as is takes care of labelling proper nouns, organizations etc.

2.1.5 Dialogue Act

In linguistics, a **Speech Act** can be denoted as an expression that not just presents information but performs some action as well. J. L. Austin[7], perhaps

[7]https://en.wikipedia.org/wiki/J._L._Austin

best known for his development of the theory of speech acts can be credited for the contemporary use of the term. He also introduced the concept of illocutionary acts. According to his framework, locution is something that was said and meant, illocution is what was done, and perlocution is something that happened as a result [3]. John Searle[8] in his book on speech acts [50], provides his account on illocutionary acts. He classified these acts into five groups such as assertives, directives, commissives, expressives and declarations [49].

Dialogue Acts can represent the meaning of an utterance in accordance to the illocutionary acts. They are similar to the speech acts given by Searle. Dialogue Acts can also be seen as a function that labels utterances based on a combination of pragmatic, semantic, and syntactic criterion. One of the most influential work in identifying dialogue acts was by Stolcke et al [52], who considered a taxonomy of 42 unique dialogue act labels to classify an utterance. They used the switchboard corpus [16] for labelling the utterances.

2.1.6 NLP open source libraries

Over the several years, many tools and libraries designed to solve NLP problems have become popular. In Python, the most commonly used ones are Natural Language Toolkit (NLTK) [57], SpaCy [22], Gensim [44] and Stanford's CoreNLP [31]. Each of these libraries have their own strengths.

NLTK[9] has over 50 corpora and lexicons, 9 stemmers, and dozens of algorithms to choose from. The Stanford CoreNLP is a Java library that can be quite easily integrated in a Python script, and supports some of the most spoken languages. Gensim is a specialized open source Python toolkit which is easy and efficient for vector space, topic modelling and document similarity analysis. SpaCy is a relatively newer Python NLP library that is tailored for performance. For each problem area, it just implements the current state-of-the-art algorithm.

This work has largely used SpaCy, e.g. for tokenization, POS[10] and NER[11] tagging as it is one of the newer libraries.

[8] https://en.wikipedia.org/wiki/John_Searle
[9] http://www.nltk.org/
[10] https://spacy.io/usage/linguistic-features#section-pos-tagging
[11] https://spacy.io/usage/linguistic-features#section-named-entities

2.2 Learning: Supervised and Unsupervised

Within the field of machine learning, data scientists use different kinds of algorithms to discover patterns in data which can lead to actionable insights. At a high level, depending on the manner they 'learn' using the data to make predictions, there are two main types of algorithms: **Supervised** and **Unsupervised** learning [34].

Supervised learning as the name suggests requires guidance in the form of labelled data to teach the algorithm on what outcome is expected. The objective is to adequately produce correct output data by identifying patterns or relationships in the source data. One prerequisite for supervised learning is the availability of correctly labelled data for training. This is typically done in the context of classification - when we want to map input to output labels, or regression - when we want to map input to a continuous output. Common algorithms include linear and logistic regression, support vector machines and artificial neural networks.

For example, consider a scenario in the real estate market, with data about the size of houses and their prices available, trying to predict the price of a new house. This is a regression problem as price is dependent on the size which is generally a continuous value.

Unsupervised learning, on the other hand, is when an algorithm can learn by itself to identify complex patterns and natural structures present within the data, without any guidance. Problems can be approached with little or no idea about what the results might look like. Although they tend to be complex, they empower novel ways to tackle problems that humans normally would not. In most cases, it is difficult to compare their performance as there are no labelled data provided. Some of the common tasks under unsupervised learning are clustering, principal and independent component analysis, and autoencoders.

Factors such as the amount of data, their structure and the specific scenario in hand influence the choice between supervised and unsupervised machine learning algorithm.

2.3 SVM

Support Vector Machine (SVM) is a supervised machine learning algorithm that can be used for both regression and classification tasks. Here, each available data item is plotted as a point in an n-dimensional space (where n is the number of features). Its objective is to find a hyperplane which separates the data points into two different classes [53]. There can be many possible hyperplanes that can separate such a set of data points, but the goal is to choose the best one. It translates to finding the plane which has the maximum margin, i.e. the maximum distance between data points of both classes. This strengthens the belief that future data points will be classified with better certainty.

SVM is designed uniquely, to find the best separating hyperplane between the data points. This is facilitated by searching for the closest points amid the classes of data points, which are referred to as the "Support Vectors". Hence the name Support Vector Machine as the best hyperplane "depends on" or is "supported by" the closest points from both the classes. Once these points are identified, the SVM draws a hyperplane connecting them by the virtue of vector subtraction, which is nothing but the margin. The best hyperplane is declared which bisects and is perpendicular to the connecting hyperplane. This can be visualized as shown in Figure 2.1, credit to Wikipedia[12].

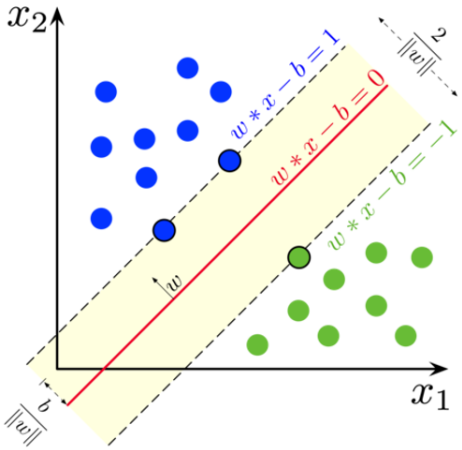

Figure 2.1: The red line represents the best hyperplane that separates two classes. Data points on the dotted lines are support vectors. The distance between them is the margin. Image taken from Wikipedia.

SVM works well on new data because a hyperplane which separates two

[12] https://en.wikipedia.org/wiki/Support-vector$_machine/media/File$: $SVM_margin.png$

classes in the best way is already identified. This hyperplane also keeps the points from two classes as far as possible. So, on the account of new data points, it is less likely that it trickles across into the other side.

Until now, we have seen examples of separable data points on a 2-dimensional space, hence the plane was linear in nature. In case the data points are not separable, it is not possible draw a hyperplane in 2-dimensional space. To overcome this, SVM has a technique called the Kernel Trick. These 'Kernels' transform the lower dimensional input space into a higher dimensional space, hence converting the non-separable task into separable task. These functions are generally used for non-linear separation problems. The simplicity of SVM is that it becomes a linear algorithm in high dimensional feature space and hence does not comprise of any complex calculations in that higher dimensional space.

2.4 Tree Based Learning

2.4.1 Decision Tree

Decision trees are a technique of splintering the data based on features to predict or classify some value. The data is split into groups depending on the number of branches the tree supports (two in case of a binary tree). A class or a predicted value is allocated to each leaf nodes of the tree. The label of the leaf node is the prediction for the data which is allocated to a particular leaf node.

Interpreting a decision tree is quite simple. Decision trees have structures similar to flowcharts. Here, each lower level node represents a 'test' on the feature of its parent node. Each branch depicts the effect of the test and each leaf node represents a class label. This label is impacted by all the previous decisions taken for each feature or attribute. Each path from the root node to a leaf node represents a classification rule.

Decision trees are flexible and explainable. The feature importance is clear and the relations can be easily viewed. Depending on the features to consider and the splitting conditions, the trees can be grown. Here, growing refers to the splitting of a node, based on a chosen feature and conditions for splitting. The splitting always begins from the root node in a top to bottom approach.

On the account of a split, each of the created child nodes corresponds to the applicable subset of the training data. Additional splits of these nodes based on a condition results in new nodes which correlate to even smaller sets of data, and so on. Leaf nodes are formed when further splits are not performed on a particular node. These leaf nodes in turn corresponds to classification decisions. For further comprehensive explanations, please refer to this tutorial[13].

A single decision tree might not generalize well as it is prone to overfitting. Trying to overcome it by means of limiting its depth for instance can result in an underfit decision tree. A simple approach to navigate through such situations are to use multiple trees together. Gradient boosting decision trees [14] [15] or shortly GBDT are one such mechanism of bringing together the predictions from multiple decision trees, by adding them, to make predictions which generalize well. The logic behind such a technique is that the subsequent predictors can learn from the mistakes of previous ones.

GBDTs are trained one tree at a time iteratively, the primary focus being to minimize a loss function. This is done by recursively splitting the data in a manner which maximizes some criterion is met. The selection of this criterion is made in a way such that the loss function is minimized at each split. The succeeding tree is then trained to minimize the loss function when its outputs are combined to the first tree. To achieve this, a new criterion can be utilized for splitting of this tree. There is no analytical or statistical explanation to decide the best split at each step, other than to try diverse set of splits and gauge the criterion for each split.

2.4.2 LGBM

There are several implementations of GBDTs and one among them is the LightGBM [28] or simply LGBM. They are fast, distributed, high performance gradient boosting framework that uses tree-based learning algorithms. They are used for ranking, classification and many other machine learning tasks.

LightGBM differs from other algorithms in the way which they grow. While other trees grow horizontally or level-wise, LGBM grows vertically or leaf-wise. A leaf node is chosen when the loss occurring in choosing it would be

[13] https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-understanding-decision-trees-adb2165ccab7

the minimum. When growing the same leaf, they can minimize the loss than a level-wise algorithm. Figures 2.2 and 2.3 display the growth of level-wise and leaf-wise trees respectively. Image credit to Microsoft blog[14].

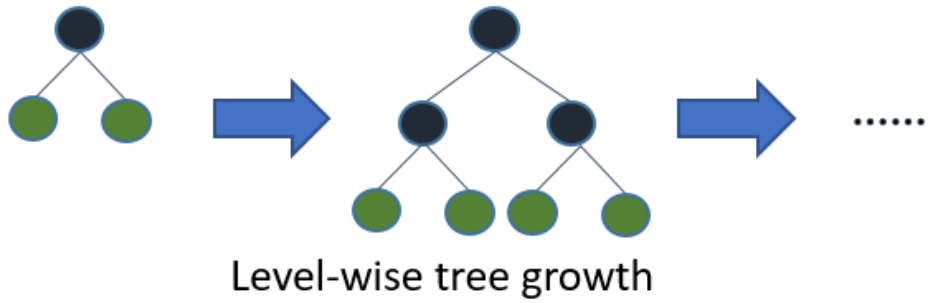

Figure 2.2

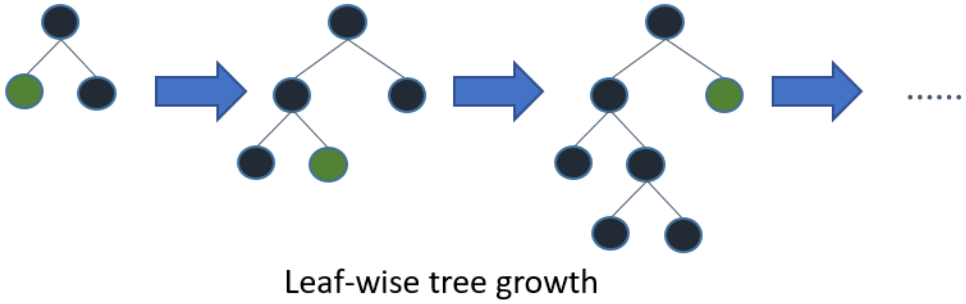

Figure 2.3

There are several reasons why LGBM are growing in popularity, and also why it is used in this thesis. The prefix 'Light' comes from the fact that they are very fast. As the size of data is ever increasing, traditional algorithms start to lag, but LGBMs handle them effortlessly, albeit utilizing lower memory for execution. The focus on accuracy of results is yet another reason for its popularity along with parallel processing capabilities and support for GPU learning.

[14]https://blogs.technet.microsoft.com/machinelearning/2017/07/25/lessons-learned-benchmarking-fast-machine-learning-algorithms/

2.5 Natural Language Processing

The field of Artificial Intelligence which concerns the processing and understanding of human languages by computers is described as Natural Language Processing (NLP). It is a combination of Natural Language Understanding (NLU) which is the ability of a computer or a system to consume and decipher what is expressed, interpret the context, ascertain applicable operation; and Natural Language Generation (NLG) which rejoinders in a user understandable language (by processing structured data into text). NLU is not limited to understanding the words, but also its meaning or even the context of its usage, possibly considering human errors, spelling mistakes or mispronunciations. NLG on the other hand, acts as a translator which represents machine data in a form that humans understand.

Let us consider a case when NLP is used on a sentence; the NLU algorithms tries to comprehend its meaning. In natural languages, it may be so that various words have the same meaning, or the same word(s) have alternate meanings, or even that the meaning can vary with the context. Contrarily, NLG scrutinizes the given data and provides a description in conversational human language.

Some common tasks under NLP are language detection or translation, sentiment analysis, topic modeling, information retrieval, spam detection etc. Recently, in many of the above tasks, researchers have been able to achieve or even surpass human set gold standard levels, machine translation being an outstanding example.

2.6 Language Representation

A machine understands a language by transforming textual data into numerical values and training mathematical models. Bengio et al. [5] coined the term word embeddings and trained a neural language model. Collobert and Weston [11] showed the applicability of pre-trained word embeddings using a neural network architecture; acting as a bedrock for several modern techniques. Word embeddings gained huge popularity with the development of word2vec by Mikolov et al. [33] which is a toolkit designed to easily train and use pre-trained embeddings.

Word embeddings, given their reputation and benefits, are one of the salient examples for supervised learning applications, or rather self-supervised learning. The labels for each word is already present, which are nothing but the succeeding words; eliminating the need for human annotation. One of the primary advantages is the option to skip the expensive annotations, which can be derived from a readily accessible large unannotated corpus. Pre-trained embeddings can also be used in subsequent tasks with the help of limited labeled data.

Vectorization is the process of transforming text to numbers by mapping words onto vectors of real numbers. A collection of such vectors of various words in the considered vocabulary is called as a vector space, which are a very effective method to understand language. Considering the case of word2vec, for a given vector space and large enough training data, it is easy to determine word relations with the help of vector addition or even measure the similarity between words by measuring the angles between the word vectors. Figure 2.4 shows such an example, image credit to Keitakurita's blog[15].

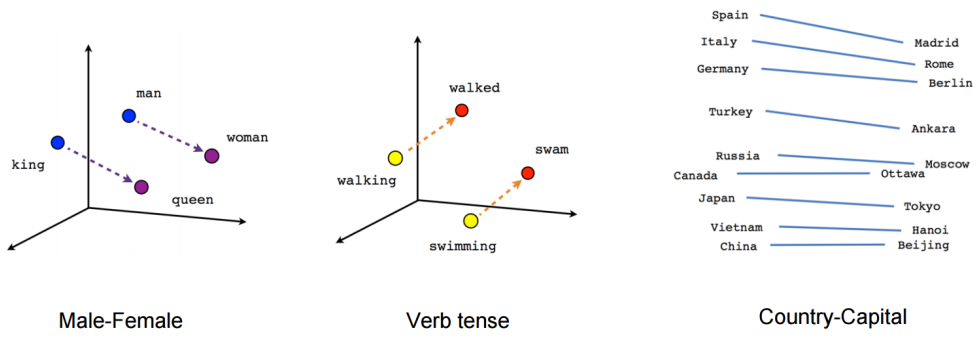

Figure 2.4: Researchers at Google were able to quantify word relationships using the word2vec embedding

Vectorization is not just limited to mapping each word to a vector; even characters or sentences or documents as well can be mapped to a vector. Vectorizing groups of words or sentences would be beneficial in discriminating between words with more than one semantic meaning. For example, "bank" can refer to a "bank account" or "bank of the river" or to "depend" upon. Vectorizing documents helps to analyze text at the document level. Here, document refers to an entity of text which is of interest to the task at hand. Topic modelling, text classification, spam email detection are some of its use cases.

[15]http://mlexplained.com/2018/04/29/paper-dissected-glove-global-vectors-for-word-representation-explained/

The essence of creating these vectors can be done by probing the context of word usage, or how often the set of words co-occur, or how frequent a particular word appears in every document. These techniques are reliant on the distributional hypothesis, often represented by the slogan put forth by J.R. Firth[16], “You shall know a word by the company it keeps”. Simply, the more semantically similar two words are, the more they will tend to show up in similar contexts and with similar distributions.

There are several embedding techniques ranging from the simplest like Bag of Words, to classic (or static) word embeddings like word2vec [33], GloVe [39], FastText [7], to modern dynamic word embeddings like ELMO [40] and BERT [12]. Classic or Static word embedding techniques, given a word, regardless of the context it appears will always have the same representation. On the contrary, dynamic word embedding techniques considers the context of the word, some of them using a language model to represent a word.

One limitation of word embedding models like Word2vec and FastText is that they are generally not very powerful as they are trained on very shallow language modeling tasks. This inhibits the capacity to capture the knowledge about the words, or combinations of words, or negation, etc.

Another fundamental drawback with using static word embeddings is their lack of ability to capture polysemy i.e. to identify the meaning of the word given the context. They generate a single word representation for all the words in the vocabulary. Consider the sentences "Let us crash the party" and "I was involved in a car crash", the word crash will be represented in the same manner for both the cases regardless of its meaning.

In order to overcome these limitations, deep language models were introduced for transfer learning. These models train a deep and sophisticated neural network to map each word to a vector based on the entire sequence of words or by considering the neighboring words for context.

Dynamic or contextual word embeddings capture word semantics in different contexts. Contextual embeddings can be of two kinds; unidirectional or bidirectional. In the sentence: "Humpty Dumpty sat on a wall", the unidirectional embedding for the word "sat" will only depend on "Humpty Dumpty" and not "on a wall", i.e. only the past seen data is considered for prediction.

[16]https://en.wikipedia.org/wiki/John_Rupert_Firth

Bidirectional embeddings such as ELMO [40] and Flair [2] train two representations for the word "sat", one left-to-right and one right-to-left, and then form a single representation by concatenating them. On the other hand, BERT will represent the word "sat" considering both its left and right contexts by a technique called as masking.

The basic idea is to train a complex, deep language model and utilize the representations it has learnt in downstream tasks with a possibility of further training or fine-tuning.

2.6.1 Recurrent Neural Networks

Artificial Neural Networks are information processing systems used in machine learning. As the 'neural' part of the name suggests, they are inspired by the biological nervous system and intended to learn the way that humans do [17], [19]. They are used for a variety of tasks, such as classification, character recognition, image compression, self driving cars to name a few.

An artificial neural network consists of neurons, an input layer, hidden layer(s) and an output layer as shown in Figure 2.5, image credit to Zanid's blog[17]. Neurons accepts an input value, perform a mathematical operation on it, and produce an output value. The number of hidden layers define the depth of a neural network.

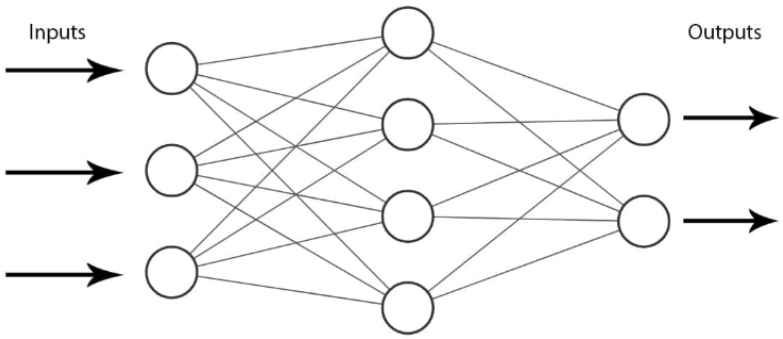

Figure 2.5: Feedforward neural network with one hidden layer in between an input and an output layer.

A variant of deep neural networks is the Recurrent Neural Network (RNN). RNN maintains an internal state (hidden state) which is updated as a function

[17]https://blog.zhaytam.com/2018/08/15/implement-neural-network-backpropagation/

of the current state and the next input symbol. The output symbols are generated as a function of the internal state. RNNs make this possible by introducing the notion of time into the system, enabling RNNs to process sequence of inputs, where the input tokens can be interpreted differently based on their surrounding tokens [18]. Natural language processing is one such example of processing sequential data.

Usually in language models, recurrent networks differ from feedforward networks in their architecture. They ingest their own past outputs again and again as new input, forming a loop over the network. This allows RNNs to utilize their internal state and perform tasks which feedforward networks cannot. These are called as 'long-term dependencies', meaning that an output token is influenced by an input token that appeared long before in the input sequence. Considering an example in natural language; in the sentence "The search for more tasty but still durable apples is ongoing", the use of the word 'is' (rather than 'are') is determined by the word 'search' which appeared eight words before. These kind of dependencies can span much bigger stretches of text than eight words.

RNNs can make use of such long-term dependencies when the gap between the relevant information and the place it is needed is small (i.e. number of words in case of text). As this gap grows, RNNs are unable to learn to connect the information [43]. In theory, RNNs are surely capable of handling very long-term dependencies. Sadly, in practice, RNNs do not appear to learn them as shown by Bengio et al [6].

Long Short-Term Memory or LSTM [21] networks, which are a special kind of RNNs, were designed to handle the problem of long-term dependencies. LSTMs although similar to RNNs, differ in the way they operate on the information. They make minor modifications to the information using multiplications and additions. This allows LSTMs to selectively remember or forget information over long periods. For further reading, refer to tutorials[18, 19].

[18]https://medium.com/technology-invention-and-more/everything-you-need-to-know-about-artificial-neural-networks-57fac18245a1

[19]https://colah.github.io/posts/2015-08-Understanding-LSTMs/

2.6.2 BERT

BERT, or Bidirectional Encoder Representations from Transformers [12], is a modern technique to train language models. It is a deeply bidirectional, self-supervised language representation, pre-trained using only a plain text corpus.

First, let us understand what a language model is. Language model, though it may seem complicated, is substantially just predicting words given a sequence of words or a context or to fill in the blank given a sentence. For instance, given the following context, "The _____ is in the jar" where _____ is the word which has to be predicted, a well-trained language model might suggest that the word "cookie" will have a probability of 60% and "jam" 25% to fill the blank etc.

Model Architecture

Language models have been ordinarily trained from left to right that is to predict what comes next given a sequence of words. Language models can be used for text generation; by predicting and appending one word at a time considering the previously generated sequence of words.

Different language models follow different architectures. Models such as ELMo [40] and ULMFit [23] are based on bidirectional LSTM [21]. For example, ELMo is trained using two sets of LSTMs; one for the standard forward (left to right) model and another backward (right to left) model. The combination of these two sets of LSTM enables the bidirectional properties of ELMo.

BERT on the other hand, randomly masks words in the given sentence and tries to predict the masked word. This is an effective way to increase the knowledge of the model by giving the context and asking it to predict the masked word which perfectly suits the sentence. This technique is one of the things that sets BERT apart from other Bi-LSTM models.

The bidirectional characteristic allows BERT to use the information from the whole sentence to predict a masked word disregarding its position in the sentence. Another fundamental difference is that instead of LSTM, BERT makes use of a potent state-of-the-art architecture called Transformer [56].

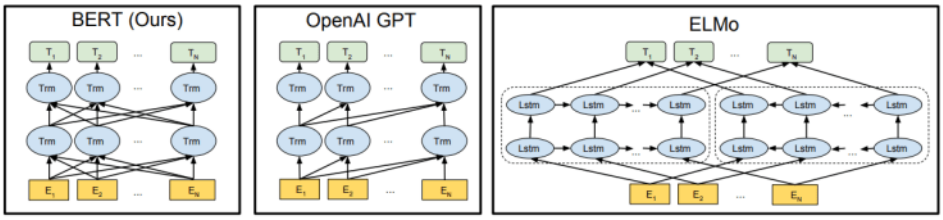

Figure 2.6: Comparing language model architectures for some transfer learning techniques. Image from original BERT paper.

Attention [10] is a concept which allows a model to attend to other words when concentrating on a particular word. While operating on an input word, attention is given to other words which may be important by having a large impact on the outcome of the operation. An attention vector is calculated and contains information about how much attention is to be paid for different words at each step. Self-attention is one of the many types of attention in which the focus is given only to the words from the same input sequence. This concept of self-attention is utilized in this thesis.

"Attention is All You Need" [56] first introduced the Multi-layer bidirectional Transformer aka the Transformer. It is a sequence to sequence model [54] which aims to map a fixed length input to a fixed length output, where the the input and output may be of different lengths. Most of the previous sequence to sequence models used RNN based architectures. A major drawback of RNN is that the input is processed sequentially by reading one input token at a time. This in turn results in a long training time, especially when dealing with huge amounts of data. Transformers on the other hand allow for parallelization during training, thus overcoming the above mentioned hurdle [55].

The Transformer in BERT adopts the attention techniques which enables it to learn the contextual relations between words in the given sequence and also the long term dependencies between them [37]. Similarly to LSTM, the Transformer stacks a layer which maps sequences to sequences, and hence when an input of "n" words is fed into the network, a sequence of "n" mathematical objects would be produced. These objects are called as tensors[20] and they generalize scalars, vectors and matrices to potentially larger orders. In depth details about attention and transformer can be found in this tutorial[21].

[20]https://www.tensorflow.org/guide/tensors
[21]https://mlexplained.com/2017/12/29/attention-is-all-you-need-explained/

The Transformer, in its vanilla form, consists of two separate mechanisms; an encoder that reads the input text and a decoder that generates a prediction for the task. Since the objective of BERT is to generate a language model, only the encoder mechanism is necessary. In contrast to directional models where the input text is either read forwards or backwards, the Transformer encoder reads the entire sequence of words straightaway. This represents the bidirectional reading, though some defend that it should instead be called as non-directional reading. This attribute makes it possible to learn the context of the word considering its neighbors.

The input schema during training as represented in figure 2.7 is made up of three different embeddings; token, segment and positional. The tokens are in the form of word-pieces (e.g. playing -> play + ##ing) instead of words. This is done to reduce the vocabulary size and also to increase the amount of data available for each word. Segment embeddings are used to distinguish between sentences as BERT can be trained on sentence pairs. This in turn enables for unique embeddings for each sentence which would be beneficial for tasks like question answering or natural language inferences. Positional embeddings are required to keep a track of the words in a sentence. This is due to the fact that, unlike RNNs, Transformer is unable to account the order of the input sequence i.e. it will treat all the tokens similarly as if they were the same word. The combination of all the above three embeddings are fed into the model.

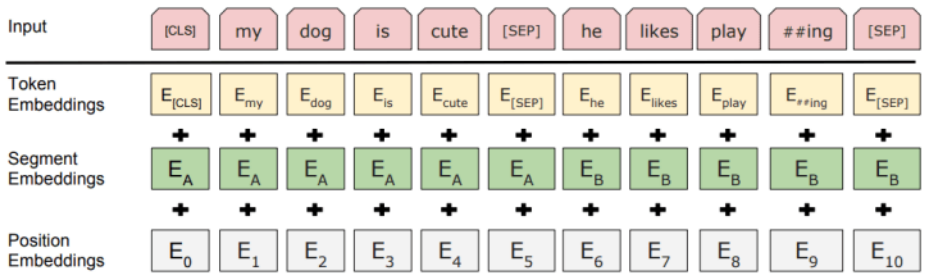

Figure 2.7: Input Schema for BERT. Image from original BERT paper.

2.6.3 Pre-Training

The BERT model was pre-trained on two functions; “masked language model” and “next sentence prediction”.

Language Model with Masking

The idea of masking a percentage of words for learning may seem modest. However, there are a few intricacies attached to it.

The masked language model, given a sequence of words, is trained in a way where the model learns to predict a randomly masked word within a sequence instead of the predicting next word to the given sequence. The simplest way would be to randomly replace a predefined percentage of words with a certain [MASK] token and ask the model to predict it. In most cases, BERT is trained by randomly setting 15% of the tokens in this manner. Although there might appear to be a flaw with this concept, the model tries to predict only when the [MASK] token is present in the input sequence. This would mean that there would not be any learning in the absence of the [MASK] token.

To overcome this sluggish behavior, not all tokens were masked in the same manner. To understand this process of masking, let us consider the same example the authors used in the paper: "My dog is hairy".

- Of the 15% randomly selected tokens, 80% of them were replaced with a simple [MASK] token.

  Example: “My dog is **[MASK]**”

- 10% of the tokens were replaced by a random token.

  Example: “My dog is **apple**”

- The remaining 10% of the tokens were left intact without any changes.

  Example: "My dog is **hairy**"

In the case where the model was trained only to predict the [MASK] tokens, and if it never saw this token during fine-tuning, there may have been chances of hindered performance as the model might have thought that there is no need to predict at all. The model would further be restricted in the sense that it would only learn a single contextual representation of the [MASK] token. With the requirement for it to predict a word at a position where there was

no [MASK] token, the model would have to learn different contextual representations of all the tokens from the sequence, in the event that it was asked to predict them later.

The authors biased the model by leaving some of the tokens intact so that the model would learn a proper and meaningful representation of the masked tokens. Ideally, this representation would be better than the random representation as well.

In comparison to a language model which would predict 100% of the tokens, BERT would converge more slowly as it only predicts 15% of the tokens. But, this is warranted by the fact that it had significant performance improvement in downstream tasks.

Next Sentence Prediction

In order to comprehend the relationship between two sentences, BERT complemented the masked language modelling with next sentence prediction. The task at hand consists on providing the model with two sentences and asking the model to predict if the second sentence follows the first in a corpus or not. The idea behind this is the ability for the model to learn to relate two distinct sentences to perform downstream tasks like natural language inference or question answering, since the masked language model would not capture such a relationship knowledge.

When considering multiple sentences as input, BERT introduces a special [SEP] token to distinguish between two sentences. A specific [CLS] token is also inserted at the beginning of the first sentence, for the facilitation of classification tasks, which is used only when classifying. During training, in half of the cases, the actual adjacent sentence was paired with the sentence in the corpus. For the rest 50% of the cases, randomly chosen sentences from the corpus was paired.

The entire input sequence i.e. both the sentences are now fed to the transformer and the probability that the second sentence follows the first is calculated with softmax. An example of a pre-training sequence presented in the paper is shown in figure 2.8.

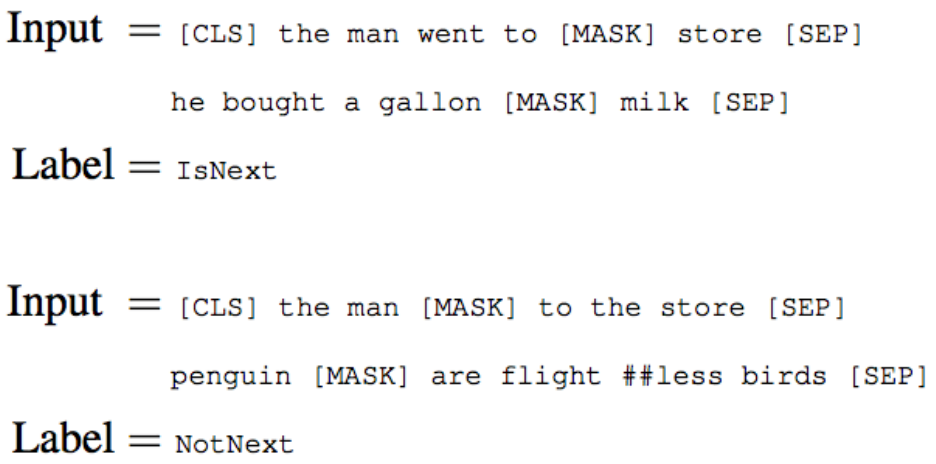

Figure 2.8: BERT Next Sentence Prediction. For the first case, the sentences are adjacent, so the label in [CLS] would be 'IsNext'. Image from original BERT paper.

2.6.4 Fine-Tuning

The fine-tuning procedure is quite straight-forward as the model, as described earlier, can be used in wide range of downstream tasks. For classification tasks, such as the case of this thesis work, the process is similar to next sentence prediction, by adding a classification layer on top of the BERT output for the [CLS] token. The goal here is to convert the input sequence to a single vector which represents the whole sequence so that it can be classified. to achieve this, the authors have opted a simple method; just consider the hidden state corresponding to the first token i.e. the [CLS] token.

Also, most of the hyper-parameters are unchanged in comparison with the training. The authors have provided specific guidelines in the paper and they have achieved state-of-the-art results or a wide array of natural language tasks, as demonstrated in Section 4 of the paper [12].

2.7 Text Summarization: Extractive and Abstractive

There are two fundamental approaches to text summarization: $extractive$ and $abstractive$. The former creates a summary based on strictly what is seen in the original text by extracting words and word phrases. It is comparable to copying down the important sections of the text without any alterations and

combining them to produce a grammatically acceptable summary. The latter tends to generate more human-like summaries, preserving the intent of the original text. They have the ability to produce words or phrases which are not present in the original text without wavering from the original context. These are harder to implement than extractive summarization in general.

2.8 Evaluation

In Machine Learning, there are several metrics available to gauge the performance of models or systems. In this thesis, the following metrics are used to evaluate the performance of the models developed:

- Accuracy
- Precision
- Recall
- F-score or F-measure[22]

The **Accuracy** of a model measures the ratio of correct classifications to the total predictions made. This calculation is shown in equation 2.4.

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions} \tag{2.4}$$

Prior to other metrics, let us first look at the confusion matrix[23] as described below. Confusion matrix is a performance measurement for machine learning classification problem where output can be two or more classes. It is extremely useful for measuring Precision, Recall, Accuracy, F-score, Specificity and other metrics.

		Actual Label	
		Positive	Negative
Predicted Label	Positive	*True Positive*	*False Positive*
	Negative	*False Negative*	*True Negative*

[22] https://en.wikipedia.org/wiki/F1_score
[23] https://en.wikipedia.org/wiki/Confusion_matrix

Where,
True Positives (TP) refers to the samples rightly classified as positive.
False Positives (FP) refers to the samples wrongly classified as positive.
False Negatives (FN) refers to the samples wrongly classified as negative.
True Negatives (TN) refers to the samples rightly classified as negative.

Precision (or positive predictive value) is a ratio between all adequately identified items and all identified items. It is calculated by dividing the *True Positive* with the total number of samples associated as positive, as shown in equation 2.5.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{2.5}$$

Recall (or sensitivity) is a ratio of all identified items and all existing items. It is calculated as the number of *True Positives* divided by the total number of positive samples as depicted in 2.6.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{2.6}$$

Simply put, precision says how many selected items are relevant, while recall says how many relevant items are selected. Precision and recall counter each other, that is, increasing one of them reduces the other. To get an optimal blend between the two, **F-score** (also F_1 score or F-measure) can be computed. F-score is the harmonic average of the precision and recall considering both metrics into account:

$$F - score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{2.7}$$

Accuracy may not be the best way to measure the performance of a system, especially in the case of an unbalanced data set. Here, even wrongly labelling the minute cases, will result in very high accuracy which does not make sense. Precision and Recall provide a way to tackle these scenarios. A better picture of the performance of a model can thus be seen by looking at **Precision**, **Recall**, **F-score** and **Accuracy** together.

2.9 Previous Work

In recent years, there have been several studies in the decision detection field of multi-party conversation. This is due to the fact that, a key aspect of meetings have been shown to be decisions. One of the most important aspects of a meeting are decisions as per user studies [4] [32]. Another study [59] discovered that for re-using meeting archives, a critical component is the automatic decision detection. Identifying decisions from meeting transcripts can form a base for applications which automates the process of producing minutes of meeting or a framework which can keep a track of all meetings and the report the flow of a project during its life cycle.

A study by Hsueh and Moore [24] attempted to automatically identify decision related dialogue acts from the meeting transcripts obtained from the AMI Meeting Corpus. Using manually created summaries, one was an extractive summary of the entire meeting and the other was an abstractive summary of all the decisions taken place, decision related dialogue acts were determined. Comparing the dialogue acts from the extractive summary which also aid any of the abstractive summary decisions were tagged as decision related. A Maximum Entropy classifier was trained to identify this dialogue act class utilizing features such as lexical, prosodic, dialogue act and conversational topic features. An indicator to the challenge at hand was the F-measure of 0.35 attained.

Fernandez et al. [13], unlike the previous study, made an effort to model the format of decision-making dialogue. An annotation design as shown in Figure 2.9 was conceived which took into consideration all the possible roles which different utterances played in the process of decision making. This scheme, for example, discriminates between decision dialogue acts which initiate a discussion by causing a topic / issue, to those which suggest a resolution, and to those which express agreement for a proposed solution and cause it to be accepted as a decision. A part of the AMI corpus was manually annotated with the above mentioned design, and then in what they refer to as a hierarchical classification approach, in the interest of identifying decisions and their dialogue act sequences. They achieved this by implementing one binary Support Vector Machine (SVM) per each hypothesized decision dialogue act class and a so called super-classifier (a further SVM) which is positioned on top of the earlier sub-classifier, to determine the sections of the discussions which incorporated decisions. This method produced an F-measure of 0.58 according to a lenient match metric, performing better than the kind of “flat classifica-

tion" approach using a single classifier pursued earlier by Hsueh and Moore.

key	DDA class	description
I	*issue*	utterances introducing the issue or topic under discussion
R	*resolution*	utterances containing the decision that is adopted
RP	– *proposal*	– utterances where the decision adopted is proposed
RR	– *restatement*	– utterances where the decision adopted is confirmed or restated
A	*agreement*	utterances explicitly signalling agreement with the decision made

Figure 2.9: Set of decision dialogue act (DDA) classes

Matthew Purver et al. [42] used a new technique to identify action items in conversational speech. First, they identified the sub-dialogues where the action items were proposed or discussed. Then, extracting word phrases that precisely encapsulate the task they involve. A structure as shown in Figure 2.10 was used to identify the sub-dialogues, using the ISCI dataset [25]. They used four independent classifiers for the detection of each individual action-item specific dialogue acts (AIDA) class and a super-classifier to identify them using multiple features. All the classifiers used were linear-kernel Support Vector Machines, using SVMlight [48]. They obtained an F-score of 0.51 for the detection of Action Items.

D	*description*	discussion of the task to be performed
T	*timeframe*	discussion of the required timeframe
O	*owner*	assignment of responsibility (to self or other)
A	*agreement*	explicit agreement or commitment

Figure 2.10: Action Item Dialogue Act(AIDA) classes

There were other studies very similar to the above mentioned ones. Trung H Bui et al. [8], Lu Wang and Claire Cardie [58], Matthew Purver et al. [41] and several others have tried to tackle the task. Their performances were lower than the ones described above and hence these methods are not discussed in detail.

Chapter 3

Methods

The scope of this project is to research and incorporate the latest techniques in NLP and comparing them with the current state-of-the-art techniques. The comparison may not be the case of comparing apples with apples. We are interested in gauging the recent advancements in NLP and hence we compare with the current state-of-the-art methods which not only considers transcripts but may also incorporate various prosodic features. This is due to the fact that NLP has evolved greatly during the last decade. Also, incorporating a system just on transcripts will make it easier to deploy as a software and does not depend heavily on the meeting recordings or the speech to text parts of the system.

3.1 Dataset

The Augmented Multi-Party Interactions (AMI) Meeting Corpus [9] contains a diverse set of multi-party meetings[1] with over 100 hours of content as both audio recordings and manual transcripts. Majority of the data is elicited using a scenario in which the participants play different roles in a design team. Others are a collection of naturally occurring meetings in a range of domains. In our experiments, the manually annotated transcript will be used instead of the transcription output from the Automatic Speech Recognition (ASR). A small subset of the available dataset will be used for evaluation as well.

[1] http://groups.inf.ed.ac.uk/ami/download/

3.2 Data Extraction and Description

The AMI Corpus, which is freely available, is constructed along with all annotations using NITE XML Toolkit[2] (NXT). Transcript files, decision files and the dialogue act files are stored in the XML format which can be accessed from reading the word files as shown in Figure 3.1. Here, all the words which are spoken in the meeting are stored in a row along with other information such as the speaker, start-time and end-time of the word utterance and also if the word is a punctuation or a laugh etc., and the actual word uttered. These are first converted to easily readable and accessible text files by reading the words for each speaker of a meeting, separating it into sentences (when a question mark or a period is found), combining all sentences of all speakers for a given meeting and finally sorting the sentences based on the start-time of the first utterance of the sentence. This will ultimately give us a clearly readable and understandable format which will later be used for further processing.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="true"?>
<nite:root xmlns:nite="http://nite.sourceforge.net/" nite:id="ES2002b.A.words">
    <w nite:id="ES2002b.A.words0" endtime="174.31" starttime="174.28">Mm</w>
    <w nite:id="ES2002b.A.words1" endtime="174.47" starttime="174.31">yeah</w>
    <w nite:id="ES2002b.A.words2" endtime="174.47" starttime="174.47" punc="true">.</w>
    <w nite:id="ES2002b.A.words3" endtime="203.46" starttime="202.95">Nope</w>
    <w nite:id="ES2002b.A.words4" endtime="203.46" starttime="203.46" punc="true">,</w>
    <w nite:id="ES2002b.A.words5" endtime="203.61" starttime="203.46">we're</w>
    <w nite:id="ES2002b.A.words6" endtime="203.73" starttime="203.61">all</w>
    <w nite:id="ES2002b.A.words7" endtime="203.98" starttime="203.73">set</w>
    <w nite:id="ES2002b.A.words8" endtime="203.98" starttime="203.98" punc="true">.</w>
    <w nite:id="ES2002b.A.words9" endtime="414.02" starttime="413.84">Which</w>
    <w nite:id="ES2002b.A.words10" endtime="414.25" starttime="414.02">which</w>
    <w nite:id="ES2002b.A.words11" endtime="414.41" starttime="414.25">is</w>
    <w nite:id="ES2002b.A.words12" endtime="414.51" starttime="414.41">the</w>
    <w nite:id="ES2002b.A.words13" endtime="414.84" starttime="414.51">clunky</w>
    <w nite:id="ES2002b.A.words14" endtime="415.06" starttime="414.84">one</w>
    <w nite:id="ES2002b.A.words15" endtime="415.06" starttime="415.06" punc="true">,</w>
```

Figure 3.1: Snippet of how the AMI corpus is stored. Each row can be interpreted as the meeting name, the speaker id, and the unique id for each word (all separated by a period), the relative timestamp of the utterance and the actual word uttered. Additionally, punctuations are marked distinctly.

Early diagnosis of the dataset led to a few interesting observations. Most of the meetings were synthetic, as in the speakers were given the transcript and told to read it out to record the meeting. This may not be a good indicator of general conversational meetings.

Another aspect was the structure of the dataset. Decision related statements are grouped together as part of a decision as shown in Figure 3.2. The decisions are labelled for relevant sentences, but for most of the meetings, it is quite unbalanced. There are a few meetings where out of hundreds of

[2]http://groups.inf.ed.ac.uk/nxt/

sentences only a few are decisions. With such a skewed dataset, it becomes difficult for models to learn from them.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="true"?>
<nite:root xmlns:nite="http://nite.sourceforge.net/" nite:id="ES2009a.decision">
  - <decision nite:id="ES2009a.decision.melissa.1">
        <nite:child href="ES2009a.A.words.xml#id(ES2009a.A.words1923)..id(ES2009a.A.words1961)"/>
        <nite:child href="ES2009a.B.words.xml#id(ES2009a.B.words243)..id(ES2009a.B.words244)"/>
        <nite:child href="ES2009a.A.words.xml#id(ES2009a.A.words1962)..id(ES2009a.A.words2010)"/>
        <nite:child href="ES2009a.D.words.xml#id(ES2009a.D.words539)..id(ES2009a.D.words541)"/>
    </decision>
  - <decision nite:id="ES2009a.decision.melissa.2">
        <nite:child href="ES2009a.C.words.xml#id(ES2009a.C.words188)..id(ES2009a.C.words210)"/>
        <nite:child href="ES2009a.A.words.xml#id(ES2009a.A.words2268)..id(ES2009a.A.words2273)"/>
        <nite:child href="ES2009a.C.words.xml#id(ES2009a.C.words211)..id(ES2009a.C.words221)"/>
        <nite:child href="ES2009a.B.words.xml#id(ES2009a.B.words245)..id(ES2009a.B.words249)"/>
        <nite:child href="ES2009a.A.words.xml#id(ES2009a.A.words2274)..id(ES2009a.A.words2278)"/>
        <nite:child href="ES2009a.D.words.xml#id(ES2009a.D.words555)..id(ES2009a.D.words566)"/>
        <nite:child href="ES2009a.B.words.xml#id(ES2009a.B.words250)..id(ES2009a.B.words261)"/>
        <nite:child href="ES2009a.D.words.xml#id(ES2009a.D.words567)..id(ES2009a.D.words568)"/>
    </decision>
```

Figure 3.2: Example of how Decisions are annotated in the AMI corpus. For each identified decision, their components can be fetched by extracting the words using the given details.

The AMI corpus has been annotated[3] based on a predefined set of guidelines[4] depending on the context and its usage. Table 3.1 shows the dialogue act schema and their frequencies, constructed based on the guidelines.

Group	Dialogue Act	Frequency	
Segmentation	1. Fragment	14,348	14.00%
	2. Backchannel	11,254	11.00%
	3. Stall	6,933	6.80%
Information	4. Inform	28,891	28.30%
	5. Elicit Inform	3,703	3.60%
Actions	6. Suggest	8,114	7.90%
	7. Offer	1,288	1.30%
	8. Elicit Offer or Suggestion	602	0.60%
Discussions	9. Assessment	19,020	18.60%
	10. Comment about Understanding	1,931	1.90%
	11. Elicit Assessment	1,942	1.90%
	12. Elicit Comment about Understanding	169	0.20%
Social	13. Be Positive	1,936	1.90%
	14. Be Negative	77	0.10%
Other	15. Other	1,993	2.00%
Total		102,198	100.00%

Table 3.1: AMI Dialogue Act Scheme and the DA distribution. The dialogue act type is preceded by the dialogue act id.

[3] http://groups.inf.ed.ac.uk/ami/corpus/annotation.shtml

[4] http://groups.inf.ed.ac.uk/ami/corpus/Guidelines/dialogue_acts_manual_1.0.pdf

This structure can create ambiguity for models. For example, the words like 'Yeah', 'Right' or 'Okay' are generally grouped under *Segmentation*. But, when the same words appear among decisions or actions, they are grouped under *Actions*. Even though this is completely natural, it can sometimes confuse a model during training. Also, while gauging the performance of the models, it can be noticed that many such words and sentences are identified under *Segmentation* group, but the actual label would be under *Actions* group.

Coming to the annotations of dialogue acts, word phrases are labelled as belonging to one of the fifteen dialogue acts. There are no multi labels, meaning that one word phrase (or sentence) is assigned to exactly one dialogue act. Figure 3.3 displays a snippet of how Dialogue Acts are annotated.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="true"?>
<nite:root xmlns:nite="http://nite.sourceforge.net/" nite:id="ES2002b.A.dialog-act">
  - <dact nite:id="ES2002b.A.dialog-act.dharshi.1">
        <nite:pointer role="da-aspect" href="da-types.xml#id(ami_da_1)"/>
        <nite:child href="ES2002b.A.words.xml#id(ES2002b.A.words0)..id(ES2002b.A.words2)"/>
    </dact>
  - <dact nite:id="ES2002b.A.dialog-act.dharshi.2">
        <nite:pointer role="da-aspect" href="da-types.xml#id(ami_da_4)"/>
        <nite:child href="ES2002b.A.words.xml#id(ES2002b.A.words3)..id(ES2002b.A.words8)"/>
    </dact>
  - <dact nite:id="ES2002b.A.dialog-act.dharshi.3">
        <nite:pointer role="da-aspect" href="da-types.xml#id(ami_da_5)"/>
        <nite:child href="ES2002b.A.words.xml#id(ES2002b.A.words9)..id(ES2002b.A.words24)"/>
    </dact>
  - <dact nite:id="ES2002b.A.dialog-act.dharshi.4">
        <nite:pointer role="da-aspect" href="da-types.xml#id(ami_da_9)"/>
        <nite:child href="ES2002b.A.words.xml#id(ES2002b.A.words25)..id(ES2002b.A.words26)"/>
    </dact>
  - <dact nite:id="ES2002b.A.dialog-act.dharshi.5">
        <nite:pointer role="da-aspect" href="da-types.xml#id(ami_da_3)"/>
        <nite:child href="ES2002b.A.words.xml#id(ES2002b.A.words27)"/>
    </dact>
```

Figure 3.3: Example of how Decisions are annotated in the AMI corpus. For each identified decision, their components can be fetched by extracting the words using the given details.

These data are used to extract the words, form sentences and label them as belonging to a dialogue act, which is here annotated in the format 'ami_da_' followed the the dialogue act number which is comparable to the dialogue act id in Table 3.1. This tells what kind of a dialogue act the word phrase represents. A dialogue act is marked with the word id's to denote the start and end of each act. Accessing the file with all the 'words' for the given range of word id's, a dialogue act can be extracted.

These extracted data are then stored in text files for easier access. Figure 3.4 shows a snippet of such an extracted file by combining the annotated information from both Figure 3.3 and Figure 3.1. Note that the pre-processing

step explained in the next section is already carried out to obtain this data.

```
.
A 174.28 yeah. ami_da_1
B 180.07 or yeah, just the idea, but I am not sure. ami_da_4
.
.
B 196.19 time for presentations then. ami_da_4
A 202.95 Nope, we are all set. ami_da_4
.
```

Figure 3.4: Extraction of dialogue acts into text file. The sentences for speaker 'A' can be compared with figure 3.3

3.3 Preprocessing

In most of the general cases, as in text from news articles or novels or even the commonly used Wikipedia data set, the sentences are mostly void of any spelling mistakes and are for the most part grammatically correct. But, in the case of multi-party meetings, disfluencies or half-spoken words or mispronounced words or wrong spelling (when translated) are common. In addition to these, there may be common occurrences like laughing, clapping or other sounds which are not of any use.

To achieve better performance, preprocessing the source data would be the first step. Here, all conversational disfluencies (like hmm, mh-hmm, ah etc.) are removed. An important point to note here is that some disfluencies such as uh-huh or mh-hmm can actually carry some meaning, and even be a part of a dialogue act or decision. Yet, all disfluencies are removed to evaluate the methods only on actual text and their context. Contracted words (eg. you're) will be expanded to the actual words (you are) to better represent the textual data. Using SpaCy library, the input data is tokenized and the lemma along with the Parts-of-Speech (POS) tags for each word is also obtained. Named Entity Recognition (NER) is also applied to obtain a more generalized data. This ensures that entity words do not get more weight in the context of the sentence. There are different preprocessing steps which have been used for evaluation of the system and these are be explained in detail in the coming chapter.

3.4 Classification

The project was divided into two separate tasks for the ease of understanding and better evaluation of different methods. The first task was to detect only the decisions and the second was to detect all Action related Dialogue Acts (Decisions, Ideas, Action Items). Also, for both the tasks, two different techniques (i.e. LGBM and Google BERT) was used for the detection and evaluation against each other, as well with the current state-of-the-art technique. The annotated data for both the tasks were extracted from the original AMI data corpus. One thing to note is that this thesis does not concentrate on segmenting the decisions or dialogue acts. Segmentation here refers to the identification of the start and end of decision or dialogue acts. The segmentation is just done by separating sentences at the end of each dialogue act, or on observing a question mark or a period. The sole focus of this thesis is to identify if a word phrase is a decision or a dialogue act based on the task.

3.4.1 LGBM

Light Gradient Boosting Machine (LGBM) [28] was chosen to classify the decisions and dialogue acts for the advantages they bring to the table, as explained earlier in section 2.4.2.

Hyper-parameters

LGMB models have numerous parameters which can be passed. The parameter tuning was done manually following the parameter tuning guide[5] and the previous experience of training LGBM. Thereafter, the parameters were varied and tested for the best result in terms of accuracy. The hyper-parameters which produced the best result were:

- n_estimators = 5000 (Number of boosted trees to fit)
- n_jobs = 4 (Number of parallel threads)
- learning_rate = 0.03
- num_leaves = 64 (Maximum tree leaves)

[5] https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html

- max_depth = 17 (Maximum tree depth)
- colsample_bytree = 0.5 (Subsample ratio of columns when constructing each tree)
- subsample = 0.9 (Subsample ratio of the training instance)
- min_split_gain = 0.01 (Minimum loss reduction required to make a further partition on a leaf node)
- min_child_weight = 3 (Minimum sum of instance weight needed in a child)
- early_stopping_rounds = 50 (Validation score needs to improve at least every early stopping round to continue training)

Encoding Input and Output

For the training of LGBM, the data from all the training files are read and pre-processing is performed as explained before. Tensorflow's Universal Sentence Encoder[6] is used to encode the sentences into high dimensional vectors to be used for classification. LGBM Classifier is trained with these vectors and the model is saved.

For the testing of model, the previously saved model is loaded and the test data is fed in a similar fashion to train except for the class labels. The performance of the model in terms of precision, recall, f-score and accuracy are displayed after the test is completed.

3.4.2 BERT

BERT authors have released two variants of pre-trained language models, $BERT_{LARGE}$ and $BERT_{BASE}$. These models differ in their architecture, as shown in table 3.2. The above two models can in turn be segregated in *Cased* and *Uncased* models. The difference here being that the cased models supports the capitalization of training and fine-tuning data while the uncased model converts all data to lower case prior to training and fine-tuning. This thesis makes use of the **$BERT_{BASE}$ Cased** model. The rationale behind choosing the

[6]https://tfhub.dev/google/universal-sentence-encoder-large/3

$BERT_{BASE}$ was the hardware capability limit, and *Cased* model for the preservation of context with regard to named entities such as names of person or places or other text.

Table 3.2: Architecture of different BERT models

Architecture	$BERT_{LARGE}$	$BERT_{BASE}$
Transformer Blocks	12	24
Hidden Layers	728	1024
Attention Heads	12	16
Parameters	110M	340M

Hyper-parameters

As with any deep learning model, hyper-parameter settings can make or break the results. For this purpose, the hyper-parameters were first set as similar as possible to the pre-trained model. Thereafter, experiments were run by changing all parameters one after the other and the results were compared. The ones that worked the best are shown here. Later, in the results section, the effects of varying hyper-parameters are depicted.

For fine-tuning, the following settings worked the best:

- Sequence length: 128
- Batch size: 32
- Number of epochs: 3
- Learning rate: 2e-5
- Number of classes: 2
- Data: AMI corpus [9]

BERT [12] was pre-training by the authors using the following hyper-parameters:

- Sequence length (single example): 256
- Batch size: 512

- Training steps: 1,000,000 (Approximately 40 epochs)
- Optimizer: Adam [30]
- Learning rate: 1e-4
- Learning rate schedule: Warm-up for 10,000 steps, then linear decay
- Dropout: 0.1 [51]
- Activation function: Gelu (Gaussian Error Linear Unit) [20]
- Data: BooksCorpus [60] and English Wikipedia [38], totally comprising of 800 million and 2500 million extracted words.

The BERT models were implemented by the authors of behind the original paper introducing BERT [12], using the TensorFlow machine learning library [1]. It is important to note that the pre-training took 4 days on 16 cloud Tensor Processing Units (TPU) (64 TPU chips), thus depicting the complexities involved.

Encoding Input and Output

For the purpose of fine-tuning of BERT, the data had to be encapsulated into a single Tab Separated Value (TSV) format file. The data from all the meeting transcripts marked for training was appended into a TSV file. Each meeting data was separated by a blank line for denoting the end of context of a meeting and also for easier understanding. The data was modified into the format accepted by BERT, i.e. the unique id for data, the class label, a placeholder and the actual data. The file used for testing will only have the first and last columns from the above mentioned columns.

In the case of input data, if the sequence length exceeded the parameter set for the same, the words occurring after the limit will just be ignored. In this case, since the sequence length was 128, any sentence where the number of words surpassed 128, would have been ignored. Although, when the data was examined, there were no sentences with such large number of words. This is because the sentences were split during pre-processing at the occurrence of question mark or period.

The resultant output file will contain as many columns as the number of classes considered during the fine-tuning and as many rows as the number of sentences passed for testing. The values of these rows and columns are the probabilities of the corresponding sentence belonging to that class. This was later combined with the testing data file to interpret the results. The identification of True Positive, False Positive, True Negative and False Negative sentences were performed to further examine the effectiveness of the models.

3.5 Decision Detection

Detecting decisions were not straight-forward, rather quite complicated especially with a very small and unbalanced dataset. LGBM performed well below that of dual layer SVM's which is the current state-of-the-art model. BERT was almost unable to identify decisions at all, identifying 99% of the sentences as not decisions. The results were extremely disappointing and further experiments on decision detection were halted.

3.6 Dialogue Act Detection

The focus of the experiment was shifted from just decisions to also include action items and ideas. This time around, the data set was more balanced and also bigger by couple of times compared to just decision data set.

With all set, the approach chosen was a top-down one to detect dialogue acts. Instead of the 15 classes of dialogue acts, the effort was put into the 6 groups of dialogue acts, as shown in Figure 3.3. At first, a multi-class classifier was designed considering the 6 groups of dialogue acts as each class. The performance of LGBM was still under the state-of-the-art model, while the BERT based model surpassed it.

To boil down even further, only 4 classes were considered. The classes were *Information*, *Actions*, *Discussions* and *Others*. The latter comprised of the *Segmentation*, *Social* and *Other* groups. The results were similar but the performance was better now. At this point of time, it was decided to cease further experimentation with LGBM as they were well below the state-of-the-art model while BERT was well above them.

With all the focus now on BERT based models, and the insights from the above models, it was decided to try with binary classifiers. There individual models were designed to identify Information, Actions and Discussions classes separately. For example, in the Information model, the sentences belonging to this group were labelled as important and rest all sentences from all other groups were labelled as not important. The performance of these models were significantly better than the multi class classifiers.

When the individual class models were examined, it was found that the Actions class contained all the necessary material required for our task. That is, it contained all the action items, decisions, ideas put forth and some more related aspects. With all the required data under a single class, it became easier to concentrate all the efforts on the binary classifier on the Actions model.

3.7 Post-processing

Once the decisions (task 1) and dialogue acts (task 2) are identified, minor post processing can help in optimizing the results even more. For the task at hand, different methods such as ignoring sentences with less than a specified number of words, or removing ambiguous sentences from the classified results can be performed. A point to note is that we have to be careful not to blur the boundary lines between the actual classification and post processing. The sentences which are added or removed should be compared and verified that they do not alter the scope of the classes in consideration.

Due to the conflicting labels of dialogue groups for certain word phrases, some of the sentences were labelled as not important by the model. When these were thoroughly examined, it made sense that these sentences were in fact not relevant to the action group of dialogue acts. And also, almost all of the sentences which consisted of 10 words or less were not useful in adding any value. So, after careful and comprehensive analysis, all sentences where 8 or less words were present in the *False Negative* category were moved to *True Negative* category. This in turn, increased the efficiency of the model without removing any valuable data.

3.8 Summarizing

Upon identifying the decisions or dialogue acts, summarizing them would be beneficial for multiple reasons. First, it is appropriate to get a small yet sufficient set of information in concern. If not, in case a meeting runs for hours together, the identified acts can amount to a considerably large set of sub-information. Secondly, in most cases, there is no need to get all the identified information, but just the important ones which can effectively sum up most of the meeting agenda just with decisions or dialogue acts.

With the above advantages, it is helpful to summarize all the identified items into a smaller report. Since the scope of the project was to research into the identification of decisions and dialogue acts, and not in summarizing them, we will not develop methods for the same. Instead, we will be utilizing already developed or available summarizer. For this project, we have used one such summarizer from Gensim[7].

[7] https://radimrehurek.com/gensim/summarization/summariser.html

Chapter 4

Results

The results of all the important models developed are presented in this chapter. They are segregated into three sections; Top performing models, Decisions and Dialogue Acts detection. Table 4.1 contains the acronyms used in this chapter.

Table 4.1: Acronyms used in this Chapter

Acronym	Expansion
SVM	Support Vector Machine
LGBM	Light Gradient Boosting Machine
BC	Binary Classifier
MCC	Multi Class Classifier
BERT	Bidirectional Encoder Representations from Transformers
PP	Post Processing

4.1 Top Performing Model

The current state-of-the-art models and few high performing models are shown in the table 4.2, with their respective F-scores. This table is presented to answer the research question of this thesis as it shows a comparison between current models and the models developed in this project. The highest performing models are also highlighted for easier understanding. A detailed discussion regarding all the models will be presented in the next chapter.

It is quite evident that the current state-of-the-art model which uses 2 layers of SVM classifiers to identify decisions performs the best for decision detec-

Table 4.2: Comparison between the top performing models

Model	F-score
Decision Detection	
SVM	**0.58**
LGBM: BC	0.31
BERT: BC	0.08
Dialogue Act Detection	
SVM	0.51
LGBM: MCC-6	0.41
BERT: MCC-6	0.56
LGBM: MCC-4	0.46
BERT: MCC-4	0.72
BERT: BC-Information	0.77
BERT: BC-Actions	0.76
BERT: BC-Discussions	0.70
BERT: BC-Actions + PP	**0.86**

tion. LGBM models are not best suited for the task at hand as they performance is below the current models. It is very easily evident that the BERT models, especially the single class identifier model coupled with very minimum post processing outperforms even the current state-of-the-art model be a huge margin.

4.2 Decision Detection

Table 4.3 displays the performance scores of the developed models. Both the classifiers are binary classifiers, trying to identify if a sentence was either a decision-making sentence or not.

Table 4.3: Decision Detection models

Model	Precision	Recall	F-score	Accuracy
LGBM: BC	0.24	0.75	0.31	0.53
BERT: BC	0.53	0.04	0.08	0.77

4.3 Dialogue Acts Detection

Table 4.4 depicts the multi class classifiers implemented and their performance. The premise behind choosing the six and four classes will be explained in the following Discussion chapter.

Table 4.4: Multi class classifier

Model	Precision	Recall	F-score	Accuracy
BERT: MCC-6	0.57	0.57	0.56	0.65
LGBM: MCC-6	0.29	0.68	0.41	0.48
BERT: MCC-4	0.73	0.72	0.72	0.66
LGBM: MCC-4	0.34	0.70	0.46	0.51

Table 4.5 contains the comparison for the individual classifiers for the important groups of dialogue acts.

Table 4.5: Individual binary classifiers for each group of dialogue act

Model	Precision	Recall	F-score	Accuracy
BERT: BC-Information	0.81	0.75	0.77	0.80
BERT: BC-**Actions**	**0.78**	**0.74**	**0.76**	**0.91**
BERT: BC-Discussions	0.69	0.73	0.70	0.77

Once the dialogue act group which was relevant for the thesis work was identified, the rest of the experimentation was focused only on this group. With a lot of variations and possibilities of testing done on the *Actions* group, only the best model is shown here.

Table 4.6: Top performing model: Individual classifier for Action class with post processing

Model	Precision	Recall	F-score	Accuracy
BERT: BC-Actions + PP	0.83	0.90	0.86	0.95

The other models which were not up to the mark will are described in the appendix B, as they give an idea on what kind of experiments were done and an insight on what can be built upon or avoided in the future.

4.4 Action Report

After identifying the best model, tests were run on it to classify the dialogue acts which were relevant for the research question. Figure 4.1 displays a snippet of correctly identified dialogue acts for the Actions group.

```
We should have a general idea of how it is going to look.
Should not we start with the most important parts?
but we should start with the power button?
Maybe we should centralise the discussion here.
We will draw two, and then we will see
But you just can put the the the the the whole interface a bit down,
Well let us talk about that later
Maybe you should another pen.
I think it should be at the button, bottom.
Well but what what if we first decide the different functions, and then look at the design.
yeah we should summon the the different aspects of the thing.
And moreover I think that you two should be come to consensus about the LCD s.
Alright, let is keep it central.
I think we should use something like this for the the channel up and channel down button?
Well let is look at your design.
as I already said, we could drop some of these buttons.
You should put that power button, channel and volume should have the most importance.
I think these should be in one big circle in the middle.
So it might be smarter to put them a little more away from each other.
So the the channel channel buttons should be far far apart, I think, up and down.
```

Figure 4.1: Snippet of rightly identified sentences for the Actions group

These identified sentences were then fed into a summarizer to get a concise action report, with a compression value of 0.25. This compresses the sentences to 25% of their original size. Figure 4.2 shows an action report for one of the meetings.

Not just going by the numbers from the results, but going through the sentences manually, one can see that most of the sentences are quite consistent under the Action group of the dialogue acts.

Well I will start just with another presentation, Well I think we shouldshow them before your presentations, And then we can discuss some more closely.
I think I have to start.
Thus it might be might be smart to make a a big zapping button or something in the middle, so you can reach it with your thumb.
It should actually be loose from the television, I think our user expert should also consider a manual for the remote, of course.
I think you should put more time in the in the design of pick up and use, than a manual.
But I think it is very important to make the power, channel and volume buttons near to the thumb, so you cannot have RSI consequences.
Maybe you can make, for channel changing, two little buttons on the side of the remote, so you can just do like this.
but the most important buttons maybe you can just put them a bit apart you can make a double feature like a button on the top and under it.
and that the less important buttons, like the different channels, the numbers one I think we should go further with the idea of a removable front.
we can use this board again, I think.
but we should start with the power button?
Well let is talk about that later Maybe you should another pen.
I think it should be at the button, bottom.
Well but what if we first decide the different functions, and then look at the design.
I think we should use something like this to The the channel up and channel down button?
Well let us look at your design.
You should put that power button, channel and volume should have the most importance.
So the channel buttons should be far apart, I think, up and down.
We we can make make a little row of like four buttons down here.

Figure 4.2: An action report for one of the meeting transcripts

Chapter 5

Discussion

This chapter presents a discussion on the experiments and the results. It starts with a general discussion about the results. Following is a discussion about the language model fine-tuning, text processing and practicality of results. Finally, a small dialogue about the Social and Ethical impact of the thesis.

5.1 General Discussion

At the start of this thesis, although there was a clear idea of the research question and a high expectation on the results, there were no pre-determined routes to achieve it. During the literature study, the approach of NLP features used were noticed, and the hope was to implement newer techniques and investigate on the task at hand. LGBM was chosen due to its aforesaid advantages and also due because it had worked well on a different NLP task at the host company where this thesis was carried out. Once LGBM was implemented and experimented with, it was pretty clear of its shortfall. Another iteration of literature study led to the shortlisting of using pre-trained language models, and BERT was the first choice considering its recent encounters of several NLP tasks.

Analyzing the results, it is very evident that BERT based language models surpass the current state-of-the-art models by a long shot, and that too with the limited dataset available for training. This came as no surprise, considering its results from the paper proposing BERT [12]. The multi class classifier performed well and the binary classifier eclipsed it with much ease.

Another focal point from the results is the adaptability of language models. The data or text on which the models were pre-trained on is completely different in structure to the text on which it was fine-tuned. With almost no correlation between them, the appeal towards them to be applied to various other NLP tasks grows. The flexibility of pre-trained language models complemented with their performance highlights their importance in NLP tasks and speaks for itself when it comes to its future potential.

5.2 Language Model Fine-tuning

Identification of dialogue acts from multi-party meetings has certainly benefited by the use of pre-trained language model, with evident results. The fine-tuning makes it possible for the language model to understand the meeting data and adapt the structure. With a large base knowledge, and the versatility to learn and revise to new data, there is no doubt about the capabilities of language models and its future.

5.3 Text pre-processing and post-processing

Looking at the results, one can judge that the data quality fed for training was quite good. Steps like removal of disfluencies, expanding contracted words have worked well. Although, when actions like feeding the speaker information, or PoS tags, or NER tags were tried, the efficiency of the model decreased. It can be speculated that too much pre-processing can remove valuable data which otherwise could help the model to better understand the data. Another reason may be that the BERT model has been very well pre-trained and there is no need for additional metadata for fine-tuning.

As the results indicate, the post processing step definitely helped the performance of the models. An important point to note here is that the post-processing done was not too huge that it affected the overall meaning of classification tasks. Additionally, this procedure was done only after meticulous inspection and understanding of the results.

5.4 Practical Usefulness

As seen earlier, the results look good on paper. Here is a small discussion on the practical usefulness of these results. A highly efficient method to correctly identify all the action related dialogue acts may not suffice alone. An extractor which can determine the gist of all such identified acts may also be equally necessary. Without a proper summarizer, the action reports will just be a collection of sentences and may not make sense grammatically and also make it hard to read. If not compressed enough, the report might be misleading by including action related dialogue acts which are not relevant to the context of the meeting.

5.5 Social and Ethical Impact

Going by the results, it would be amazing to see the model deployed in the real world. With such efficiency, and the ability to iteratively learn, there is no doubt that sooner than later such models will be deployed. This directly benefits the companies by providing a concise report, saving lot of time and money to the alternative of doing it manually, and provide a way to track the progress of projects from an eagle's view. It can also create new jobs of analyzing these data to improve the effectiveness of meetings. On the other hand, there may be less jobs offered to people (for example, the secretary positions). This can be overcome by training the current workforce with new and updated skill set.

Such a system, when developed, should be carefully deployed, especially considering the privacy aspect for collecting new data and re-tuning the model. Additional security can ensure data protection form hackers.

Chapter 6

Conclusions

This chapter presents the conclusion of this thesis work. It also consists a section discussing probable approaches which can be investigated in future research.

6.1 Conclusions

This thesis was an attempt to research into the identification of action items, decisions and other dialogue acts from meeting transcripts. In particular, to use the latest technological advances in NLP, considering only the textual features and not the prosodic features or other audio features. The application of newly proposed, pre-trained language models showed that transfer learning can be used to outperform traditional learning by leaps and bounds. The work wonders even in the case when there is a limited data to be trained (fine tuned) upon. **Bidirectional Encoder Representations from Transformers** or simply BERT is the language model which has been utilized in this thesis, along with a traditional decision tree-based **Light Gradient Boosting Machine** or Light GBM.

The LGBM model was not able to outperform the current state-of-the-art models which was built using a two-layer SVM classifier. But then, BERT was able to easily and enormously surpass the current state-of-the-art model. It should also be noted that the data available was quite limited and also not the cleanest. This thesis also shows that BERT based models are multidimensional in the case of usage. That is, they can be exercised on a task that uses data which can be completely different to what it was pre-trained on. In this

case, the data used to pre-train the BERT model was a collection of books and Wikipedia articles and applied to identifying all dialogue acts from meeting transcripts. The fine-tuning task was comparatively faster and quite efficient as seen from the results. It shows the capacity of the BERT model to adapt to different kinds of fine-tuning tasks, with improved results. This demonstrates the potential of pre-trained language models like BERT and the multitude of tasks it can be applied in the field of NLP.

6.2 Future Work

Even after achieving such a good performance, there is virtually no limit to possible future improvements. If given a chance again, considering the knowledge and insights acquired over the last six months, there are some interesting possibilities which can further be investigated.

First and foremost would be the application of $\text{BERT}_{\text{LARGE}}$ model on the same task of identifying dialogue acts task, if better hardware is available. It is more likely that the usage of the large model would improve the performance of developed models under consideration, even though the improvement may not be much. This assumption is made on the fact of comparisons the authors have made for the small and the large models [12].

Secondly, either using a different and bigger data set or to create one by recording meetings and converting speech to text to obtain the transcripts. The latter was actually tried as an additional part of this thesis, but the speech to text performance was not good enough even with the usage of good quality microphones combined with Google and Amazon speech to text API's since it was tried on meetings were both English and Swedish languages were spoken. Application of the model only on English meetings can be a good start.

Another possible pathway can be to pair this research with a more powerful summarizer. Utilizing an abstractive summarizer would be beneficial as conversational meetings may sometimes have grammatically incorrect sentences. It can provide a concise and easily readable report.

Additional avenue to research can be the incorporation of BERT models in other fields of NLP which have not yet been delved into. Pairing BERT with other advanced or complex architectures can lead to better performance.

Bibliography

[1] Martın Abadi et al. “Tensorflow: Large-scale machine learning on heterogeneous distributed systems”. In: *arXiv preprint arXiv:1603.04467* (2016).

[2] Alan Akbik, Duncan Blythe, and Roland Vollgraf. “Contextual string embeddings for sequence labeling”. In: *Proceedings of the 27th International Conference on Computational Linguistics*. 2018, pp. 1638–1649.

[3] John Langshaw Austin. *How to do things with words*. Oxford university press, 1975.

[4] Satanjeev Banerjee, Carolyn Penstein Rosé, and Alexander I. Rudnicky. “The Necessity of a Meeting Recording and Playback System, and the Benefit of Topic-Level Annotations to Meeting Browsing”. In: *INTERACT*. 2005.

[5] Yoshua Bengio et al. “A neural probabilistic language model”. In: *Journal of machine learning research* 3.Feb (2003), pp. 1137–1155.

[6] Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[7] Piotr Bojanowski et al. “Enriching word vectors with subword information”. In: *Transactions of the Association for Computational Linguistics* 5 (2017), pp. 135–146.

[8] Trung H Bui et al. “Extracting decisions from multi-party dialogue using directed graphical models and semantic similarity”. In: *Proceedings of the SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics. 2009, pp. 235–243.

[9] Jean Carletta et al. “The AMI meeting corpus: A pre-announcement”. In: *International workshop on machine learning for multimodal interaction*. Springer. 2005, pp. 28–39.

[10] Jianpeng Cheng, Li Dong, and Mirella Lapata. “Long short-term memory-networks for machine reading”. In: *arXiv preprint arXiv:1601.06733* (2016).

[11] Ronan Collobert and Jason Weston. “A unified architecture for natural language processing: Deep neural networks with multitask learning”. In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 160–167.

[12] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[13] Raquel Fernández et al. “Modelling and detecting decisions in multi-party dialogue”. In: *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*. Association for Computational Linguistics. 2008, pp. 156–163.

[14] Jerome H Friedman. “Greedy function approximation: a gradient boosting machine”. In: *Annals of statistics* (2001), pp. 1189–1232.

[15] Jerome H Friedman. “Stochastic gradient boosting”. In: *Computational statistics & data analysis* 38.4 (2002), pp. 367–378.

[16] John J Godfrey, Edward C Holliman, and Jane McDaniel. “SWITCHBOARD: Telephone speech corpus for research and development”. In: *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 1. IEEE. 1992, pp. 517–520.

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[18] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. “Speech recognition with deep recurrent neural networks”. In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, pp. 6645–6649.

[19] Kevin Gurney. *An introduction to neural networks*. CRC press, 2014.

[20] Dan Hendrycks and Kevin Gimpel. “Gaussian error linear units (gelus)”. In: *arXiv preprint arXiv:1606.08415* (2016).

[21] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[22] Matthew Honnibal and Ines Montani. “spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing”. In: *To appear* (2017).

[23] Jeremy Howard and Sebastian Ruder. “Universal language model fine-tuning for text classification”. In: *arXiv preprint arXiv:1801.06146* (2018).

[24] Pei-yun Hsueh and Johanna D. Moore. “Automatic Decision Detection in Meeting Speech”. In: *MLMI*. 2007.

[25] Adam Janin et al. “The ICSI meeting corpus”. In: *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*. Vol. 1. IEEE. 2003, pp. I–I.

[26] Johanna Jones. “The Impact of Counterproductive Meeting Behaviors on Meeting Effectiveness, as moderated by Meeting Attendee Personality”. In: (2016).

[27] Karen Spärck Jones. “A statistical interpretation of term specificity and its application in retrieval”. In: *Journal of documentation* (2004).

[28] Guolin Ke et al. “Lightgbm: A highly efficient gradient boosting decision tree”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 3146–3154.

[29] Been Kim and Cynthia Rudin. “Learning about meetings”. In: *Data mining and knowledge discovery* 28.5-6 (2014), pp. 1134–1157.

[30] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[31] Christopher Manning et al. “The Stanford CoreNLP natural language processing toolkit”. In: *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 2014, pp. 55–60.

[32] Agnes Lisowska Masson, Andrei Popescu-Belis, and Susan J Armstrong. “User Query Analysis for the Specification and Evaluation of a Dialogue Processing and Retrieval System”. In: *LREC*. 2004.

[33] Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).

[34] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[35] Vincenzo Pallotta, John Niekrasz, and Matthew Purver. “Collaborative and argumentative models of meeting discussions”. In: *In Proceeding of CMNA-05 international workshop on Computational Models of Natural Arguments (part of IJCAI 2005*. 2005.

[36] Vincenzo Pallotta, Violeta Seretan, and Marita Ailomaa. “User requirements analysis for meeting information retrieval based on query elicitation”. In: (2007).

[37] *Paper Dissected: “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding” Explained*. https://mlexplained.com/2019/01/07/paper-dissected-bert-pre-training-of-deep-bidirectional-transformers-for-language-understanding-explained/. Accessed: 2010-09-15.

[38] Jeff Pasternack and Dan Roth. “The Wikipedia Corpus”. In: 2011.

[39] Jeffrey Pennington, Richard Socher, and Christopher Manning. “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[40] Matthew E Peters et al. “Deep contextualized word representations”. In: *arXiv preprint arXiv:1802.05365* (2018).

[41] Matthew Purver, Patrick Ehlen, and John Niekrasz. “Detecting action items in multi-party meetings: Annotation and initial experiments”. In: *International Workshop on Machine Learning for Multimodal Interaction*. Springer. 2006, pp. 200–211.

[42] Matthew Purver et al. “Detecting and Summarizing Action Items in Multi-Party Dialogue”. In: *In Proc. of the 9th SIGdial Workshop on Discourse and Dialogue*. 2007.

[43] *Recurrent Neural Networks and LSTM explained*. https://medium.com/@purnasaigudikandula/recurrent-neural-networks-and-lstm-explained-7f51c7f6bbb9. Accessed: 2010-09-15.

[44] Radim Rehurek and Petr Sojka. “Software framework for topic modelling with large corpora”. In: *In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Citeseer. 2010.

[45] Rutger Rienks, Dirk Heylen, and Erik van der Weijden. “Argument Diagramming of Meeting Conversations”. In: 2005.

[46] Nicholas C Romano and Jay F Nunamaker. “Meeting analysis: Findings from research and practice”. In: *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*. IEEE. 2001, 13–pp.

[47] Gerard Salton, Anita Wong, and Chung-Shu Yang. “A vector space model for automatic indexing”. In: *Communications of the ACM* 18.11 (1975), pp. 613–620.

[48] B Scholkopf. “Making large scale SVM learning practical”. In: *Advances in Kernel Methods: Support Vector Learning* (1999), pp. 41–56.

[49] John R Searle. “A taxonomy of illocutionary acts”. In: (1975).

[50] John R Searle and John Rogers Searle. *Speech acts: An essay in the philosophy of language*. Vol. 626. Cambridge university press, 1969.

[51] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[52] Andreas Stolcke et al. “Dialogue act modeling for automatic tagging and recognition of conversational speech”. In: *Computational linguistics* 26.3 (2000), pp. 339–373.

[53] *Support Vector Machine — Simply Explained*. `https://towardsdatascience.com/support-vector-machine-simply-explained-fee28eba5496`. Accessed: 2010-09-15.

[54] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.

[55] *Transformer Architecture: Attention Is All You Need*. `https://medium.com/@adityathiruvengadam/transformer-architecture-attention-is-all-you-need-aeccd9f50d09`. Accessed: 2010-09-15.

[56] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[57] Wiebke Wagner. “Steven bird, ewan klein and edward loper: Natural language processing with python, analyzing text with the natural language toolkit”. In: *Language Resources and Evaluation* 44.4 (2010), pp. 421–424.

[58] Lu Wang and Claire Cardie. “Focused meeting summarization via unsupervised relation extraction”. In: *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics. 2012, pp. 304–313.

[59] Steve Whittaker, Rachel Laban, and Simon Tucker. “Analysing Meeting Records: An Ethnographic Study and Technological Implications”. In: *MLMI*. 2005.

[60] Yukun Zhu et al. “Aligning books and movies: Towards story-like visual explanations by watching movies and reading books”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 19–27.

Appendix A

SpaCy NER Types and Descriptions

The built-in entity types in the models trained on the *OntoNotes 5*[1] corpus support the following entity types:

TYPE	DESCRIPTION
PERSON	People, including fictional.
NORP	Nationalities or religious or political groups.
FAC	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states.
LOC	Non-GPE locations, mountain ranges, bodies of water.
PRODUCT	Objects, vehicles, foods, etc. (Not services.)
EVENT	Named hurricanes, battles, wars, sports events, etc.
WORK_OF_ART	Titles of books, songs, etc.
LAW	Named documents made into laws.
LANGUAGE	Any named language.
DATE	Absolute or relative dates or periods.
TIME	Times smaller than a day.
PERCENT	Percentage, including "%".
MONEY	Monetary values, including unit.
QUANTITY	Measurements, as of weight or distance.
ORDINAL	"first", "second", etc.
CARDINAL	Numerals that do not fall under another type.

Figure A.1: List of supported entity types and descriptions in SpaCy

[1] https://catalog.ldc.upenn.edu/ldc2013t19

Appendix B

Supplementary Results

A diversified amount of variations were tried on the Actions model. Some of the interesting ones are presented below.

B.1 Test 1: Varying Learning Rate

Different learning rates such as 2e-5, 3e-5, 5-e5 and 10e-5 were tested with. The performance was found to be best in 2e-5, but the other learning rates produce very similar performances and so all other experiments were run using 2e-5 as the learning rate.

B.2 Test 2: Varying Epochs

The models were trained with 3, 5 and 10 epochs. The 3 and 5 epoch models had very similar performance while the model trained on 10 epochs was fractionally lagging behind. The difference in models were so minute in terms of performance, but in terms of time taken to train, was almost twice. The model with 3 epochs were chosen due to the advantage of time over very minute performance improvement over the other.

B.3 Test 3: Parts of Speech

The next model trained was input with parts of speech (PoS) tags side-by-side to each word of a sentence. The PoS tags were derived from the SpaCy

library. The trained model performed minutely poor than the regular model. Hence, PoS tags were not used in further experimentation.

B.4 Test 4: Named Entity Recognition

Subsequent model was now trained with the Named Entity Recognition(NER) which were obtained from SpaCy again. The named entities were replaced with the pre-defined entities from SpaCy. This model was better than the PoS model but still lagged behind the regular model. So, NER tags were not used anymore.

B.5 Test 5: Meeting by Meeting Training

Until now, all the training data were combined into a single file and fed into BERT. A question was raised, whether BERT could identify different meetings and their context or was there a need to segregate the data and induce context only for that particular meeting transcript. This led to an incremental fine-tuning of BERT meeting by meeting. The resultant model was one of the worst performing models. Almost all of the sentences were labelled as not an Action group dialogue act.

This shows that BERT can in fact identify the context between meetings and there is no need of separately training it.

B.6 Test 6: Markings for each meeting

Two methods of separating each meeting in the training file were conceptualized. One was to simply have a blank data line after each meeting. The other was to explicitly mark the start and end of meetings. This was done by adding two lines, one before and one after all the sentences of a particular meeting.

While both the methods improved the efficiency very minutely, only the former was used for the regular model. The improved performance from the latter method was due to these markers also being considered for evaluation. When it was removed from evaluation, the result remained the same as the

regular model.

B.7 Test 7: Speaker Info

Adding the speaker info to each sentence was attempted, with the hope that it might provide more value to the context of the meeting. First, the speaker names were tried as just alphabets like 'A', 'B' and so on. Later, it was tried by giving garbage values like 'ajp5bf' so that these will not influence the context directly, but only act as a marker for the sentences. Although, both the methods did not effect the efficiency of the model at all. The speaker information was removed hereon.

B.8 Test 8: Ensemble Model

When working with the multi-class classifier, there was a thought of training individual models and then to ensemble them to get a well trained model. But, this resulted in a very bad model where it classified almost all the sentences as not important.

Table B.1: A comparison of all supplementary models with the Best Performing Model

Model	Precision	Recall	F-score	Accuracy
Test: 1	0.78	0.74	0.76	0.91
Test: 2	0.78	0.74	0.76	0.91
Test: 3	0.80	0.68	0.72	0.91
Test: 4	0.77	0.74	0.75	0.91
Test: 5	0.86	0.50	0.48	0.88
Test: 6	0.78	0.73	0.76	0.91
Test: 7	0.77	0.75	0.76	0.86
Test: 8	0.57	0.57	0.48	0.49
Best Model	**0.83**	**0.90**	**0.86**	**0.95**

TRITA EECS-EX