11-11-2022 2:00 PM

Predictive Modelling For Topic Handling Of Natural Language Dialogue With Virtual Agents

Lareina Milambiling, *The University of Western Ontario*

Supervisor: Katchabaw, Michael, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Science degree in Computer Science

Abstract

In this thesis, we aim to contribute to ongoing research in the field of human-computer dialogue and help move closer to the goal of having more realistic human-computer dialogue. We address the current challenge of topic handling in human-computer conversation by proposing a Topic Handler model that is designed in such a way that is flexible and compatible with third party dialogue systems. This model builds off of previously proposed dialogue grammars and systems and is based on speech act theory and conversation analysis. By employing feature engineering of existing dialogue act corpora and using this data in machine learning experimentation, we successfully demonstrate that not only does speech act and semantic annotation data improve the performance of classifiers for the task of identifying appropriate points of topic change, we also demonstrate how the proposed Topic Handler model can provide needed inputs to assist in topic handling when used in parallel with dialogue systems.

keywords: human-computer dialogue, topic handling, dialogue systems, dialogue act, speech act theory, conversation analysis, non-player characters, virtual agent, chatbot, artificial intelligence

Summary for Lay Audience

In this thesis, we aim to contribute to ongoing research in the field of human-computer dialogue and help move closer to the goal of having more realistic human-computer dialogue. We address the current challenge of topic handling in human-computer conversation by proposing a Topic Handler model that is designed in such a way that is flexible and compatible with third party dialogue systems. This model builds off of previously proposed dialogue grammars and systems and is based on linguistic methods and theories (e.g. speech act theory and conversation analysis). By leveraging pre-annotated corpora and further creating additional input features for machine learning experimentation, we successfully demonstrate that not only does speech act and semantic annotation data improve the performance of machine learning models for the task of identifying appropriate points of topic change, we also demonstrate how the proposed Topic Handler model can provide needed inputs to assist in topic handling when used in parallel with dialogue systems.

Acknowledgment

I wish to show my immense appreciation to my supervisor, Dr. Michael Katchabaw, for his infallible guidance and patience throughout the years. Without his support, my entry into the field of computer science and the completion of this thesis would never have been possible.

I would also like to offer special thanks to Dr. Helmut Jürgensen for his invaluable input on early drafts of this work, prior to his passing. I will be ever grateful for his counsel.

Contents

List of Figures

List of Tables

1 Introduction

A current challenge in the field of natural language processing (NLP) is the development of artificial intelligence (AI) that can engage in dialogue with a human at a seemingly human level, or rather, developing an AI that has the ability to engage in dialogue using 'non-standardized spoken language'. The increased interest in AI that can partake in dialogue with a human originated from a rise in online chatting as a medium of communication [12]. This interest has further increased with the expansion of e-commerce over the last decade and the demand for customer service on digital platforms (e.g. the increase in e-commerce business use of chatbots to handle inquiries 24/7) [39]. Online chatting originally gained popularity because it is non-invasive, unlike a phone call, and at the same time it is inherently an instant medium of exchange [12]. The author in [12] points out that communicators using online chatting are not compelled to respond instantly, but can carry on a conversation in near instant time if they desire. The chat logs can also be stored and easily referenced in future points and the medium is easily available to anyone with access to an internet enabled computer or mobile devices. The inherently informal nature of the medium lowers inhibitions, encourages spontaneity, and generally reduces the barriers to effective communication [12]. This has ultimately led to increased research in chatbot development, an enterprise that leverages the medium of online chatting, and has resulted in an increase in the number of existing chatbot platforms, architectures, and chatbot implementations available [39].

In [6], the use of 'non-standardized spoken language' means the human user of an interface is not restricted to communicate with the computer by predetermined commands or keywords; instead the human user may be able to communicate with the computer using his or her own natural (spoken) language expressions, subject of course to the limitations imposed by the topic of 'conversation'. There exists AI that can simulate conversation with another human (e.g. ELIZA, non-player characters (NPCs) in video games, chatbots in various messaging platforms, virtual assistants such as Google's Assistant or Siri), but they are not currently capable of cohesive conversations or proper topic handling at a human level, especially when digesting unplanned natural languages or expressions. Cohesive topic handling would allow an AI or virtual agent to recognize when best to continue the current topic of conversation and identify appropriate points in conversation to start a new topic. Often the AI's responses fall flat as they do not address the current or new topic in a relevant manner or miss opportunities to introduce new topics [12]. For example, virtual assistants (otherwise known as smart home devices) are advanced chatbots that act as personal virtual assistants [39]. They help users accomplish daily tasks, such as booking hotels, getting the latest news or even turning off/on the lights, through multiple communication channels such as voice control on multiple devices [39]. A persisting drawback however is that current virtual assistants still have a closed knowledge domain, as they are expected to carry out certain specified and preset tasks [39]. It is generally presumed that human-computer conversation systems are typically passive and that humans should take the role to lead the conversation and introduce new content when a stalemate occurs, and the virtual agent only needs to "respond"[29]. In the future, more improved chatbots should be able to converse openly about a variety of topics [39] as well as proactively propose new content when appropriate [29] and in a seemingly natural human like manner. This ultimately leads to the question: How would an AI be able to take into account a human user's input utterance (either verbally or textually) and generate an appropriate response, in machine readable form, given the current topic and conversation context?

1.1 Research Question

While the above question is one that would require much more extensive research, this thesis focuses on a certain aspect that has not yet been achieved. A struggle for current models and implementations of virtual agents, chatbots and Non-Player Characters (NPCs) is topic handling in conversation (i.e. identifying a change in topic, handling a new topic, identifying opportunities to introduce a new topic, engaging in longer and more cohesive conversation, etc). This is of particular interest for when engaging with NPCs as any lapse in human-like responses impacts the immersive aspect of a virtual environment. Other areas of research relating to dialogues, such as speech act theory and conversation analysis, also investigate the nature and intent of natural human dialogue and could potentially offer insight into how to improve topic handling. Therefore, in order to contribute to the ongoing research to improve this issue, this thesis will explore the following research question: Can speech act recognition and patterns aid in solving current topic limitations in human-computer dialogue? Or more specifically:

Can speech act patterns in a dialogue help an NPC better identify when proposing a new topic versus continuing the current topic is appropriate?

1.2 Motivation for Research

There are many underlying steps to achieving the goal of having more realistic human-computer dialogue. In simplest terms, the AI must be able to go from the utterance to what is being said, then generate an appropriate response. The achievement of this process could mean advantages in countless NLP and AI subfields, for example more precise and efficient chatbots and virtual agents, more immersive video games with more realistic dialogue exchange between player and NPCs, and improvement in automated summarization and information retrieval. With the popularity of chatbots, virtual assistants and dialogue capable NPCs growing, research in this has only increased. This thesis aims to contribute to ongoing research in this field, aiding in more natural topic flow in human-computer dialogue by incorporating other areas of research such as speech act theory and conversation analysis.

1.3 Proposed Method

The purpose of this thesis is to create a method for topic handling by leveraging existing speech act annotation methods, as well as developments in machine learning and NLP, in order to demonstrate that speech act annotations can be used for identifying 'natural' points of topic change.

By using speech act theory and conversation analysis as a basis, then by building off of past proposed human-NPC conversation models and dialogue grammar models, this thesis proposes a Topic Handler framework for human-computer dialogue. The Topic Handler model will be outlined within the scope of game environments and as a module to be used within the context of existing and future dialogue systems (for example, the dialogue system proposed in [36]).

The Speech Act Annotated Corpus (SPAAC) offers a corpora of data for testing the ability to identify points of topic change within a human-human conversation using speech act patterns. We apply machine learning methods here to illustrate how identified patterns in this corpus can be used to predict when it is appropriate for a virtual agent to continue a current topic or propose a new topic and thereby aims to promote natural flow of conversation in human-computer dialogue.

1.4 Research Contributions

The contributions of the thesis are as follows:

1. A Topic Handler model for human-computer dialogue leveraging linguistic methods and theories. The proposed framework is designed in such a way that is flexible and compatible with third party dialogue systems. In particular, we demonstrate the Topic Handler model's compatibility with the work in [36] which proposes a comprehensive model for conversation handling.

2. A proof of concept for main modules in framework by implementing:

   a) Feature engineering of dialogue act and semantic related data for input in machine learning module

   b) Machine learning experimentation to demonstrate the positive impact of using speech act and semantic annotation data on identifying appropriate locations for an NPC/virtual agent to propose a new topic

   c) Machine learning experimentation to show potential architecture for effective implementation of proposed Topic Handler model

To our knowledge this is the only framework that leverages linguistic methods and theories for topic handling in human-computer dialogue systems. Our experimentation shows that the framework is particularly well suited to task-oriented dialogues, as this is the genre of the corpora used, however we believe it to be equally applicable to other genres as well.

1.5 Structure of Thesis

In the next Chapter, we first discuss speech act theory, existing corpora with speech act annotations and motivations behind the creation of such corpora. We then look at the field of conversation analysis, research in dialogues and dialogue modelling, and then discuss how this is applicable to this thesis' endeavour.

In Chapter 3 we highlight the history of human-computer dialogue, current methods in human-computer dialogue development, and delve into more detail about current limitations of human-computer dialogue development. This chapter then discusses overall challenges in NLP and AI, the current limitations of existing chatbots and NPC, and how this thesis aims to contribute to overcoming one of these limitations.

In Chapter 4 we discuss two previously proposed dialogue/conversation models that include modules designed to handle topics and how these models can help overcome the topic handling limitation of current chatbots and NPCs.

In Chapter 5 we propose a framework for Topic Handling. We then outline each module within this thesis's proposed Topic Handler model and highlight how each module interacts in order to identify the best next action in terms of topic continuation versus topic change. We then revisit the Conversation Handler model proposed by the author in [36] and describe how this work's propose model could be integrated.

In Chapter 6 we outline the SPAAC dialogues and additional feature engineering required in order to implement the Topic Handler model simulation and experiments in Chapter 7.

In Chapter 7 we apply the Topic Handler model to sample dialogues and evaluate as proof of concept. We begin by outlining the subset of SPAAC data used for testing and explain how the use of speech act and related semantico-pragmatic data can improve dialogue topic coherence when used as input into various machine learning (ML) methodologies. The general testing approach is outlined in this Chapter. Results of initial tests are also discussed in this Chapter.

Following the demonstration of how speech act and semantic annotations can be used to predict best topic handling in Chapter 7, Chapter 8 discusses what was learned, main contributions and proposes interesting possibilities for future research.

2 Speech Act Theory, Conversation Analysis, and Dialogue Grammars, Models and Systems

In many dialogue models, a manner in which to identify when a topic needs to be addressed further and knowing if it is relevant is not expressed or explicitly outlined. For our Topic Handler model, we will use speech act patterning to identify when continuation of current topic or introduction of a new topic is appropriate in human-computer dialogue. The following discussion outlines speech act theory and conversation analysis fields of research, the basis of the Topic Handler model to be proposed in Chapter 5.

2.1 Speech Act Theory

Speech acts were introduced by J.L. Austin in 1962 [11, 20]. Austin introduced the notion of utterances as actions which change the state of the environment, dialogue participants, etc (in contrast to the previous notion of utterances as expressions which can be evaluated to true or false) [20]. Austin distinguished three types of actions performed by utterances:

1. **locutionary**: action of saying something (shaping the utterance, pronouncing, and using it to refer to real world objects) [20]; refers to the actual utterance and its intended meaning, where the utterance is meant to be taken completely literally, with no consideration for context or conventions [12].

2. **perlocutionary**: action performed by saying something (achieved effects) special to the particular situation e.g. persuading or surprising) [20]; refers to the utterance and its possible unintended consequences or effects, i.e. the utterance causes fear, anxiety, some rational or irrational action, etc.

3. **illocutionary**: action performed in saying something (e.g. informing, requesting, asking, answering, warning, apologizing, etc) [20]; refers to the utterance and its real intended meaning, i.e. what utterance actually means in a particular context as defined by some convention of social linguistic usage [12].

Of the three levels above, the illocutionary speech act is the most semantically meaningful as applicable to research in human to virtual agent dialogue [12] and this type has been the most worked on in subsequent research [20]. The literature on illocutionary speech acts broadly recognizes the following four high-level taxonomies [12]:

1. **Austin**: Expositives, Exercitives, Verdictives, Commisives, Behabitives.
2. **Searle**: Representatives, Directives, Commissives, Expressives, Declarations, Representative Declarations.
3. **D'andrade and Wish**: Expositives, Interrogatives, Exervitives, Reactions, Verdictives, Commissives, Behavitives.
4. **VerbMobil**: Request, Suggest, Convention, Inform, Feedback.

Searle extended Austin's work on illocutionary utterances and attempts to define necessary conditions for the act to be performed. These were present as game definition rules:

- conditions of normal input/output (necessary for one to express himself and others to understand)
- propositional content (content restrictions)
- environment
- sincerity (alignment of actual attitudes with the ones expressed in the utterance) [20]

Searle also presented several dimensions along which the speech acts can vary, and proposed the above speech act taxonomy based on the dimensions [20]. The taxonomy in more detail is:

1. **representatives**: which commit the speaker to the truth of the expressed proposition (paradigm cases: asserting, concluding, etc.)
2. **directives**: which are attempts by the speaker to get the addressee to do something (paradigm cases: requesting, questions)
3. **commissives**: which commit the speaker to some future course of action (paradigm cases: promising, threatening, offering)

4. **expressives**: which express a psychological state (paradigm cases: thanking, apologizing, welcoming, congratulating)

5. **declarations**: which effect immediate changes in the institutional state of affairs and which tend to rely on elaborate extra-linguistic institutions (paradigm cases: excommunicating, declaring war, christening, firing from employment) [20]

With this taxonomy, assuming speech acts have been recognized accurately, we can pair or group utterances based on the speech act and determine if the utterance is discussing and continuing the current topic or is introducing a new topic.

2.2 Dialogue Act Annotated Corpora and Dialogue Systems

As illustrated above, the speech act is an utterance with a general performative function and can range from five to ten functions, however a dialogue act is a specialized speech act that is defined only by the context of specific dialogue system [12]. Research in automated annotation of dialogue acts has stemmed from development of spoken dialogue systems, and the typical task domains include the retrieval of information form database or making reservations, such as airline information [34]. Such systems are designed for specific contexts, for example DARPA Communicator was specifically for airline information; ARISE and MASK were created to retrieve train information [34]. Majority of these studies assumed a definite and consistent user objective, and the dialogue strategy was usually designed to minimize the cost of information access [34]. Other tasks included tutoring and trouble-shooting dialogues, in these dialogue scenarios the agendas are typically described by using a dynamic tree structure and the objective is to satisfy previously outlined requirements [34].

2.2.1 Switchboard DAMSL Annotation

All the most commonly and universally defined dialogue acts have been collected in the modified Switchboard - Dialogue Act Markup in Several Layers (SWBD-DAMSL) tag set [12]. The SWBD-DAMSL tag set is a comprehensive collection of all speech acts defined in the literature and contain forty-two speech acts [12]. In this dialogue act taxonomy, the dialogues are typically annotated on four different levels (communicative, information, forward-looking function, backward-looking function). It is important to note that some utterances might lack a tag on some levels; for instance an utterance can have the backward-looking function but the forward-looking function is missing [20].

Such a taxonomy and annotations have already been leveraged in existing chatbot models. For example, the proposed chatbot model in [12] uses pre-outlined conversation types and certain speech acts in order to decide the next action. In [12], the author uses the Searle's above taxonomy in Section 2.1 (with recognition of assertives in lieu of representatives) in order to model conversation types after the conversation type has been decided. The author in [12] also uses a Goal-Fulfillment Speech Act to acknowledge the utterance causing the state of the conversation to reach a conclusion (i.e. when all issues raised in the conversation have been resolved and acknowledged by the participants of the conversation) [12].

We will revisit chatbot developed in [12] in Section 3.3.2 and discuss the model and its incorporation of speech acts further detail. In the section, we look at another example dialogue system that leverages dialogue act annotations and patterns in order to offer consulting systems for specific contexts.

2.2.2 Use Dialogue Act Annotations to Construct Dialogue Systems for Consulting

In [34], the authors introduce a spoken dialogue corpus for developing consulting dialogue systems, systems that help the user in making a decision (e.g user/-customer consulting with sales clerk while shopping or consulting with the the concierge staff at a hotel). In [34], a dialogue act annotation scheme is designed to describe two aspects of a dialogue act: speech act and semantic content. They annotate dialogue acts to describe user's intention and a system's action and consider both speech act and semantic content to be important information to handle consulting dialogue. The speech act tag set is used to capture the communicative functions of an utterance using domain-independent multiple function layers and the semantic content tag set is used to describe the semantic contents of an utterance using domain-specific hierarchical semantic classes [34].

In [34], the authors point out that consulting is a frequently used and very natural form of human interaction. Consulting forms part of a series of information retrieval dialogues and contains various exchanges, such as clarifications and explanations. The dialogue system in [34] senses the user's preferences from his or her utterances, provides some information, and then request a decision. The team in [34] also highlight that it is almost impossible to handcraft a scenario that can handle such spontaneous consulting dialogues, therefore the dialogue strategy should be boot strapped from an extensive dialogue corpus. If an extensive dialogue corpus is available, they can utilize machine learning techniques such as partially observable Markov decision processes and weighted finite-state transducers [34]. Such techniques would require sufficiently rich in information corpora to describe the consulting dialogue to construct the statistical dialogue management system [34].

As preliminary analysis, the speech act tag set is evaluated (with over thirty different speech act tags) in terms of the agreement between labellers and the patterns of tag occurrences is investigated in order to carefully understand the structure of the decision-making process [34]. By segment examples and assessing relative occurrence rate of different tags at different point in a 'decision-making process' episode, compared to occurrence throughout the dialogues, they were able to identify patterns and characterize dialogue flow within the context of sightseeing planning in Kyoto (e.g. guide and use first clarify the latter's interest (open-question, wh-questions), final part of episode had higher occurence of y/n-question, accept, and elaboration) [34]. Following this, the authors in [34] construct automatic taggers for speech act and semantic content tags by using the annotated corpora and machine learning techniques, and uses condensation or selection of dialogue acts that directly affect the dialogue flow in order to construct a consulting dialogue system using the dialogue act tags as an input.

From this example, we see how dialogue acts and additional semantic content patterning can help our understanding of dialogue structure and how best to serve human speakers with the information they are looking for. In the following section we look at an automated dialogue annotation tool and the sample output annotated corpus this thesis uses in the proof of concept experiments for the proposedTopic Handler model.

2.2.3 SPAADIA and Dialogue Annotation and Research Tool (DART)

The Dialogue Annotation and Research Tool (DART) is an annotation taxonomy and analysis tool created to enrich dialogue data largely automatically with pragmatic-relevant annotation on a number of different levels, thereby taking the potential for genuine corpus-based approaches to the field of pragmatics one step further [50]. The distinct levels covered by DART 2.0 comprise of syntax (both traditional and extended 'sentence' types), semantic ('topics'), semantico-pragmatics ('illocutionary force indicating devices'; IFIDS for short), surface polarity, and pragmatics (speech acts) [50]. In DART 1.0, there were fifty-seven speech acts, some of which could occur in combination, of course exceeding the taxonomies established by Austin and Searle, and even [34]. DART 2.0 supports even more fine-grained basic taxonomy of more than one hundred and twenty basic categories and their potential combinations, distinguishing between different types of speech acts as realized through and in different C-unit types, the sequencing of units in dialogue, the influence of modality, polarity, and features a more robust grammar for recognizing different syntactic types, larger inventory of IFIDs and an improved inferencing mechanism for deducing speech acts, all based on symbolic, rather than probabilistic identification strategies [50]. The second version of the SPAADIA (Speech Act Annotated Dialogues) corpus was re-annotated using the DART v2 taxonomy.

The SPAADIA corpus is a subset of data annotated as part of the SPAAC project [48]. The original speech-act annotation scheme was applied to British Telecom OASIS Corpus of twelve hundred telephone dialogues, and to the Trainline Corpus of thirty-five longer telephone dialogues for the Speech Act Annotated Corpus project [48]. The DART v2 taxonomy was then applied to the thirty-five Trainline dialogues in order to create SPAADIA Corpus version two [47]. The same version two taxonomy was also applied to fifty-five SRI Amex Travel Agent dialogues [47], and going forward when we refer to SPAADIA in this thesis we are referring to both the Trainline and SRI Amex dialogues that were annotated with the DART v2 scheme.

Unlike the taxonomies and methods outlined in the previous section, DART taxonomy is applied to annotate syntactic, semantic, and pragmatic data at the C-unit (also called ‘moves’ or ‘dialogue acts’ as used above) level rather than the utterance level. C-units are the basic units (or ‘envelopes’) for conveying speech acts, which can be regarded as the minimal communicative actions performed in a dialogue. C-unit is the basic unit for speech act annotation [28]. Syntactically, it is an independent clausal or a non-clausal unit (e.g. a non-clausal unit is here labelled as a ‘fragment’). Functionally, it represents a unit which can be assigned to a given communicative function, represented by its speech act attribute.

An utterance can contain more than one C-unit and each C-unit is assigned speech-act attributes and other attributes [28]. An utterance cannot contain less than a C-unit, and if an utterance contains more than one C-unit, then one of the C-units will be a discourse marker. Therefore the structure of an utterance is a C-unit plus one or more discourse markers associated with it [28].

Utterances are not given separate attributes, and so present annotation scheme pays attention only to C-units [28]. We will revisit C-unit classifications within the DART v2 taxonomy in Chapter 6 as in our Topic Handler model proposal we use annotated dialogues from SPAADIA for the model's proof of concept. Going forward, when we discuss a dialogue act (DA) in our proposed Topic Handler model, feature engineering and experimentation (Chapters 6 to 8) we are referring to C-units as defined here.

In [50], it is maintained that the DART annotations make it possible to carry out further investigation into the form-function relationship embodied in and expressed through, the different levels, potentially leading to far deep in-sights into the mechanisms that underlie different communicative strategies.

2.2.4 Speech Act and Dialogue Act Summary

As mentioned above, this thesis proposes a Topic Handler model that will use speech act theory to map topic patterns and identify when a new topic is introduced, when a topic discussion is potentially ending, when a topic can be continued, and when a new topic can be introduced. We have seen in the previous sectioned how speech act theory is already being leveraged in building dialogue systems, in recent research efforts to automate dialogue act annotation and in chatbot development. Before we delve into the proposed Topic Handler model, or continue to discuss other existing and proposed models and chatbots, it is also important to also discuss conversation analysis research in order to understand patterns and regularities of natural language dialogues.

2.3 Conversation Analysis

Conversation Analysis (CA) is a markedly 'data centred' form of discourse analysis: in the purest sense, the analyst is not supposed to appeal to any evidence that comes from outside the talk itself [11]. The fact that talking is prototypically a joint enterprise involving more than one person, and that people normally take turns at talk, is central to the CA approach which is concerned above all with describing sequential patterns which are observable in the data being analyzed (i.e. regularities in what follows what) [11]. Claims about what is going on in any particular 'bit of talk' is often supported by 'going to the next turn' or referring to the evidence of what happens in the following 'bit of talk'. While speech act theorists may define an utterance as 'question' in terms of whether it meets the felicity conditions for the relevant illocutionary act, conversation analysis practitioners would be more interested in whether the utterance is followed by an answer. Conversation analysis practitioners point out that the participants display their understanding of the previous contribution as a ‘question’ and their ‘orientation’ to the general principle by answering the question because in properly ordered talk, questions will be followed by answers [11].

Some conversation analysis practitioners apply the principle that one should not look beyond the data very strictly. For them it is not just unnecessary but illegitimate for an analyst to make use of information that the participants themselves have not chosen to ‘make relevant’ in their ongoing interaction [11]. This position reflects the origins of conversation analysis in a particular school of sociological thought: ethnomethodology. Ethnomethodologists reject the view that social order is created by abstract structures, but instead are created by the concrete actions of people going about their everyday business [11]. It follows that sociologist should concentrate on studying people’s actions on their own terms, rather than trying to fit them into an abstract theoretical framework which may have no relevance for the actors themselves [11].

Though CA’s insistence on focusing on the data and nothing but the data is considered by some to have its drawbacks, it is widely acknowledged to have certain advantages: if you cannot make reference to what is not in the data, you are impelled to pay very close attention to the fine entails of what is there. CA is a ‘microanalytic’ approach, which takes apparently mundane and unremarkable spoken interactions and finds intricate patterning in the way they are organized [11]. Just as putting a snowflake under a microscope reveals structure and complexity which are not visible to the naked eye, in [11] it is claimed that putting talk under the conversation analysis microscope defamiliarizes what we normally take for granted, and reveals the unsuspected complexity of our everyday verbal behaviour. Harvey Sacks, the pioneer of CA, maintained that because things that appear obvious are so obvious, in everyday life one would never remark on them, and it may turn out on closer inspection to be less obvious than they seem [11].

Given CA’s approach that focuses on data and sequential patterning, CA observations of natural dialogue regularities and its microanalytic approach can help us understand how we can best recognise speech act patterns along side a turn-taking structure and use our findings to create natural and cohesive topic handling in human-computer conversation. Additionally, the foundational idea that we need to focus solely on what information and data the participants provide means we should already have enough information to offer an appropriate response based participant's utterance, and derivable information, alone.

2.3.1 Conversation Analysis and Turn-taking Observations

Harvey Sacks observed that that 'one speaker speaks at a time, and speaker change recurs' [11]. Despite this insight potentially seeming obvious and not worth thinking about, this is the very reason Sacks did find it worth thinking about [11]. Things that appear so obvious in everyday life, that no one would ever remark on them, may turn out to be less obvious than they seem upon closer investigation [11].

The following is a list of follow-up insights on turn-taking found to be true upon closer inspection:

1. Participants in conversation will not usually all talk at once, and conversely there will not usually be stretches of time in which no one talks at all.

2. Simultaneous speech and silence does occur but when they do they are often treated by participants as problems which need to be 'repaired' (as something other than the normal and desirable state of affairs). Thus if the analyst's claim is that 'one speaker speaks at a time', one would expect participants in talk to display their orientation to that pattern by treating instances of simultaneous speech as problems requiring repair [11]. This is what is happening when, say, a speaker who inadvertently overlaps another apologizes and stops speaking to let the other finish, or when three speakers who simultaneously self-select themselves out in such a way that only one continues.

3. For simultaneous speech, what typically happens is that one speaker wins the floor (the right to talk and be attended to by co-participants) while others fall silent.

4. In the case of silence that becomes long enough to feel awkward, what happens is that someone breaks into it and claims the floor.

5. The floor is constantly negotiated and renegotiated as a conversation goes along, the turns are not pre-given to each participant. The continual negotiation is a general feature of conversational organization.

6. Conversation analysis holds that talk is 'locally managed', rather than from their being compelled to follow a course of action that has been determined

in advance.

Based on these observations, Sacks, Schegloff, and Jefferson proposed a model of conversationalists' behaviour which they presented under the heading "A simplest systematics for the organization of turn taking conversation". The model has two main elements:

1. It says that speakers are aware that a turn consists of one or more (but not fewer) 'turn constructional units' [11] (units as grammatical entities, like a complete clause or sentence; or even a c-unit). People who are listening to someone else's speech can use their knowledge of the possible unit types to project the end-point of the turn currently in progress. Being able to do this is important, because the end of a turn constructional unit is potentially a 'turn transition relevance place', a point at which speaker change may occur.

2. A mechanism for allocating turns to particular participants in a conversation. When a turn transition relevance place is reached, what ensues is not a random free-for-all, with everyone present having an equal change of getting the floor next. Instead there is an ordered set of rules for allocation of the next turn:

   a. Current speaker selects next speaker (or if this mechanism does not operate, then...

   b. Next speaker self-selects (or if this mechanism does not operate, then...

   c. Current speaker may (but does not have to) continue [11]

The turn-allocation mechanism that takes precedence over the alternatives is for the current speaker to select the next speaker. If the current speaker does not select a next speaker, the second option is for someone other than the current speaker to select themself, by starting to speak [11].

In conversation analysis, analysts look not merely for regular sequential patterns in data, but for evidence that participants themselves are orienting to the existence of those patterns [11]. Therefore within the scope of the CA approach, in speech there are certain patterns participants themselves are aware of but this is not necessarily limited to the grammatical level units or patterns described in 1) main element of the above model for conversationalist' behaviour. From this, this thesis proposes there are potentially sequential patterns in speech acts and other pragmatic data that outline two main elements from a semantic and pragmatic perspective:

1. The ability for people who are listening to someone else's speech to identify not only when the possible unit types are leading to the end-point of the turn currently in progress, but also the end of the current topic being discussed.

2. A mechanism for allocation of the next turn and next topic. If we understand the rules of next turn allocation to be true, then within the context of turn-taking there must be overlying mechanisms and rules topic change between turns, i.e. could the rules for allocation of the next turn topic be:

   a. Current speaker selects ~~next speaker~~ continuation of current topic or end of current topic or introduce new topic (or if this mechanism does not operate, then...

   b. Next speaker ~~self~~ selects new topic or continues current topic (or if this mechanism does not operate, then...

   c. Current speaker may (but does not have to) continue with new topic

These above mechanisms under main element 2. seem obvious however the complete understanding of what the 'rules of next turn topic allocation' within the realm of CA are would require the observation of more data from this perspective and the identification sequential pattern of topic handling between turns (and even with each turn). This is an endeavour this thesis sets out to take on using machine learning methods applied to the dialogue act annotated SPAADIA corpora.

2.3.2 Use of CA Approach and Conversationalists' Behaviour Model

As mentioned above, the potential rules in main element 2. for 'rules for allocation of the next turn topic' mechanisms outlined seem obvious however to fully understanding what the rules of next turn topic allocation within the realm of CA would require the observation of more data from this perspective and the identification of sequential patterns in topic handling between turns (and even with each turn).

Under the assumption that sequential patterns can yield main element 1., the ability for people who are listening to someone else's speech to identify not only when the possible unit types are leading to the end-point of the turn currently in progress, but also the end of the 'current topic being discussed' in turn, in Chapter 5 we will use the CA approach along side machine learning methods to identify speech act, semantic and pragmatic patterns between turns in the SPAADIA corpus. Using a new topic label, derived from the SPAADIA annotations, can we fit a model that learns the described 'next turn topic allocation' we outline here and then use within this thesis' proposed Topic Handler model? Within the scope of the of CA approach and observations: yes, we should be able to understand what patterns in the current turn leads to a new or continued topic in the next turn i.e. we can understand what is mechanism is taking place in the current turn by looking to the following turn/'bit of talk' and see if a new or continued topic occurs.

In the next section we discuss research in dialogue grammars and modelling, and how they incorporate findings such as those described above by performing CA and turn-taking analysis.

2.4 Research in Dialogues and Dialogue Modelling

In this section, we turn to research in dialogue modelling. There are various proposed models for spoken dialogue systems. Within the Survey of the State of the Art in Human Language Technology [15], the authors discuss two related research goals that are often adopted by researchers of dialogue and three approaches to modelling dialogue. The two related, but at times conflicting, research goals are:

Goal i) to develop a theory of dialogue, including at least a theory of cooperative task-oriented dialogue, in which the participants are communicating in service of the accomplishment of some goal-oriented task. The objective of *Goal i)* includes determining:

- what properties of collections of utterances and acts characterize a dialogue of the genre being studied
- what assumptions about the participants' mental states and context need to be made in order to sanction the observed behaviour as a rational cooperative dialogue
- what would be rational and cooperative dialogue extensions to currently observed behaviour

Goal ii) to develop algorithms and procedures to support a computer's participation in a cooperative dialogue.

In terms of *Goal ii)*, in general there is no consensus on the appropriate research goals, methodologies, and evaluation procedures for modelling dialogue [15]. However, these goals align clearly with this thesis's current endeavour and are important to keep in mind.

In [7], authors discuss how dialogues (human-human or human-computer) typically involve several parties who take turns (as outlined in 2.3.1 regarding CA and turn-taking). In a human-computer dialogue, the number of parties is likely to exceed two when counting each process/window a human user might talk to as a party. The authors in [7] point out that if, beyond the turn taking aspect, dialogues were predictable, a dialogue grammar or a dialogue plan would seem to be the most appropriate model. It is clear that this is not the case. The authors of [7] provide this example dialogue to show the difficulties of real-life dialogues:

Customer: "I'd like to buy a ticket to New York please. How much is it?"

Clerk: "Five."

Customer: "Here's ten. By the way, where can I get a paper?"

Clerk: "Down the hall."

Customer: "Thanks. Where's the train leaving?"

Clerk: "From Platform 10 to the left."

The above dialogue reveals that plans are quite inadequate to handle the shifting of topics and implications of the utterances [7]. While it can be assumed that the clerk's first utterance of "five" will be correctly interpreted as "five dollars is the cost of a ticket" by the pre-processing system, any plan-based system is likely to be thrown off by the customer's question of where to buy a newspaper since this is outside the original scope of the dialogue [7]. Therefore we cannot assume the same topic will be discussed when modelling dialogue, nor can be preset the conversation type or number of topics covered.

2.4.1 Dialogue Grammars and Dialogue Modelling

In [15], dialogue grammars are described as an approach with a relatively long history of development. This approach is based on the observation that there exists a number of sequencing regularities in dialogue (adjacency pairs) [15]. Similar to what was described in Section 2.3.1 regarding CA approach and observations, theorists have proposed that dialogues are collection of such act sequences (e.g. question then answer, proposals and acceptances). These collections are also said to contain embedded sequences for digressions and repairs [15]. For some, the importance of these sequences derives from the expectations that arise in conversation given the initial observations (e.g. on hearing a question one expects to hear an answer), however people can easily violate these expectations [15]. Therefore, dialogue grammars are a potentially useful computational tool to express simple regularities of dialogue behaviour but needs to function in concert with an approach or method that would allow for such violations.

Based on the above mentioned observations of conversation, phrase-structure grammar rules following the Chomsky hierarchy, or various kinds of state machines, have been proposed. The rules state sequential and hierarchical constraints on acceptable dialogues, similar to syntactic grammar rules, state constraints on grammatically acceptable strings [15]. The terminal elements of these rules are typically illocutionary act names (e.g. request, reply, offer, question, answer, propose, accept, reject, etc) and the non-terminals describe various stages of the specific type of dialogue being modelled (e.g. initiating, reacting, and evaluating) [15]. Theorists believe that dialogue grammar rules can be used in parsing the structure of dialogues just as syntactic grammar rules can be used in parsing sentences. With a bottom-up parser and top-down prediction, it is expected that such dialogue grammar rules can predict the set of possible next elements in the sequence and if the grammar is context-free, parsing can be accomplished in polynomial time [15].

From a state machine perspective, a speech act becomes the state transition label [15]. When the state machine variant of a dialogue grammar is used as a control mechanism for a dialogue system, the dialogue system first recognizes the user's speech act from the utterance, makes the appropriate transition, and then chooses one of the outgoing arcs to determine the appropriate response to supply [15]. When the system performs an action, it makes the relevant transition and uses the outgoing arcs from the resulting state to predict the type of response to expect from the user [15]. Modelling these transitions (or a communication step using our own proposed Topic Handler) in further detail, again, is the focus of this thesis (i.e. developing a way in which these transitions can be performed efficiently/accurately) and we will return to focusing on this step in following sections.

2.4.2 Previous Arguments against Dialogue Grammars and Recent Advancements that Relieve Main Issues

There are, however, arguments against the use of dialogue grammars as a general theory of dialogue [15]. For instance, dialogue grammars require the identification of the communicative action(s) that are being performed by the speaker in issuing an utterance. This is a difficult problem for both people and machines, and past solutions to this have required plan recognition [15]. However as we saw in Section 2.2, dialogue act recognition techniques have made many advancements in the the last two decades (e.g. DART) therefore the use of dialogue grammars is more possible than ever for human-computer dialogue model implementation.

As well, the model typically assumes only one state results from a transition but utterances are multifunction (e.g. an utterance can be both an assertion and a rejection and the speakers response may be expected to address more than one interpretation). Therefore a dialogue grammar subsystem would need to be in multiple states simultaneously which is a property that is not typically allowed [15]. This would require the consideration of compound speech acts for an utterance (as DART allows [50]) or multi-level speech act classification as is allowed in consulting corpora for creating the consulting dialogue system described in [34] (see Section 2.2.2).

Additionally, dialogues contain many instances of speaking using multiple utterances to perform a single illocutionary act and to analyze and respond to such dialogue contribution using a dialogue grammar, a calculus of speech acts needs to be developed that can determine when two speech acts combine to constitute another. At the time of [15], no such calculus existed (and not sure if one now exists). However, again with the advancement of automated dialogue act annotation in recent years, such patterning of illocutionary acts and utterance for sequences can be tracked more efficiently.

And finally, the dialogue grammar model does not say how systems should choose amongst next possible moves (i.e. the states currently reachable) for it to play its role as a cooperative conversant.

Therefore, dialogue grammars are a potentially useful computational tool to express simple regularities of dialogue behaviour, but they need to function in concert with more powerful methods, such as plan-based approaches, because as a theory dialogue grammars are unsatisfying as they provide no explanation of the behaviour they describe (i.e. why the actions occur where they do, why they fit together into a unit, etc) [15]. In the next section, look at plan-based models of dialogue using dialogue grammars.

2.5 Plan-based Models of Dialogue

Plan-based models are based on the observation that utterances are not just strings or words, but are the observable performance of communicative actions/speech acts [38] (e.g. requesting, information, warning, suggesting, confirming [15]). As well, humans do not just perform actions randomly but rather they plan their actions to achieve certain goals. In the case of speech acts, goals include changing the mental states of listeners (e.g. speakers' requests are planned to alter the intentions of their addressees) [15]. Plan-based theories of speech acts and dialogue assume that the speaker's speech acts are part of a plan and the listener must not just reply to the utterance, but uncover and respond appropriately to the underlying plan [15].

The major accomplishment of plan-based theories of dialogue is to offer a generalization in which dialogue can be treated as a special case of other rational non-communicative behaviour [15]. The main elements are accounts of planning and plan-recognition, which employ various inference rules, action definitions, models of the mental states of the participants, and expectations of likely goals and actions in the context. The actions may include speech acts. The execution of a speech act affects the beliefs, goals, commitments, and intentions of the conversants [15]. This cooperative dialogue model solves the problem of indirect speech acts as a side-affect, in that when inferring the purpose of an utterance, it may be determined that not only are that the speaker's intentions are those indicated by the form of the utterance, but there may be other intentions the speaker wishes to convey [15]. For example, in response to the utterance "There is a little yellow piece of rubber," the addressee's plan recognition process should determine that not only does the speaker want the addressee to believe such an object exists, the speaker wants the addressee to find the object and pick it up. Therefore, the utterance could be analyzed by the same plan-recognition process as an informative utterance to both request to find it and pick it up [15].

However there are a number of theoretical and practical limitations to this class of models (though not as limiting as dialogue tree models/implementations). The four limitations can me summarized as follows:

1. Illocutionary act recognition is redundant.

As mentioned above, an illocutionary act is action performed in saying something (e.g. informing, requesting, asking, answering, warning, apologizing, etc). Plan-based theories and algorithms have been tied tightly to illocutionary act recognition because in order to infer the speaker's plan, the listener has to recognize what single illocutionary act was being performed by each utterance (even indirect utterances) [15]. However, in the Allen and Perrault model [33] and other inferences in the scheme, illocutionary act recognition has been show to be redundant. It is argued that illocutionary acts could more properly be handled as complex action expressions, defined over patterns of utterance events and properties of the context, including the mental states of the participants. Through this analysis, a theorist can show how multiple utterances together constituted the performance of a given type of illocutionary act but conversational participants are not required to make these classifications (i.e. they need only infer what the speaker's intentions are) [15].

2. Discourse versus Domain Plans

   Although the model is capable of solving utterance interpretation problems using non-linguistic methods (e.g. plan-recognition), it does so at the expense of distinctions between task-related speech acts and those used to control the dialogue, such as clarifications [15]. In order to handle these prevalent features of dialogue, multilevel plan structures have been proposed, in which a new class of discourse plans is posited, which take task-level or other discourse-level plans as arguments. These are metaplans which capture the set of ways in which a single plan structure can be manipulated. So instead of infer directly how utterances further various task plans, as single-level algorithms do, various multilevel algorithms first map utterances to a discourse plan, and determine how the discourse plan operates on an existing or new task plan. Just as with dialogue grammars, multi-level plan recognizers can be used to generate expectations for future actions and utterances, thereby assisting the interpretation of utterance fragments and even providing constraints to speech recognizers [15].

3. Complexity of Inferences

   The processes of plan-recognition and planning are combinatorially intractable in the worst case and sometimes are even undecidable [15]. The complexity arises in the evaluation of conditions and in chaining form pre-conditions to actions they enable. Restricted planning problems in appropriate settings may still be reasonably well-behaved but practical systems cannot be based entirely on the kind of first-principles reasoning typical of general-purpose planning and plan-recognition systems [15].

4. Lack of a Theoretical Base.

   The plan-based approach still lacks a crisp theoretical base [15]. For example, it is difficult to express precisely what the various constructs (plan, goals, intentions, etc) are, what the consequences are of those ascribing those theoretical constructs to be the user's mental state, and what kinds of dialogue phenomena and properties the framework can handle. Due to the procedural nature of the model, it is difficult to determine what analysis will be given and whether it is correct, as there is no independently stated notion of correctness (i.e. what is missing is a specification of what the system should do) [15].

2.5.1 Plan-Based Models of Dialogue Conclusion

Given these limitations, plan-based models of dialogue are not able to readily compensate for topic changes between turns without explicit definition of the dialogue's purpose (i.e. preset conversation types or goals would need to be predefined). Because of these limitations other types of grammar models have been proposed. In Chapter 4, we discuss alternative grammar models (including the Parallel Communicating Grammar System (PCGS) for modellling for dialogues) and how they fit the goals of this thesis.

2.6 Summary of Speech Act Theory, CA, and Dialogue Grammars and Models

In this Chapter, we demonstrate that the idea of identifiable patterns in dialogue is not a new idea (e.g. adjacency pairs). The understanding that current turn patterns can help identify, or even define, the appropriate response in the next turn is a clear theme across the above discussed areas of research. Using this understanding to our advantage, this thesis will propose a Topic Handler model that will utilize speech act and semantic patterns in order to aid virtual agent in appropriately introducing or continuing topics in discussion. In order to provide a proof of concept, we will do this in the context of the video game environment, as a Topic Handler submodule of the Dialogue Engine proposed by Rose in [36].

However, in the next Chapter, it is important to step back and look at the history of human-computer dialogue as well as existing virtual agents and their current limitations. Then in the Chapter 4 we highlight proposed models and how their methodology overcomes current limitations.

3 Human-Computer Dialogue

3.1 A History of Believable AI and Dialogue

The development of artificial agents, such as chatbots, who can engage in dialogue with a human at a seemingly human level is not a new idea and it is by no means an endeavour restricted to only one field. Since Turing proposed his test of intelligence in [44], the Turing test, conversation has stood as the test for intelligence. Turing posited the questions “Can machines think?”, or rather, “Are there imaginable digital computers which would do well in the imitation game?” [44]. The imitation game involves a dialogue between a human interviewer and two interviewees (one man, one machine/digital computer). The interviewer stays in a room apart from the other two. The man and the machine are to compete as to which one of them is more convincing as a woman, and the interviewer would judge based on their answers who is the woman [44].

The Turing test is similar to the imitation game, but instead the object of the game is for the human interviewer to determine which of the two interviewees is human and which is machine [44]. If the interviewer is unable to tell which interviewee is human and which is machine, then the machine must be considered intelligent [30]. The above question: “are there imaginable digital computers which would do well in the imitation game?” is asking if one of the two interviewees is human, and one is a machine, would the machine be able to convince the interviewer that they are in fact the human interviewee. The Turing test has become the most infinitely used test for machine intelligence and has lead to the creation of the Loebner prize competition held annually from 1990 to 2018. The Loebner gold medal prize is to be awarded to the developers of the first machine to pass the Turing test imitation game. The requirements for winning the Loebner gold medal requires the winner’s work to respond in a human-like way to questions, images, and videos ([30]). While our endeavour is not one of this magnitude, it is still in line with the goal of this thesis.

As mentioned above, the Loebner gold medal prize has yet to be awarded to the first computer whose responses are indistinguishable from a human’s, however there are annually winners of the contest. A bronze medal is awarded to the most human-like AI entry relative to other entries that year, irrespective of how good it is in an absolute sense [1]. Many famous chatbots were created in an effort to win the bronze and gold medal prize (e.g. ALICE, Albert One, George, Rosetter, Chip Vivant, and Mitsuku) [12] however to date the gold medal prize has yet to be awarded [1].

In [36], Rose suggests that if the expectation of an NPC in a video game is to behave as a human in a convincing manner, then the NPC should be able to pass the Turing Test. NPCs are arguably not that different from chatbots other than they are used in a video game context, in fact chatbot engines is a popular method in which video game developers create NPCs capable of dialogue (see Section 3.2.4). To be able to create an NPC capable of passing the Turing test, the NPC needs the following [36]:

a) natural language processing (to understand what the human said)

b) knowledge representation (to store its knowledge)

c) automated reasoning (to use information to answer questions as well as derive new information)

d) machine learning (to find patterns in order to adapt to new situations) [36] [37]

Therefore, in the development of this thesis' proposed Topic Handler model, each of these capabilities are features are considered in developing a topic handler, in particulate c) automated reasoning and d) machine learning.

In summary, engaging in dialogue with computers has come a long way since the beginnings of chatbot development, with earlier bots designed as academic testing tools and the more recent bots being more sophisticated with use in various applications (e.g. interactive games, website navigation tools, and simulated personal help desk assistants) [12] however chatbots still have yet to achieve certain capabilities that would provide a more realistic dialogue experience as required to pass the Turing test.

3.2 Current Chatbot and NPC Development Methods

In order to understand the best means for improving topic handling in human-computer dialogue, it is important to understand how human-computer dialogue is currently being developed. In the previous section we outline motivations behind such research and how even state of the art virtual agents fall short of complete human-level dialogue, but what about more commonly used NPCs and chatbots? How are they implemented and what limitations are commonly encountered? In the following subsections we look at popular methods in creating NPCs and chatbots.

3.2.1 Methods of Creating Game AI and In-Game Dialogue

Immersive virtual environments can provide users with the opportunity to escape from the real world however scripted dialogues can disrupt the presence within the world the user attempting to escape within [17]. Scripted dialogues between NPCs and players, or even NPC to NPC, can be non-natural and rely on responding to pre-defined dialogue. This does not always meet a players emotional expectations or provide an appropriate response given the virtual environment state or current context [17]. There is much room for improvement, particularly in allowing players to type whatever they want and engage in conversation with an NPC, and there exists a common interest in the community in using natural language processing techniques to manage and mediate plausible and contextual interaction [17]. Comparative to work that been conducted in managed scripted systems, less research exists in the field of autonomous natural language interfaces [17].

While improvements for NPC-human dialogue is an impactful area for improving immersive game experience, it is potentially the most difficult way for NPCs to display intelligence [36]. As of now, the vast majority of game dialogue is scripted and the consequence of this is that either NPCs will repeat themselves unnecessarily, or they will ignore the player entirely, and once the NPCs run out of new things to say they will no longer seem like autonomous beings [36]. Vast amounts of research has already been done in game AI, yet video games continue to use outdated dialogue techniques for NPCs to interact with human players [51]. Given the difficulty of the problem, game designers are hesitant to improve game dialogue as their confidence in the necessary current techniques and the high-level AI is already low [36, 2]. Therefore, while research on improving the state of dialogue in video games has increased over the last few years, it is still necessary to investigate such research and discuss what research is still needed.

3.2.2 Scripting In-Game Dialogue

In creating dialogue for games, scripting is often utilized to manage vast amounts of dialogue, especially in role playing games (RPGs) [9]. Scripts are often used to control the flow of dialogue between a NPC and the player. A typical script would allow for a dialogue as described above: the player is prompted with a question by an NPC then given a few options through which they can respond, then based on the response selected by the character, the NPC would then respond back [9]. Here is a such a sample dialogue script for "Eric the Gross Nosed" NPC:

FUNCTION DialogueWithGrossNosedEric(Player plyr)

Speak("Welcome stranger. What brings thee amongst us gentle folk?")

int reply = plyr.SpeakOption(1, "Yo dude, wazzup?", 2, "I want your money, your woman, and that chicken")

IF reply == 1 THEN

Speak("Wazzuuuuuup!")

ELSE IF reply == 2 THEN

Speak("Well, well. A fight ye wants, is it? Ye can't just go around these parts demanding' chickens from folk. Yer likely to get that ugly face smashed in. Be off with thee!")

END IF

FUNCTION[9]

This script would be called by the main game code on the occurrence of a specific event, for example if the player enters a certain vicinity of Eric the Gross Nosed [9]. While scripting dialogue in this manner does not require consideration for unpredictable topic handling, it does not allow for very believable dialogue scenarios. Additionally, while utilizing scripts in this way makes it easy for game designers to write large amounts of dialogue quickly and easily, it again does not allow for very believable dialogue scenarios (only set turn taking points occur, no natural language input allowed, preset number of topics, no opportunity to introduce new topics, etc).

Such an approach could be done using bespoke XML files to plan dialogue so it integrates well with the rest of the game [42]. More advanced software such as visual dialogue engines (e.g. Chat Mapper) allow a user to describe their NPCs and map out branching dialogues [42]. Chatbots, which are specifically focused on engaging dialogue, encourage free text input and can produce a variety of output [42].

In order to create believable natural language interaction with NPCs, the author of [42] suggests the required skill set to make use of these approaches and available software is somewhere between a developer and a writer. This can be a very fun and creative process butan also take a long time using they approaches. Developers may find the writing aspect tedious and writers may find the logic aspect overwhelming [42]. In [42], the author also suggests that a good starting point for developing an NPC with the ability to engage in natural language dialogue is if to begin with chatbots and chatbot engines (section 3.2.4). However, video game environments are no stranger to natural language input.

3.2.3 Natural Language Interaction in Past Games

As mentioned above, compared to work that has been conducted in managed scripted systems, less research has been done in the field of autonomous natural language interfaces [17]. In [51], the author mentions that in its simplest form, the use of natural language input in video games dates back a very long time. Games, such as Zork [4] enabled free-form text-based interaction between the user and game, however such systems were far from a true natural language interaction. Such systems simply scanned the string input for keywords to interpret as command [51]. More recently, games have attempted to make natural language interaction more immersive and believable but still do not truly offer a natural language interaction [51]. The input is still scanned for certain keywords that are then used as input into a complex dialogue graph. This dialogue graph, although highly dynamic and believable, took several hundred thousand lines of code to fully author [51].

After the early days of adventure games, natural language interaction in games has been uncommon, most recent games that are considered to have a highly interactive and variable dialogue system have primarily relied on the players selecting from a finite set of options in order to traverse a dialogue tree or dialogue graph [51].

More recent research in [17] has been to improve immersive experiences in a virtual environments by applying AI and natural language processing to generate dynamic human-like responses within a themed virtual world. Research proposes a framework for text generation models in narrative authors for virtual environments, offers a platform for interfacing with contextual trained models via web requests, and a procedure for evaluating response quality from semantic NLP model output against ground truth human-sourced responses [17]. In [17], an initial proof of concept is provided and results yield useful findings aligned to natural language for immersive worlds. Authors in [17] mention that while the technology is still in its infancy, the system of dialogue generation through natural language processing described can be built upon and incorporated as a method for increased engagement or at least would benefit from an enriched experience through the use of believable characters that provide the user or player with unpredictable dialogue and narrative. Current limitations of this recent work include slow processing times based on the generation of dialogue via the models (however further development could include pre-processing responses in advance to reduce time significantly) and size of initial data sets use to find tune each thematic model (larger, richer data sets provided richer and more accurate responses) [17].

Because the technology of natural language synthesis is still very much in its infancy, it is difficult to provide players with believable interactions. In order for games to create more immersive and believable behaviour through dialogue interactions, very extensive authoring would be required if implementing natural language interactions in this manner continues. It would require designers to anticipate each possible situation and the effects it has on the game world and characters [51]. This would only result in the loss of emergence (i.e. result in a limited number of possible outcomes) [51]. Therefore methods such as using chatbot engines have been attempted. In the following section we look at chatbot development.

3.2.4 'Simple' Chatbot Software and Realistic Dialogue in Video Games

The term 'chatbot' has a different scope from 'NPC'. At one end, the term chatbot covers virtual assistants for websites and smartphones (e.g. IKEA's Anna, Apple's SIRI), and on the other end, it refers to the more academic challenge of the Turing Test to see whether an AI is capable of imitating a human [42]. But because it is the chatting ability of NPCs that seems most lacking in current games, chatbots are a good place to start when thinking about how to improve the believability of NPCs [42], in addition to task-specific virtual agents. Chatbots now exist in various messaging platforms, such as Facebook Messenger and Skype, and are largely for customer service purposes [39]. Chatbots have evolved to interact via voice as well with most common example being virtual assistants like Apple's Siri or Google's Assistant. Such chatbots are readily available and embedded in smartphones or in smart home devices to control the Internet-of-Things enabled devices [39]. Popularity of chatbots has increased over the years due to increase in Internet users and e-commerce in the last decade [39]. According to Harvard

Business Review, a mere five-minute delay could decrease a business's chances of selling to a customer and a ten-minute delay could reduce the chances by 400% and such instances have resulted in increased employed of chatbots that can handle inquiries 24/7 [39].

In [39], authors admit there is still a gap between existing chatbots and chatbots intelligent enough to replace human representatives. Given the increasing popularity of chatbots and overlapping goal of achieving human like dialogue, understanding challenges in chatbot development can help in research to improve NPC-human dialogue. Chatbot development currently faces two challenges [39]:

1. chatbots can only recognize specific sentence structures
2. the responses generated by existing machine learning techniques are not always accurate or personalized

Initial chatbots development involved simple pattern matching and simple "Q&A" style (similar to adjacency pairs discussed in section 2.4.1) and moved toward a more human-like way of carrying out and continuing conversations [39]. Advanced chatbots are now expected not only answer questions but also learn and improve themselves with each conversation, and eventually be able to respond appropriately in various context [39]. In order to achieve this, author in [39] found that current chatbots use machine learning as well as natural language processing within the domain of artificial intelligence. We will discuss advanced chatbots as their limitation in more detail in Section 3.3 .

Similar to the dialogue tree method, chatbot makers rely on the creation of a file or some sort (text, XML or spreadsheet) which is then processed by the software and turned into a bot which can chat as well [42]. The format usually associates a player's input (e.g. "which way to the dungeon?") with the chatbot's output (e.g. "second cave on the right") directly. However, the same problem still presents itself, the developer/writer must cover as many of the possible questions-answers about the "path to the dungeon" in a reasonable amount of time. The easiest approach is to look for the keyword "dungeon" in the user input and then always give the same reply, which is pretty much what NPCs do now and could be programmed quite simply without any third party software [42].

Therefore one of the main challenges is to build a chatbot which can recognize thousands or millions of inputs without you having to type them all, and which can give many variations of output. This challenge aligns with the overall challenge in developing a believable dialogue interaction with an NPC and the thesis' goal of creating a way in which multiple topics can be recognized and therefore appropriately covered with ease and in a natural manner (without having to pre-program every single possible utterance and each topic is addressed at appropriate points in dialogue).

The author in [42] outlines three of the most popular chatbot creators: Cleverscript, AIML and ChatScript. Cleverscript is a chatbot builder made by the company behind the very popular Cleverbot. It uses a spreadsheet format and has powerful tools for organizing language and maintaining variables [42]. The spreadsheets are uploaded at the Cleverscript website and turned into bots which you can test. The lines of the spreadsheet look roughly like:

input hello hello / hi / hi there

output hello Hello!

To use the bots in your own app or website, payment is required for access to the web service or to request libraries for inclusion in smartphone projects [42]. One of Cleverscript's big advantages is that it includes a mini-Cleverbot engine, which means you do not have to script all conversation. It will be able to small-talk right out of the box [42].

AIML (Artificial Intelligence Markup Language) is an XML format used for creating a chatbot [42]. The format looks like this:

WHAT IS YOUR NAME

My name is John.

ALICE (Artificial Linguistic Internet Computer Entity) is the most famous AIML chatbot [42]. When you create a AIML bot, you can include ALICE's AIML in addition to libraries contributed by other AIML users (e.g. several European languages). This reduces the amount of data you are required to enter. Once you've created your XML AIML file, you can run it using the free web service provided by Pandora Bots or other interpreters that are available in different programming languages such as C++ and Java so you can run it yourself [42].

ChatScript is another tool that can be used to create a chatbot. It is a free open-source C++ library [42]. ChatScript allows you to compile it for whatever platform you need. Outfit 7 used ChatScript to create their popular Tom Loves Angela App for Google Android and iOS. Bots are created using a text file which contains lists of expected user inputs and corresponding bot outputs that look like this:

u: (what are you) k̂eep() r̂epeat() I am a bot.

The ChatScript format includes methods for reusing text and remembering user preferences [42]. The text file is run though the software to produce a chatbot and the software can be compiled into your own game and run directly on a smartphone or your web server [42].

While Cleverscript, AIML and ChatScript methods for developing a chatbot or NPC are convenient and less time consuming than scripting every single possible outcome, the still do not offer realistic dialogue experiences for a human interacting with a computer. Each of the above examples allow for a user to highlight topics and chatbot responses to said topics, however requires extensive implementation and do not necessarily allow the for a chatbot to react to any information than the most recent non-bot human utterance. In the next section, we take a look at more advance chatbot implementations.

3.3 Limitations of More Advanced Chatbots

As mentioned in the previous section, given the increasing popularity of chatbots and overlapping goal of achieving human like dialogue, understanding challenges in chatbot development can help in research to improve NPC-human dialogue. Advanced chatbot development currently faces two challenges [39]:

1. chatbots can only recognize specific sentence structures
2. the responses generated by existing machine learning techniques are not always accurate or personalized

Initial chatbots development involved simple pattern matching and simple “Q&A” style, similar to the software described in the previous Section 3.2.4. Since then chatbots have moved toward a more human-like way of carrying out and continuing conversations [39]. Advanced chatbots are now expected not only answer questions but also learn and improve themselves with each conversation, and eventually be able to respond appropriately in various context [39]. In order to achieve this, author in [39] found that current chatbots use machine learning as well as natural language processing within the domain of artificial intelligence. Current chatbots have three main components: natural language to categorize the user's intent, dialogue management to determine the user's intent, and natural language generation to generate a response in natural language. After determining the category of relevant answers, the chatbot must first construct the relevant, personalized response using natural language and then respond with no time delay [39].

In the following sections we give an overview of research trends regarding development of human-like chatbots capable of closing the gap between existing chatbots and autonomous conversational agents (in particular research focusing on improving relevance of chatbot response), then take look at two comprehensive chatbots that have been developed, one for general purpose use/conversation and one for specific customer service use. Both exceed the capabilities of the aforementioned chatbot development methods and engines but still have certain limitations. After these limitations are discussed we argue that a Topic Handler, such as the one proposed in this thesis, could help make for a more natural human-computer dialogue.

3.3.1 Limitations of A General-Purpose Chatbot

In [12], the author investigates the limitations of current chabots and addresses their inability incorporate context, and in doing so hopes to improve the conversational power of chatbots. In [39], the authors also highlight that while existing advanced chatbots have few limitations, the main challenge for a chatbot right now is understanding the context in a conversation and generating a relevant response. With a general-purpose chatbot, this is an even bigger challenge it requires chatbot to narrow down the context for broad scope.

Author in [12] first considers the chatbot Mitsuku, the five-time winner of the Loebner bronze medal [1]. Mitsuku is a general purpose chatbot (i.e. attempts to hold an open conversation in no specific area). Author in [12] uses the below example to differentiate the concept of "chatting" as in merely uttering something and holding a conversation. This concept of chatting implies there is something more involved [12]. The following is a snippet of an actual conversation with 2013 Mitsuku (please note: unable to obtain newer snippet from more recent competition of Mitsuku conversation therefore focus 2013 to demonstrate persisting limitations across general topic chatbots):

Mitsuku: Good evening. My name is Mitsuku. What is your name?

Human: My name is Sally.

Mitsuku: Nice to meet you, Sally.

Human: Can you recommend a good store for designer clothing?

Mitsuku: I mostly wear jeans and a t-shirt.

Human: But I need a dress for a formal reception I am attending.

Mitsuku: Do you tend to like most things?

Human: No.

Mitsuku: OK. So, what do you do for a living?

Human: I am an accountant.

Mitsuku: What do you predict for the economy this year - bad or really bad?

While Mitsuku seems articulate enough, the conversations clearly did not go the way the human, Sally, probably expected. Consider the following snippet as well [12]:

Misuku: Welcome back. How are you today?

Human: Not too good. I have to make tough decision.

Mitsuku: That's too bad.

Human: Can you help me?

Mitsuku: Not often. Can you?

Human: I need your advice with a personal dilemma.

Mitsuku: Do you find yourself envious of what others own?

Human: No.

Mitsuku: Why not? Do you live around here?

Human: Yes.

Mitsuku: Do you have any pets?

Human: No.

Mitsuku: That' s a definite no? I travel too much to really have pets. We do have two chickens in the backyard.

Clearly Mitsuku did not do well in these conversations. Although it is a state of the art chatbot and has won the Loebner bronze medal prize numerous times, the conversation for the first year win still seems unsatisfactory [12]. Author in [12] points out that there is a pattern: when Mitsuku's response is evaluated only in relation to the immediate previous utterance by the human, she grades satisfactorily, it is only when evaluated on a longer sustained conversation that she grades poorly. Mitsuku performs adequately in an isolated question-answer exchange, even over a series of several consecutive question-answer pairs [12]. However, author in [12] also argues that a series of question-answer pairs, or a series of one-to-one utterances, does not constitute a conversation. Author ibelieves that holding a conversation would be the next step in the evolution of chatbots, but even this current state of the art chatbot is unable to perform this task [12]. How could Mitsuku aim to allow for a longer, and more cohesive conversation? Again a Topic Handler would help in with this aim by helping the virtual agent understand that current discussion is still focused on the same, or a related, topic rather than switching topics after a question-answer exchange.

Now we turn to task specific chatbots and discuss their development methods, current capabilities, and current limitations. As mentioned before, Mitsuku is a general purpose chatbot (i.e. attempts to hold an open conversation in no specific area). This is clearly a very difficult task computationally, so could a chatbot with a more restricted scope be able to perform better?

3.3.2 Limitations of Customer Service Chatbots

In [12], author points out the 2 biggest limitations of current customer service chatbots:

1. Inability to not be able to hold longer conversation, understand the conversation, gauge whether the conversation is going in the desired direction, and act on it.

2. The inability for spotting opportunities in a conversation for providing unsolicited but pertinent information and act on it (either by disseminating

relevant information or by transferring the conversation to a human representative seamlessly).

Author in [12] believes that the missing element is context, resulting in these two limitations. Let us begin by looking at a real-world chatbot application to highlight deficiency 1. The following example conversation is with a customer service chatbot that helps customers solve account access issues. In this example the customer is unable to login to an account and seeks assistance [12].

Chatbot: Hello. How can I help you today?

Customer: I am unable to login to my account. I forgot my password.

Chatbot: Please click on the "Forgot Password" link. You will be emailed a link that will help you change your password.

Customer: I already tried that.That link takes me to a form to reset password. But it requires me to enter my Customer Relationship Number. I do not have this number.

Chatbot: The Customer Relationship Number can be found in the "Account Information" tab when you log in to your account.

Customer: But I am unable to login to my account. I do not remember my password.

Chatbot: Please click on the "Forgot Password" link. You will be emailed a link that will help you change your password.[12]

Again, we see that the chatbot can carry on a question-answer session satisfactorily, but it is unable to determine if the answer was satisfactory for the customer [12]. It is unable to tell if its response adequately addressed the customer's query, or if the customer was confused by the response [12].

A human representative would clearly see that the conversation is not helping address the customer's issue, but since the chatbot is programmed through pattern-template pairs they are more suited for question-answer type exchanges, where there is a sequence of several question-answer pairs. They are unable to hold a longer conversation, understand the conversation, gauge whether the conversation is going in the desired direction, and act on it [12]. In the above example, we would expect the chatbot to recognize the deadlock situation for the customer and to resolve it, or transfer the conversation to a human representative if the solution is beyond it programmatic capabilities [12].

In this next example, a customer service chatbot for online electronic trading website tries to help a customer resolve an unallocated transaction (i.e. the customer's margin account does not have enough balance to service a pending buy order):

Chatbot: Hello. How can I help you today?

Customer: I have a pending buy order and I am out of margin.

Chatbot: Buy orders are processed within 2 business days.

Customer: But I am out of margin. I am concerned that my buy order will not execute.

Chatbot: Margin can be restored by a new allocation request from your holding account. It takes 3 business days to execute.

Customer: But my buy order is already scheduled. I do not want to lose it.

Chatbot: Buy orders are processed within 2 business days. [12]

This example again demonstrates that the conversational performance of these chatbots do not live up to what we would expect from a human customer service agent [12]. There is clearly an opportunity, and potential need, to improve their conversational ability.

The below is an example of limitation 2. 'the inability to spot opportunities in a conversation for providing unsolicited but pertinent information' [12]. The exchange below is between a prospective insurance customer and a chatbot:

Chatbot: Hello. How can I help you today?

Customer: I received a quote for home insurance last week. I would like to purchase that policy.

Chatbot: All right. Do you remember the quote reference number?

Customer: No. I had printed it out but I left the printout in my car.

Chatbot: Not a problem. I can look it up by your last name and date of birth. What is your last name?

Customer: Johnson.

Chatbot: What is your date of birth?

Customer: April 21, 1973?

Chatbot: Great. I found your quote. Would you like to purchase the policy now? [12]

It would seem the chatbot handled the conversation satisfactorily, but there are some missed opportunities [12]. The conversation indicates the customer possibly owns a car and a human representative would have noticed this and spotted an opportunity for another sale [12]:

Human Representative: You mentioned that you have car. Did you know that if you get auto insurance in conjunction with your home insurance, it could save you $500 over 6 months?

Therefore, author in [12] suggests that the next generation in the evolution of chatbots should be able to spot such opportunities in a conversation and act on it. What seems to be the missing element in current chatbots is context. In a pattern of question-answer exchanges, the context switches from one pairwise utterance to the next, but in most conversations the context remains the same throughout the exchange of utterances [12]. Contemporary chatbots are unable to adhere to context in conversations.

Conversations are abound with uncertainties of various kinds that may lead to misunderstanding and other communication failures. These uncertainties permeate every level of conversation, form attending to what was said and identifying what words were spoken, to understanding the intentions behind words [12]. While human representatives manage these multiple uncertainties with almost effortless ease, chatbots often breakdown in these situations.

In [12], author proposed his own model and implementation for customer service chatbots. The proposed customer service chatbot offers a conversation model that uses preset context-specific types of conversations and models each type using goal-fulfillment maps, then uses speech act recognition to determine the next step in the automata for that type model (e.g. in informational conversation type, an assertive speech act can take the conversation back to the Advisory state and an Expressive speech act verifies that the procedure has been completed) [12]. While his model handles conversation flow and opportunities more seamlessly than Mitsuku, it is designed for a narrow situational context, it is limited to the outlined conversation types, and it is not clear how this model and implementation would do for general purpose. How would this chatbot recognize point at which a new topic can be proposed or when the current topic requires continuation without preset conversation types for a specific context identified at the beginning for every conversation?

3.4 Summary of Advanced Chatbot Limitations

Current methods for chatbot and NPC development are currently still lacking in terms of topic handling and flow of conversation without present conversation types. State of the art general-purpose chatbots, such as Mitsuku, excel only in relation to immediate previous utterance by the human but does not evaluate so well on longer sustained conversation. Context-specific chatbots excel better at this however require design for very narrow situational context for each dialogue.

As mentioned above, authors in [39] point out that even more recent advanced chatbots, while now see fewer limitations, still struggle with the main challenge of understanding the context in a conversation and generating a relevant response (which would include adhering to natural topic handling and flow). Same authors suggest that chatbots should implement improved natural language processing techniques to accurately recognize the content of the user input as well as learn to understand the context of conversations and respond accordingly with emotions or personalized context. If the ultimate goal of chatbots is to replicate human-human interactions continued focus on leveraging improved advanced technologies, machine learning and natural language processing-based artificial intelligence is needed [39]. And leveraging such technology and methodologies could help built a technique that allows for a more organic flow of topic and conversation is needed, one that does not rely on question-answer pairs (i.e. a series of one-to-one utterances) and again is not limited by:

1. Inability to not be able to hold longer conversation, understand the conversation, gauge whether the conversation is going in the desired direction, and act on it.
2. The inability for spotting opportunities in a conversation for providing unsolicited but pertinent information and act on it (either by disseminating relevant information or by transferring the conversation to a human representative seamlessly) [12]

We now look at proposed models that consider topic change in conjunction with the overall dialogue model design and in turn these models overcome the above two limitations outlined by [12].

4 Topic Handling in Human-Computer Dialogue Models

In this Chapter we look at existing models and systems that can aid in topic handling. We look at the conversation model and dialogue engine implementation proposed in [36] and how the model's proposed Topic Handler module would incorporate into the larger conversation model. We then take a look at the the StalemateBreaker conversation system that can proactively introduce new content when appropriate and how it detects such opportunities [29]. And finally, we look at research in dialogue grammar modelling for background on how to model our own Topic Handler. In particular we take a look at dialogue grammars and the Parallel Communicating Grammar System for modelling dialogue (PCGS model) proposed in [6]. In contrast to other dialogue models proposed in literature, the PCGS model can capture non-deterministic phenomena in dialogues like change of topic, questions of clarification, etc [6]. We begin by discussing the conversation model and dialogue engine found in [36].

4.1 Realistic Dialogue Engine for Video Games and Topic Handling

In [36], the author proposes a method to allow a human player to interact with NPCs using natural language input, an important feature for offering a realistic and natural language dialogue with a NPC. The method is a highly customizable conversation system that uses various AI techniques such as information retrieval and sentiment analysis, allowing NPCs the ability to engage in dynamic dialogue. This method allows NPCs to answer questions, ask questions, remember events, and more [36]. The customizability allows for the adjustment of NPC response types given a game's storyline. While the system proposed currently only allows for simple dialogue, it demonstrates the potential for a more robust way to simulate believable NPC-human dialogue in video games [36].

The author in [36] aptly points out that in order to fulfill the desire to allow the human player to interact with NPCs in a freeform way, and not in the multiple choice scripted manner, the use of natural language processing and information extraction will be required. The method proposed does not develop new techniques in either field, but assumes the various necessary tasks of these fields are solved (e.g. Part of Speech (POS) tagging and sentiment analysis) [36]. The method uses these modules as a given, making assumption that they

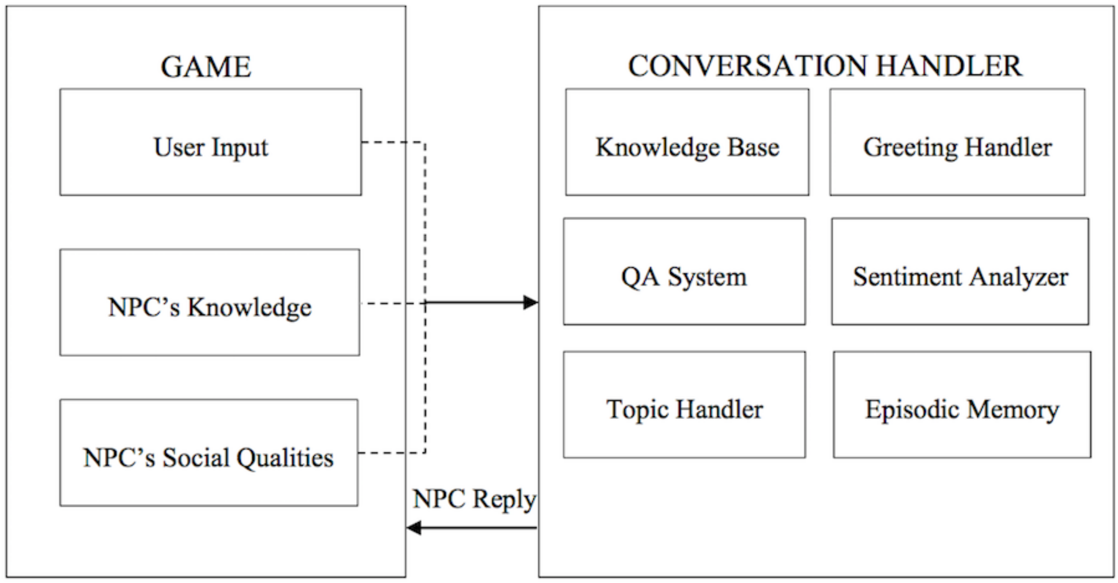


Figure 1: The Conversation Handler proposed by Rose in [36]

work perfectly even though they do not necessarily, and develops a framework for natural language game dialogue by integrating these elements into believable NPCs [36].

This model for a realistic dialogue engine incorporates natural language input, greetings, question-answering, small talk, ellipsis resolution, coreference resolution, named entity recognition and classification, ontology, relationship extraction, logic handler, sentiment analyzer, and a topic handler. While only a partial implementation was developed (allowing for natural language input, greetings, question-answering, small talk, and episodic memory), each of the above mentioned features were discussed with possible ways to implement [36].

In particular, the Conversation Handler incorporates a topic handler module that would be responsible for determining what topic the NPC should discuss with the player. The author in [36] says that it can either be a continuation of the current topic, or the Topic Handler can propose a new topic [36]. During a small talk stage, the topic handler must only select neutral subjects to help build rapport with the player, and during core stages the topic handler module is allowed to select from other types of topics that maybe be more polarized (e.g. asking which side of an in-game war the player supports) [36].

Within the Conversation Handler, the topic handler module would also allow for new topics to be introduced either by having the NPC ask the player a question or remember an event, but says that it is preferable to avoid introducing a new topic by simply reciting a fact from the knowledge base because the effect may be harry and could ruin the flow of the conversation. Instead the author's proposed topic handler would be stating facts that is only appropriate when continuing a discussion about a particular topic or when answering a query [36]. Based on this design, we can see how the Conversation Handler utilizes its various modules to resolve the two limitations current customer service chatbots outlined in the previous Chapter (i.e. 1. the inability to to hold longer conversation, understand the conversation, gauge whether the conversation is going in the desired direction, and act on it; 2. The inability for spotting opportunities in a conversation for providing unsolicited but pertinent information and act on it (either by disseminating relevant information or by transferring the conversation to a human representative seamlessly) [12]) and could better handle the examples conversations above provided the model it given the correct inputs at implementation.

In terms of implementing such a tool in the overall engine, author in [36] says that they would have the following items from a JSON file and stored them in memory: general topics to discuss, different utterances within each topic (including potential replies to some player statements), how emotionally charged the topics are, and how important the topics are for the game's storyline (JSON was selected because it can be edited by a human, and it is trivial to convert it to a Java object by using third-party Java libraries) [36]. Again, the topic handler module would have been responsible for deciding when to switch topics (e.g. if there are no more utterances to say for a topic, or if a particular topic is important for the plot of the game), and determining if the player has changed topics.

While this engine allows for a broader amount of variation in context than [12] customer service chatbot model and implementation, and offer more human aspects like small talk, it is still limited by the pre-defined topics and context of the game. If an input utterance is of a new topic, how would this new topic be recognized and stored? Would the description of this topic handler module be applicable/expandable for a general purpose NPC/chatbot design outside the realm of a defined game environment or just customer service?

4.2 StalemateBreaker: A Proactive Content-Introducing Approach to Automatic Human-Computer Conversation

The authors in [29] suggest that it is generally presumed that humans should take the role to lead conversations in human-computer interactions and introduce new context when a stalemate occurs, and the computer only needs to "respond." They observe that in human-human conversation, both participants have a duty to play a leading role in a continuous dialogue session. They discuss the observation that, shortly after the conversation begins, both parts are likely to be the stalemate breaker [29]. If only one side keeps finding something to talk about while the other side responds in an unmindful way, the conversation becomes less attractive and is likely to end pretty soon [29]. Therefore the authors in [29] suggest that in human-computer conversation, the computer side should also be initiative and introduce new context when necessary.

Following discussion of these observations, the authors in [29] propose the StalemateBreaker conversation system that can proactively introduce new content when appropriate. The proposed pipeline design helps determine when, what and how to introduce new context during human-computer conversation. The StalemateBreaker system is build upon a conventional retrieval-based conversation system, which is typically passive and the proactive content introduction begins with the first of the below steps shown in Figure 2. In this Figure, we see the process flow chart of the 4 steps within the content introducing system:

1. *Stalemate Detector*: First step in content introduction. Identifies appropriate location to introduce new content.
2. *Named Entity Detection*: Backtracks previous utterances in current conversation session and detects all names entities within a window.
3. *Candidate Reply Retrieval*: Uses entities and conversation context to retrieve up to fifty candidate replies from a large pool of collected dialogue data (Candidate reply much contain at least one entity.
4. *Selection by Reranking*: Finally the candidate replies are reranking by a random walk-like algorithm and is further enhanced by the proposed novel algorithm Bi-PageRank-HITs.

For the purposes of our study here, the step of interest is the stalemate detector and how it identifies the appropriate points to introduce new content. Authors points on in the more detailed description that this was not the main focus of the paper and therefore they kept the implemented approach simple [29]. The authors apply keyword matching of meaningless expressions, or discourse markers, like "..." or "Errr" [29]. In total, nearly a hundred filters were used. Authors provide experimentation and evaluated performance using human evaluation through

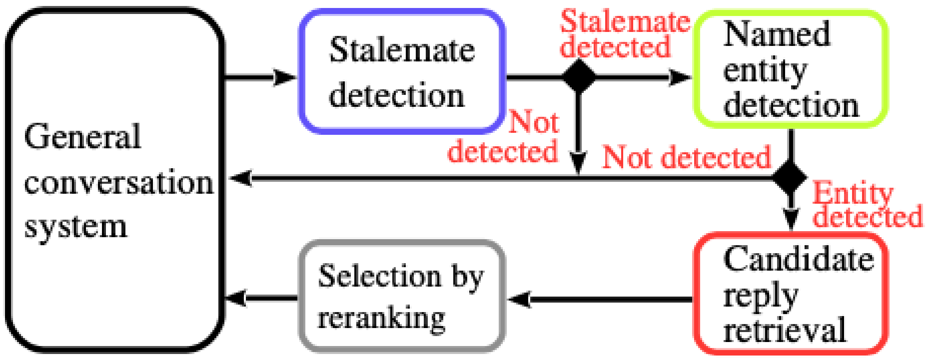


Figure 2: Process flow of triggering content introducing in [29]

a hundred and eighty sessions from real-world user conversation logs. The results demonstrates that both their content-introducing approach, including the relatively simple approach to step 1. above, and novel reranking algorithm, Bi-PageRank-HITS, are effective. In future work, they would like to apply learning-based sentence modelling for the stalemate detection step.

From the work in [29], we can see identifying points of introduction do not necessarily need to be a complex process however can still be further optimized using learning methodology as the heuristic methodology used here is not scalable and only identifies points in time where the stalemate is acknowledge with a keyword filter. Additional natural topic change points would be missed using this method long term.

4.3 Dialogues as Co-operating Grammars and Parallel Communicating Grammar Systems

In Section 2.4.1, we outlined research in dialogue grammar and their limitations and what recent advancements allow us to over come some of these advancements in implementation. In this Section we revisit dialogue grammars and look at modelled dialogue using the distributed grammar systems. In earlier work, the authors of [6] modelled dialogue between a human and a computer by using co-operating distributed grammar systems [7] and co-operating distributed grammar systems with memories [5]. These grammar systems consisted of several grammars working together in a sequential way (i.e. each utterance of a participant is modelled by a grammar and processed one after the other and by taking turns) [6].

In [7], the authors study how to model two-party, application-specific, non-standardized language dialogues (dialogues for short) using systems of co-operating grammars. Each of these grammars in the system corresponds to a party in the dialogue [7]. This representation is highly efficient both in the sense of description size and complexity of implementation [7]. The basic principle behind this model proposed in [7] is that participants in a dialogue have a common goal, a public goal, which may not be apparent at the beginning of the dialogue. A goal can be thought of as a topic that must be discussed in order to be resolved. This proposed model is formalized in terms of co-operating distributed grammar systems. The authors in [7] argue that on the syntactic level this model is significantly more economical than existing dialogue grammars or plan-based models.

In the follow up model to [7], dialogue structure is modelled using a co-operating distributed grammar systems with memories (mCD grammar systems). The memories utilized with the cooperating grammars in the system found in [5] contain private notes for each participant and in this way the private and public information can be kept separate [6]. In [5], the authors demonstrate that this model can deal with variability and unpredictability of dialogues (e.g. disagreements between participants, unexpected changes of opinions or topics, difficulties concerning comprehension and misunderstandings between the participants). In this model, some of these tasks are performed privately while others require communication. The distinction between private and public tasks is implicit [5]. The dialogue is regarded as a joint activity as in [15], that is the dialogue is modelled as a joint activity performed by all the participants of the dialogue. The participants share goals they want to achieve. There are two kind of goals to be distinguished:

1. Public goals are known to all participants - these goals are public information. All participants will try to achieve an accepted public goal in a cooperative fashion.

2. Each participant can have a private goal (beliefs, questions, etc.) of which the other participants do not know [5].

The mCD grammar systems model provides the possibility to represent public and private goals separately [5]. The definition of mCD grammar system is modified with the object of dialogue modelling in mind to the effect that no messages are sent and memory management depends on the single grammars owning the memory. The memories are used by the single grammars for keeping private notes. The public exchange of information is by means of the current sentential form [5].

4.3.1 Past Implementation of mCD-Grammar Systems

The work in [23] uses an example from an existing domain to show how the mCD-grammar system can be implemented in Prolog. The limitation of the dialogue in the given domain is realistic. Prolog was successfully used to construct the rules of the mCD Grammar System implementation. However, this implementation focused on showing the handling of the configuration and the utterances were pre-assigned symbols for each topic, and these symbols were collected and used in the implementation. The author in [23] maintains that dialogues between people are difficult, if not impossible, if the topic of conversation is not of knowledge to the participants. Therefore recognitions of new topics to be accounted for and resolve of this was not dealt with in this implementation.

4.4 The Parallel Communicating Grammar System for Modelling Dialogue

Building on their work in [5] and [7], the authors compare parallel versus sequential grammar systems for modelling dialogues. In a formal model for dialogues based on PCGS (Parallel Communicating Grammar System), each participant involved in the dialogue is modelled by a grammar, such that all participants of the dialogue are formalized together as a PCGS [6] is that of parallel communicating grammar systems as defined in [16] [46]).

PCGS model is more natural than the previously described sequential models, without giving up the advantages of the earlier sequential model discussed in the previous section [6]. Not only does the PCGS model allow for faster information processing than the sequential ones, there is the advantage that the use of any awkward memories that make a mathematical treatment difficult can be avoided [6]. The PCGS model allows utterances to be spoken by one participant while being heard and processed by other participants simultaneously.

The system grammars, that model the participants involved in the dialogue, work on their own sentential forms. The sentential forms represent the utterances and the progression of turn taking [6]. The sentential forms include the private goals of each participant, which are not known to the other participants of the dialogue, as well as public information as developed during the dialogue [6].

When modelling the PCGS, a master grammar is also needed, one that handles the general conventions of a dialogue for it to be successful (i.e. correct and reasonable) [6]. The master grammar controls and records the dialogue progress and handles the variability and unpredictability of dialogues. The master grammar can send queries to the other grammars, in this case the other grammars have to supply part of the sentential forms but never the participants private goals [6]. The master grammar issues queries whenever new information has been made available and tells when "new information is given" [6].

4.4.1 The Definition of the PCGS model

Definition 1 A parallel communication grammar system is a construct:

$$\Gamma = (\ N,\ K,\ T,\ G_1,\ G_2,\ \ldots,\ G_n,\ ).$$

with n ϵ N = { 1, 2, ...}, *with the following properties:*

- N is a finite non-empty set of nonterminal symbols, the nonterminal alphabet

- T is a finite non-empty set of terminal symbols, the terminal alphabet

- K is a finite set of query symbols, the query alphabet,

$$K = \{Q_1,\ Q_2,\ldots,Q_n,\ Ter\}$$

where the index i refers to the i-th grammar G_i of Γ.

-For i = 1, ..., n each $G_i = (N \cup K,\ T,\ P_i,\ w_i$ is a grammar with set of rewriting rules P_i and axiom w_i.

G_1 is said to be the master grammar (or just master) of Γ.

The formal definition of how a PCGS works can be found in [16][46]. It is assumed that all grammars G_i are context-free; this implies, in particular that $w_i = S_i$ for some symbol S_i ϵ *N*. Each grammar G_i maintains its own sentential form x_i, therefore the initial configuration of the system is the n-tuple $(S_1,\ldots,S_n)$. A configuration $(x_1,\ \ldots,\ x_n)$ is final if Ter is the last symbol of x_1. To determine a final configuration, the sentential forms $x_2,\ldots xn$ of the other grammars are not taken into account [6].

The step-wise behaviour of such a grammar system Γ depends on whether any of the sentential forms in the present configurations contains a query symbol or not. If no query symbols are present all grammars proceed independently according to their own rules. On the other hand, if a query symbol Q_i is present in any current sentential form, it is replaced by the corresponding sentential form x_i. This is called a *communication step*; whenever a grammar introduces a query symbol for normal re-writing process is halted until all query symbols have been replace. In this way any grammar can ask for information about the current sentential form of any other grammar in the system and include the information in subsequent computation steps [6].

To model dialogues, the authors in [6] claim that they do not need the full power of PCGS, it is sufficient that only the master grammar Γ can introduce query symbols, creating a *centralized* grammar system.

4.4.2 Motivations Behind Looking at the PCGS Model

The PCGS model has a master grammar which issues queries whenever new information has been made available by the participants grammars ("new information" includes change of topic, questions concerning comprehension, change of opinion, etc). For example the master grammar will introduce a query symbol after the utterance "Can you tell me how I get to the university by bus?" as this changes the topic of the dialogue. So how would the master grammar recognize the need for a new query symbol in the configuration in an implementation? There exist tools and implementations for allowing natural language input into a dialogue engine, but the module for tracking topics, topic change recognition, and resolving all conversation goals has yet to be developed.

We believe a combination of the parallel grammar system design, in which public and private goals/topics are tracking and updated in parallel by a master grammar or module, and a conversation handler model such as those discussed in [36] would aid in getting closer to developing such a system.

We believe the structure of the PCGS model can help deciphering the unpredictability of natural language dialogue between a human and computer, one of the main elements missing before implementing such is a topic/goal detector, or as a step towards this the recognition. Many conversation implementations discussed above require preset topics and/or conversation type that are pre-modelled in order for a new topic to be identified, let alone discussed. While every dialogue is set in some level of context, if we want to have virtual agents capable of general discussion at at human level, a topic/goal handling that does not require preset topics or conversation types must be achieved to over come the limitations of restricted context dialogue such as those outline by [12] in Section 3.3.1.

4.4.3 So what is still missing from PCGS Model for Dialogue

In the PCGS Model for dialogue, what is still not clear is how the master grammar would realize the public and private goals if implemented into a chatbot application or video game NPC. How does the communication step occur? How would the master grammar know when to introduce a query symbol, especially when used for a human computer dialogue?

4.5 Topic Handling in Human-Computer Dialogue Models Summary

In this Chapter we looked on previously proposed conversation systems and dialogue models and how they handle multiple topics in one dialogue. In particular, we focused on the topic handler module of the Conversation Handler model proposed in [36], the StalemateBreaker conversation system that proactively introduces new content when appropriate, and the PCGS model that handles topics by focusing on public and private goals (topics to be discussed and potentially resolved). The proposed methods of these models are useful in developing our topic handler, as they illustrate how topic handling can be integrated in parallel with a dialogue model, as well as what elements of a topic handler are important for the natural flow in human-computer dialogue (e.g. identification of new topic, flexibility without limitations of present conversation type, continuation of current topic until resolved) and in turn, the Topic Handler proposed in the next Chapter can aid in developing a full implementation of these three models.

5 Proposed Model for Topic Handling

Following the above discussions regarding proposed models that would overcome current limitations in human-computer dialogue with existing chatbots, in this Chapter, we propose a Topic Handler model for human-computer dialogue. The proposed framework is designed in such a way that is compatible with dialogue systems. In particular, is compatible with the work in [36] which proposes a comprehensive model for conversation handling. This Topic Handler model that could be used as the Topic Handler submodule to a full implementation of the realistic dialogue engine proposed in [36], could be used as a machine learning based stalemate detector step for the StalemateBreaker conversation system proposed in [29] and could lead to a complete implementation of a PCGS model in [6] by assisting in the creation of the master grammar and the communication step.

5.1 Topic Handler model

Based on the research discussed above, in Figure 3 we propose a Topic Handler structure. This model would allow the NPC to identify when they can propose a new topic (i.e. identify opportunities for providing unsolicited but pertinent information and act on it) or when it is most appropriate to continue the current topic in the next turn. Using speech acts and additional semantic and pragmatic information of each turn in a dialogue, this module will use the CA and turn-taking observations and based on the current turn patterns will predict if the next speaker should propose a new topic or continue the current one. In the following Section we discuss the role of each module of the Topic Handler model.

Figure 3 highlights each of the six modules and how they could be used by the Conversation Handler System proposed in [36] (see Conversation Handler system architecture in Figure 4). As outlined in previous section, the author in [36] proposes a Conversation Handler Model that incorporates multiple modules in order to facilitate NPCs that can engage in realistic dialogue with a human player. In the following subsection, we discuss each of the six Topic Handler modules in more detail.

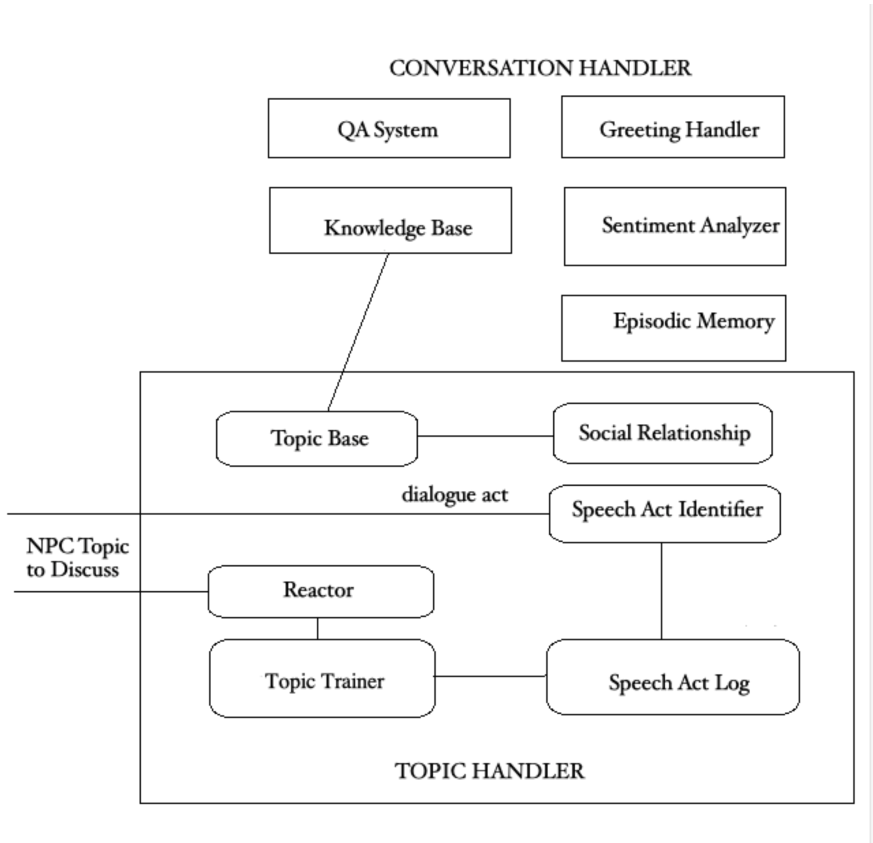


Figure 3: Expanded Topic Handler Model within Conversation Handler Model proposed [36]

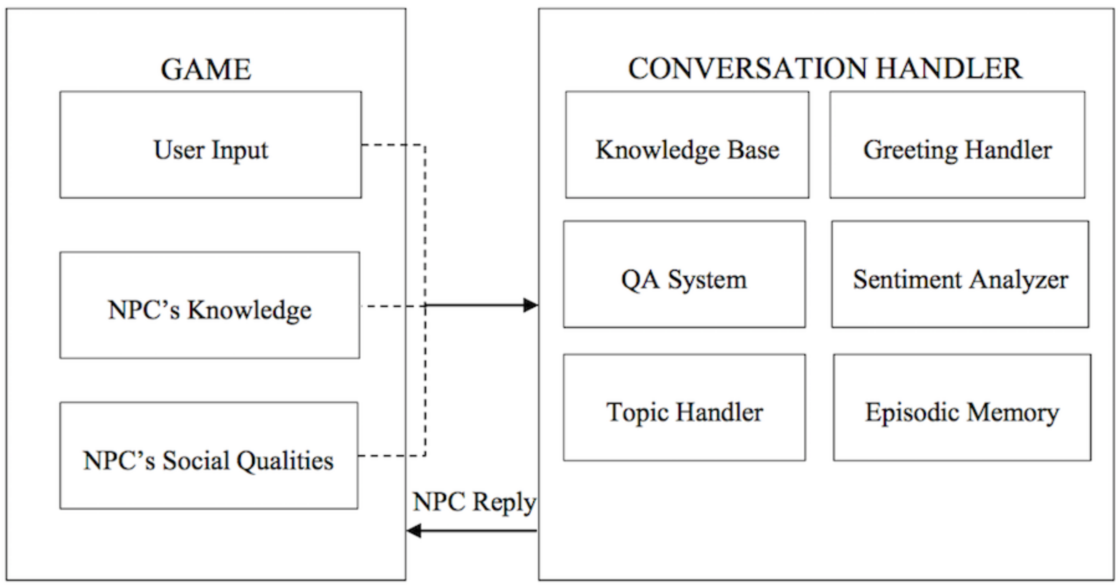


Figure 4: The Conversation Handler proposed by Rose in [36]

5.2 Pre-Dialogue Processing Modules

There are two pre-dialogue processing modules: Social Relationship Module and Topic Base module.

5.2.1 Module 1: Social Relationship Module

Composed of two submodules: roles and familiarity. The role submodule defines the role the participant is taking in the current interaction (acquaintance, mother, customer service representative). From this the familiarity, or social distance, is scaled and output to the Topic Base module. This is similar to the calculation done in the Greeting Handler module of the Conversation Handler model proposed in [36]. In The Greeting Handler, level of politeness is based on social distance, relative power of the listener, and the absolute ranking of the imposition [36]. From here the handler selects the greeting that most closely matches the necessary politeness level and ensures that the greeting makes sense in the current situation [36]. In the same manner, the social distance of the participants is defined here and input into Module 2: Topic Base where the appropriate topics that could, or should, be discussed is decided.

5.2.2 Module 2: Topic Base

Familiarity from the Social Relationship Module then refers to the Knowledge Base module of the Conversation Handler model, where the NPC's knowledge is stored and is dynamically updated as the game progresses with new facts continuously learned from the game or by the human player [36]. From the Knowledge Base Module, outside of the Topic Handler model, the Topic Base allocates the related topics for this social relationship. If the current social relationship is less familiar but not requiring institutional talk, then small talk of general topics are also included in the topic base (e.g. weather). In the following subsection, we turn to Topic Handler Modules that are active during dialogue processing.

5.3 Topic Handler Modules During Dialogue Processing

There are four dialogue processing modules: Speech Act Identifier, Speech Act Log, Topic Trainer, Reactor.

5.3.1 Module 3: Speech Act Identifier

The Speech Act Identifier receives the DA as input and outputs the DA's speech act, semantic and pragmatic data (e.g. current speaker, turn count, polarity, etc; in the next Chapter will go into more detail regarding what semantic and pragmatic data can or will be identified by this module).

5.3.2 Module 4: Speech Act Log

The Speech Act Log keeps record of the DA and its corresponding semantic and pragmatic data in the order they occur.

5.3.3 Module 5: Topic Trainer

Following each DA and turn, the Topic Trainer module accesses the existing data in the Speech Act Log and utilizes patterns in speech act and additional stored features in order to predict if in the next turn's initial dialogue act it is appropriate to introduce a new or related topic or if continuing with the current topic is what is required.

Once Topic Trainer assesses the patterns found in the Speech Act Log, the module will output whether or not the next turn's initial dialogue act topic will:

1) introduce new topic

2) continue the current topic

The Topic Handler then outputs its choice to Module 6: Reactor.

5.3.4 Module 6: Reactor

Reactor then takes the output from Module 5: Topic Trainer, retrieves an appropriate topic from the Topic Base and yields an appropriate response to the human's utterance. If the output from the Topic Trainer is to address the new topic (i.e. "1) introduce new topic" above), the Reactor searches the Topic Base for knowledge on an appropriate topic and respond accordingly (as author in [36] outines, in its next turn the NPC will either continue with the current topic, ask a question about a new topic, or remember an event). If the output from the Topic Trainer is option “2) continue the current topic” above, then the Reactor proposes to comment on the current topic.

5.4 Summary

The above Topic Handler model framework is again proposed for within the scope of the Conversation Handler model in [36]. As a test and proof of concept of this proposed Topic Hander model, in the following Chapters we utilize the SPAADIA corpus and create Module 4: Speech Act Log and Module 5: Topic Trainer to show how the needed topic handling predictions can be created for input into Module 6: the Reactor. For the scope of this thesis, we will focus on Module 4: Speech Act Log and Module 5: Topic Trainer as proof of concept, using the SPAADIA corpus as the output of Module 3: Speech Act Identifier, and assume all other modules work perfectly.

In Chapter 6 we begin by introducing the different experiments conducted in this study, describe the SPAADIA corpus in more detail as well as the additional feature engineering implemented to create the DA level datasets (as described in section 2.2.3) and the Turn level datasets (i.e. a DA level dataset where each sample includes features for one DA and a Turn dataset where each sample includes features for a speaker's full turn). The following Chapters describe in more detail each experimentation, implementation, results and analysis.

6 Feature Engineered Datasets for Application of Topic Handler model to Sample Dialogues

In the following Chapter we conduct two machine learning experiments to demonstrate how the utilization of speech act and semantic annotations (SAASA) data can improve topic handling in order to offer more realistic topic handling by virtual agents or NPCs (see Table 1 below). Before describing these experiments in more detail, we first summarize the experiments at a high level then go into more detail regarding the SPAADIA corpus of dialogues and how we use them as inputs into the machine learning models.

In our experimentation, we use the DA level training features listed in Table 7 and the Turn level features listed in Table 8. In Table 1, we see that the main difference between Experiments 1 and 2 is which dataset is used as input features (Experiment 1 uses the DA level dataset whereas Experiment 2 uses the Turn level dataset). The features in these datasets are used in order to create the SPAADIA derived Module 4: Speech Act Log and experimental iterations of Module 5: Topic Trainer of the proposed Topic Handler in order to offer proof of concept for the proposed framework. Going forward we will refer to Module 4: Speech Act Log as M4 and Module 5: Topic Trainer as M5.

Experiment	Classifier	Dataset
1	Binary XGBoost Classifier for predicting "Next DA New Topic" feature class	DA-Level
2	Binary XGBoost Classifier for predicting "Next DA New Topic" feature class	Turn level

Table 1: High-Level Experimentation Summary

In order to provide aforementioned proof of concept of the proposed Topic Handler in Figure 3, we conduct the experiments highlighted in Table 1 as representative of M5 in order to find the most optimal initial implementation requirements. Each experiment is based on binary classifiers predicting "Next DA New Topic" feature class described in Section 6.1.5. As previously mentioned in Experiment 1, the twelve DA features in Table 7 are used as input features to train a binary classifier that predicts the "Next DA New Topic" feature class in Table 6 and Experiment 2 utilizes the Turn level dataset in Table 8 to train binary classifiers that predict the same "Next DA New Topic" feature class. Within each of these experiments, five data splits will be tested to see which feature set yields the classifier most efficient at predicting the prediction label (see Table 2 for five data splits). We use DA Bag of Words (BoW) and Term Frequency - Inverse Document Frequency (TF-IDF) vectors as baseline feature inputs to train baseline binary classifiers and compare their performance to classifiers trained using SAASA features. We will go into further detail about our comparative baselines BoW and TF-IDF and performance metrics in the following sections.

Binary Classifier	Data Split	Dataset
Baseline 1	BoW of DA or Turn level Dataset	BoW feature vectors based on Dialogue Act/Feature 1 in Table 7
Baseline 2	TF-IDF of DA or Turn level level Dataset	TF-IDF feature vectors based on Dialogue Act/Feature 1 in Table
Topic Trainer 1	DA or Turn level Dataset SAASA features minus DA	Feature 2 - 11 listed Table 7
Topic Trainer 2	DA or Turn level Dataset SAASA features Plus BoW DA	Feature 2 - 11 listed Table 7 and BoW DA Vectors
Topic Trainer 3	DA or Turn level Dataset SAASA features Plus TF-IDF DA	Feature 2 - 11 listed Table 7 and TF-IDF DA Vectors

Table 2: Data Split per Experiment Summary

6.1 Experiment Input Dataset and Additional Feature Engineering

In this section we describe in detail the corpus and data engineering needed before implementing proof of concept M5. As mentioned above, we will offer proof of concept for our proposed Topic Handler model by creating Python based scripts to run experimental examples of the dialogue processing modules within the framework. We leverage an existing corpus annotated with semantic and pragmatic data as outputs of the Module 3: Speech Act Identifier (M3) and as the inputs into M4. In the below experiments, we create variations of M4 and M5 in which outputs of M4 are prepared corpora data, including additional feature engineering, in order to create input for machine learning model representing M5. Following each experiment we discuss the varying M5 performance in predicting classification labels (to be discussed in more detail below), highlight which variations of M4 and M5 perform most optimally for future work, and most importantly we demonstrate the better performance when using SAASA data. In the next sub-section we discuss in more detail the small corpus of dialogues used for the purposes of this thesis.

6.1.1 SPAAC Speech Act Annotation Scheme and Annotated Dialogue

In order to test the proposed model, we will be using 89 telephone dialogues from the Speech Act Annotated Corpus (i.e. SPAADIA, a subset of SPAAC) that were annotated using the DART v2 scheme [47]. These annotated dialogues are task-oriented service dialogues that were recorded via telephone and then transcribed. The dialogues are between customers and either Trainline or AMEX representatives (35 and 55 dialogues respectively) [48]. The SPAAC Annotation Scheme was originally applied to a British Telecom (BT) Oasis Corpus of 1,200 telephone dialogues, and the Trainline Corpus of 35 longer telephone dialogues, however due to customer information and security reasons BT no longer allows external access to the complete corpus [48]. We assume the DART v2 works perfectly and can therefore use the DART v2 annotated SPAADIA corpus as our M3 output. From this output's annotations, we build a M4 with the below described additional feature engineering.

The 89 telephone dialogues consist of 12,874 dialogue acts (in the SPAAC and DART literature these were referred to as C-units, but for consistency with other literature will refer to them as dialogue acts) and 6,129 turns for use in this proof of concept (see Table 9). In each of these 89 dialogues, there are two participants in each dialogue: A and B. The dialogue is divided into units called turns and each turn is attributed to a particular speaker [48]. A given turn starts when a speaker begins speaking, and ends when the same speaker stops speaking with the turn of another speaker intervening [48]. A turn can contain more than one dialogue act, but often a turn contains just one.

6.1.2 SPAADIA Dialogue Acts and DART Taxonomy v2

The original SPAAC was annotated manually by hand and SPAADIA v1 is a subset of SPAAC. SPAADIA v2 however was annotated using version two of the Dialogue Annotation and Research Tool (DART) [50]. As mentioned in Section 2.2.3, going forward when we refer to SPAADIA we are referring to SPAADIA v2. V2 of DART supports fine-grained basic taxonomy for more than a hundred and twenty basic categories and more potential combinations, distinguishing between different types of speech acts as realized through and in different dialogue act types, the sequencing of units in dialogue, the influence of modality, polarity, etc. In comparison to v1 of DART, it also features a more robust grammar for recognizing different syntactic types, a larger inventory of IFIDs/mode labels, and an improved inferencing mechanism for deducing speech acts, all based on symbolic, rather than probabilistic identification strategies [50]. Speech act form and labels are discussed in further detail below.

6.1.3 SPAADIA Dialogue Acts Annotations

As discussed in Section 6.1.6, within the realm of SPAAC and DART annotations, an utterance can contain more than one dialogue act, but usually contains only one dialogue act [28]. Dialogue acts (or C-units as they are referred to in [50, 28] and can be defined as independent syntactic units of spoken English grammar) are assigned speech-act attributes and other attributes. Utterances are not given separate attributes, and so the original SPAADIA annotation scheme pays attention only to dialogue acts. The segmentation of a dialogue into turns is a 'given'/provided by the transcription. On the other hand, the segmentation of a turn into dialogue acts is part of the SPAADIA annotations [28]. It was accomplished automatically by a splitting routine, which took into account pauses and other phenomena [28].

An utterance cannot contain less than a dialogue act and if an utterance contains more than one dialogue act, then one of the dialogue acts will be just a discourse marker (e.g. 'well' or 'now' or 'okay'). Therefore the structure of an utterance is a dialogue act plus optional discourse markers associated with it [28].

Where anomalies occurred in the original transcription, the annotators did not attempt to correct or otherwise change the transcription. The transcriptions were however changed in respect to anonymisation [28]. During this procedure there were randomized changes to personal information such as names, addresses, and credit-card numbers, as well as some other names in the dialogue. One result is that there is sometimes a lack of match between numbers and letters that would have been identical in the original transcript (e.g. an 'echo' reply, in which speaker x repeats a name or number mentioned by speaker y, may no longer appear to be repetition) [28].

6.1.4 SPAADIA Dialogue Acts and DART v2 Annotations

A dialogue act (DA) is the basic unit for speech act annotation. Syntactically, it is an independent clausal or a non-clausal unit (SPAADIA a non-clausal unit is labelled as a 'fragment' - frag). Functionally, it represents a unit which can be assigned to a given communicative function, represented by its speech act attribute [28].

DAs are the basic units (or envelopes) for conveying speech acts (also called 'moves' or 'c-units'), which can be regarded as the minimal communicative actions performed in a dialogue. DAs are initially classified in terms of their grammatical form (syntactic categories) using one of the following form labels found in Table 3:

Speech Act Form	Description
decl	declarative clause
q-yn	yes-no question
q-wh	wh-question
address	terms of address
imp	imperative
dm	discourse marker
exclam	exclamations
yes	affirmative reply
no	negative reply
frag	fragment, i.e. syntactically ungrammatical or incomplete/elliptical syntactic units

Table 3: DART v2 Speech Act Forms [49]

Of these, **yes** and **no** could be treated as special cases of discourse markers however because of their significance in dialogues, they are places in distinct categories [48].

In addition to the Speech Act form, DAs are also annotated with attributes relating to their syntactic elements however not all DAs require these annotations as they are not always relevant. In Table 4 we see all possible syntactic element attribute annotation categories.

Syntactic Element Attribute	Description
n	the number of the relevant unit/dialogue act or turn in conversation, numbered consecutively and independently of turns
sp-act	the speech act or speech-combination identified by DART v2; See Appendix for full list of 120 DART v2 speech act labels; a dialogue act can be labelled as a combination of speech act labels
polarity	surface polarity (sometimes null)
topic	semantic information pertaining to what is being talked about
mode	semantico-pragmatic (interpersonal) information indicating the form of interaction [49]; sometimes null

Table 4: Additional DART v2 Syntactic Element Attributes [49]

The above annotations found in Tables 3 and 4 are added to each sample dialogue in an XML file by DART. In Figure 5 below we see the basic DART XML format structure. In Figure 6 is an example of an annotated dialogue act from SPAADIA. Here we see the a new turn container initialized with the turn count (**n** = "6") and speaker (**speaker** = "caller") identified. Following this the first dialogue act is denoted with the form container and annotated as form "dm" (i.e. discourse marker) and is annotated as with its relevant count (**n** = "17") and as speech act label "hesitate" (i.e. sp-act = "hesitate"). Following these annotations, the dialogue act itself is denoted: "uh" and then the form "dm" container is closed.

In the same turn, we see the next dialogue act is a fragment and is therefore in a "frag" container. The dialogue act count is **n** = "18", speech act label is a compound label (**sp-act** = "answer-refer"), **polarity** is annotated as "positive", **topic** is identified "to-location-US" as and **mode** is annotated as "partial". Finally the dialogue output is shared as "to Newark" before the fragment container is closed.

Using this format, XML files of each of the eighty-nine SPAADIA dialogues we have are input into the M4 Python script, which outputs Speech Act Log in the form of a CSV file for each dialogue. While the Python script for M4 reads in each for the dialogue XML files, additional features are derivable for each dialogue act before adding the DA to its respective conversation Speech Act Log CSV file. In the next section, we outline the the additional features and how they are added into each CSV Speech Act Log file.

Format of the output CSV file will depend upon which experiment the M4

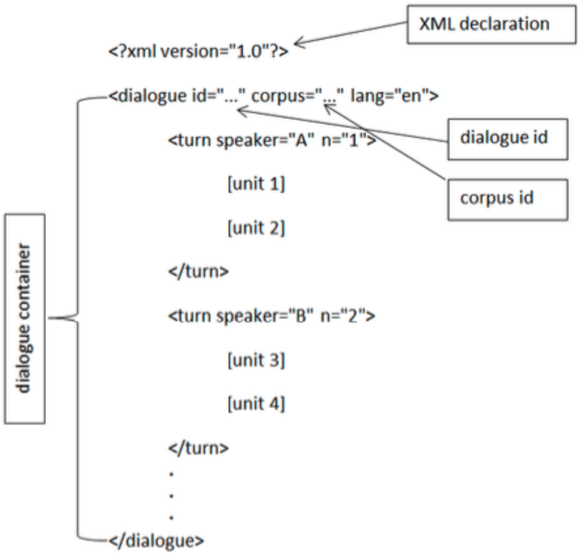


Figure 5: The basic DART XML format [49]

```
<turn n="6" speaker="caller">
<dm n="17" sp-act="hesitate">
uh
</dm>
<frag n="18" sp-act="answer-refer" polarity="positive" topic="to-location_US" mode="partial">
to Newark
</frag>
```

Figure 6: Sample Dialogue Acts from SRI Amex Conversation 10b

was being used for (predicting based on full turn patterns or individual DA). We will outline the difference in the following Chapters as we describe each machine learning experiment.

6.1.5 Additional Feature Engineering for Speech Act Log Module

In order to properly simulate a M3 and M4, additional feature engineering was done. These dialogues have been marked up in the DART v2 scheme in XML files and each dialogue has the same feature categories per DA (including identification of current topic or sub-topic). A Python script creates a Speech Act Log class and logs each dialogue into a CSV file. Each row refers to a DA and each column refers to one of the features outlined in Section 6.1.6.

In addition to the features provided by the DART annotations in the SPAADIA corpus, the features in Tables 5 and 6 were derived from the existing annotations in order to aid in training M5:

Derived Features	Description
Speech act in turn count	The in-turn dialogue act count
Topic count	The count everytime a topic is introduced (whether or not it has been previously discussed). While not every dialogue act was annotated for a topic, we assumed that if a dialogue act was not annotated with a topic, that is because it is continuing the current topic and no new topic was introduced.
Combination Speech Act (y/n flag)	Flag identifying if the dialogue act is a combination of 2 or more of the 120 speech act labels in the DART v2 annotation scheme.
Topic *expanded*	This is not an additional feature, just an expansion of the topic annotation. While not every dialogue act was annotated for a topic, we assumed that if a dialogue act was not annotated with a topic, that is because it is continuing the current topic. Therefore each dialogue act is annotated with a topic based on this assumption.

Table 5: Derived Features Set 1

The final label, as described in Section 5.3.3, is the feature whose classification is to be predicted by M5 ("Next DA New Topic" as defined below in Table 6). For each DA, we include the following flag in terms of the next DA as we want to use features of current DA or turn in order to predict if the next DA addresses a new topic or continues the current topic. This will allow of ease of using these labels for training, testing and validating. If the output is "no/0" for the prediction feature in Table 6 below, then output into the Module 6: Reactor (M6) will be to continue the current topic.

Prediction Feature	Description
Next DA New Topic (y/n flag)	A topic is only annotated yes/1 when a new topic is introduced that differs from the previous dialogue act's topic.

Table 6: Derived Prediction Feature / Training Label

6.1.6 SPAAC Dialogue Sample with Final Annotations for input in Speech Act Log

See below for a sample dialogue act including all fourteen features from trainline35.xml conversation in SPAADIA (see Appendix Section 9.1 for full trainline35.xml conversation):

1) Dialogue Act: "I do"

2) Speaker: B

3) Polarity: positive

4) Mode: decl

5) Turn Count: 6

6) Speech Act Count: 16

7) In-Turn Speech Act Count: 16

8) Speech Act Form: decl

9) Speech Act Label: answer-state

10) Topic: creditcard

11) Topic Counter: 7

12) New Topic: 0

13) Combination Speech Act Label: 1

Prediction Label:

14) Next DA New Topic: 1

In the following experiments various binary classifiers are trained to predict the "Next DA New Topic" feature class. The classifiers are trained using DA level or the Turn level dataset. We will go into more detail when introducing each experiment in the next Chapter. The results of the model's ability to predict on test data is compared to classier prediction performance when trained using speech act bag of words (BoW) and Term Frequency - Inverse Document Frequency (TF-IDF) baselines. The selected features of each DA are considered when training each model in order to predict one label. In M4, twelve of the above features are stored (we do not consider **Speaker** listed above as a feature) and available for input into M5.

In the next Chapter's experimentations, we start with testing which features and input feature data splits provide the more effective input into M5 for predicting the "Next DA New Topic" feature class, to maximize our demonstration that Speech Act and related-semantic and pragmatic data can aid in the creation of more cohesive human-computer dialogues. Evaluations for feature importance will not be included in the scope of this thesis, we will instead focus on the varying dataset splits and the impact of this on the overall performance to demonstrate value of using SAASA data for topic handling.

6.1.7 Summary: Using SPAADIA Annotations and Feature Engineered Datasets

In combining the annotations form the SPAADIA and pairing it with additional feature engineering, we have twelve training features, and one topic-related feature for prediction, for each DA. Using the derived topic y/n flag relating to topic changes or continuation (described in Table 6 above), we can use the training features (will not use **Speaker** as an input feature) and test label when using machine learning methods as the M5 for a proof of concept of the overall Topic Handler model framework.

In the next section, we begin by describing in more detail the DA level and Turn level datasets followed by the performance metrics we use to assess the performance of each classifier.

6.2 Dialogue Act Level vs Turn Level Datasets

6.2.1 Dialogue Act Level Dataset for Experiment 1

For each DA in the SPAADIA corpus, there is a total of thirteen features however for our experimentation Speaker is not included, reducing total number of training features from thirteen to twelve (see Table 7 for list of sample of features for on DA in trainline35.xml conversation in SPAADIA). This set of features include those described feature engineering in Section 6.1.5 and are used in Experiment 1 for DA level binary classification.

	DA Input Feature	Example of DA
1	DA	"I do"
2	Polarity	positive
3	Mode	decl
4	Turn Count	6
5	Speech Act Count	16
6	In-Turn Speech Act Count	16
7	Speech Act Form	decl
8	Speech Act Label	answer-state
9	Topic	creditcard
10	Topic Counter	7
11	New Topic	0
12	Combination Speech Act Label	1

Table 7: Training Features in Dialogue Act Level Dataset

6.2.2 Turn Level Dataset for Experiment 2

As mentioned above, the input would be sequences found in-turn rather than the feature vector for one SPAADIA DA used in Experiment 1. In order to get the Turn level dataset, additional feature engineering is done. In an updated version of M4, the data is stored at the Turn level. In Table 8 below is an example of a resulting input Turn level features with three DAs in turn (from amex20a.xml dialogue in SPAADIA corpus).

Because we are looking at the data at the Turn level, we are significantly reducing the training and test samples to just 6,129 sample turns (versus the 12,874 samples in the DA level dataset). Predictions will be made based on the sequence of each feature for one turn (i.e. given the sequence of the other speaker's turn, in this experiment we will use the full turn's sequence data to predict the one topic-related feature described above, simulating the output of a binary classification M5. For input into each experiment's classifier, the in-turn sequence features (e.g. In-Turn Speech Act Form Sequences (ITFS) in Table 8: "dm q-wh decl") are turned into 2D binary vectors.

	In-Turn Input Feature	Example of one Turn with 3 DAs
1	In-Turn Dialogue Acts *In-Turn Dialogue Act 1:* "ok" *In-Turn Dialogue Act 2:* "what about a a change of planes or a quick stop-over?" *In-Turn Dialogue Act 3:* "or one stop maybe in either Portland or Seattle and continuing on down"	**In-Turn DA:** "ok what about a a change of planes or a quick stop over or one stop maybe in either Portland or Seattle and continuing on down"
2	Turn Count	37
3	In-Turn Speech Act Form Sequence (ITFS)	dm q-wh decl
4	In-Turn Speech Act Label Sequence (ITLS)	acknowledge suggest reqInfo
5	In-Turn Polarity Sequence (ITPS)	None positive positive
6	In-Turn Mode Sequence (ITMS)	tag alternative-closed-query alternative-opinion-query

Table 8: Training Features in Turn Level Dataset

For Experiment 2, each in-turn sequence feature is turned into a binary or TF-IDF vector where the presence of each possible feature value is present however the sequence order is not maintained. In the Turn level dataset we only focus on sequences of six of the twelve features used in Experiment 1 as certain features no longer make sense to include at the turn level (e.g. In-Turn Speech Act Count). These binary vectors are created using the Sci-Kit Learn libraries. For example, if we take the in-turn data found in Table 8, a binary vector is created where each column represents a different form in the speech act form vocabulary and the length of the ITFS vector is dictated by the length of the speech act form vocabulary. If all possible speech act forms found in Table 3 are in the training dataset then the ITFS vector would be ten in length. Because the example in Table 8 has the following 3 speech act forms: "dm q-wh decl", in the columns representing each of these three forms there would be a one and for the rest of the seven columns a 0. We will revisit this again in Section 7.6 when we delve further into the Experiment 2 implementation.

In the next section, we begin our focus on Experiment 1 by discussing metrics we use to compare the experiment classifiers. In Section 7.1 we first consider what are the best metrics for measuring comparative success and cross-validating the Experiment 1 and 2 binary classification machine learning models.

7 Proof of Concept Experiments for Topic Handler Model

In the following experiments, we leverage the small SPAADIA corpus to demonstrate positive impact consideration of SAASA data can have on conducting cohesive topic handling by an NPC or chatbot during human-computer dialogues. We use the eXtreme Gradient Boosting (XGBoost) algorithm in Experiments 1 and 2 for create binary classifiers for predicting if the subsequent DA would address a new topic in a natural and cohesive dialogue.

In the next section, we first review the success metrics used to measure the performance of each experiment classifier. As we are dealing with an imbalanced datasets, we will focus on metrics outside of accuracy.

In Section 7.2 we highlight the cross validation methodology used throughout experimentation and describe how we leverage learning curves to describe the potential impact of additional data on each experiment classifier.

In Section 7.3, we describe the different baselines we use to demonstrate the impact of using SAASA data for predicting if the next DA will address a new topic. In Section 7.4, we highlight in detail the XGBoost algorithms used in Experiments 1 and 2 and touch on why other popular NLP machine learning algorithms such as Bidirectional Encoder Representation from Transformers (BERT) are not used in this experiment.

And finally, in Sections 7.5 and 7.6, we discuss in detail the different classifiers implemented, present the pre- and post-optimization results for each classifier, compare classifier performance and describe the implications of these findings.

7.1 Success Metrics for Imbalanced Binary Classification Experiments

The evaluation of a model's prediction performance is of great importance when judging its usefulness as well as when comparing competing methods [40]. As we will be comparing model performance in the following experiments, a significant consideration we need to take into account is that we will be using the annotated SPAADIA v2 corpus which is a relatively imbalanced dataset in terms of the "Next DA New Topic" class distribution. In the current context we are dealing with an imbalanced binary classification problem: a negative case with the majority of the examples and a positive case with a minority of examples. As mentioned above we have 12,874 DAs to train and test. Of these DAs just over 4,167 DAs are positive for the "Next DA New Topic" feature whereas 8,707 DAs (more than double the positive instances) are negative for "Next DA New Topic" feature (see Table 9 below). This is also the case when we look at the Turn level (can contain multiple DAs or one DA) and consider if the first DA following this turn is introducing a new topic (i.e. "Next DA New Topic" feature for turn is positive). Of the 6,129 turns, only 1,330 are positive for "Next DA New Topic" feature. In both the DA level dataset and Turn level dataset cases, we then encounter the accuracy paradox: "high accuracy is not necessarily an indicator of high classifier performance" [45]. For instance, in a predictive classification setting, predictive models with a given (lower) level of accuracy may have greater predictive power than models with higher accuracy [45]. In particular, if a single class contains most of the data, a majority classifier that assigns all input cases to this majority class would produce an accurate result [45]. For example, if we created a model that always predicted negative for the "Next DA New Topic" feature, we would get 67.6% accuracy when predicting the "Next DA New Topic" class for each of the DAs in the DA level dataset and 78.3% accuracy when predicting the "Next DA New Topic" class for each of the the turns the SPAADIA v2 corpus. Highly imbalanced or skewed training data such as ours is a commonly encountered phenomena. The classes' distributions of the samples do not necessarily reflect the distributions in the whole population since most of the time the samples are gathered in very controlled conditions. The imbalance therefore hinders the capability of the statistical model to predict the behaviour of the phenomena being modelled [45].

Knowing this, we cannot rely on accuracy and error rate for the evaluation of performance within the following experiments and will need to find alternative metrics for measuring success. In the next section we will discuss such alternative metrics. It is important to note that there are various methods for overcoming an imbalanced binary classification problem (e.g. undersampling, synthetic minority oversampling technique (SMOTE), adaptive synthetic sampling method (ADASYN) [10] random undersampling techniques with bagging or boosting ensembles [21]) however we will not dive into using additional methods for model optimization. As aim of the following experiments is to demonstrate how SAASA features allow for improved prediction of "Next DA New Topic" feature class, not to fit a model for continued use and future predictions, we will focus on selecting the best success metrics for demonstrating its improvement upon the baseline trained predictive models.

Dialogue Act and Turn Dataset Distribution		
Dialogue Act Dataset Distribution		
“Next DA New Topic” feature = 1	4,167	32.37 %
“Next DA New Topic” feature = 0	8,707	67.63 %
Total DA Count	12,874	-
Turn Feature Dataset Distribution		
“Next DA New Topic” feature = 1	1,330	21.70 %
“Next DA New Topic” feature = 0	4,799	78.30 %
Total DA Count	6,129	-

Table 9: Dialogue Act Dataset Distribution

7.1.1 Confusion Matrix, Precision, Recall and F1 score

Before jumping into other potential metrics for our imbalanced classification problem, it is relevant to first discuss accuracy from the perspective of a confusion matrix. A confusion matrix is a table that summarizes how successful a classification model is at predicting examples belonging to various classes (see Table 10 [10]). One axis of the confusion matrix is the label that the model predicted, and the other axis is the actual label [10]. For a binary classification problem, we can create a confusion matrix that highlights each class, the instances that were correctly classified (i.e. number of True Positives (TP) and True Negatives (TN) and the instances that were wrongly classified (i.e. the number of False Positives (FP) and False Negatives (FN) [8].

In Table 10, we have a confusion matrix for a binary classification model where the two classes are “spam” and “not spam” (for a multi-class classification or a multi-label classification, the confusion matrix would have as many rows and columns as there are different classes). It shows that of the twenty-four examples that were spam, the model correctly classified twenty-three as spam. In this case we say that we have twenty-three TPs or TP = 23 [10]. The model incorrectly classified 1 example as “not spam”. In this case we have one FN (or FN = 1). Similarly, of the five hundred and sixty-eight examples that are actually not spam, five hundred and fifty-six were correctly classified as TN = 556 and twelve were incorrectly classified (FP = 12). Therefore, a confusion matrix can help determine mistake patterns and can be used to calculate two performance metrics [10].

Example Confusion Matrix for Spam Classification		
	spam (predicted)	not spam (predicted)
spam (actual)	23 (TP)	1 (FN)
not spam (actual)	12 (FP)	556 (TN)

Table 10: Example Confusion Matrix [10]

Based on the metric highlighted in a confusion matrix, accuracy can be defined as follows:

accuracy = (TP + TN) / (TP + FN + TN + FP)

As discussed above, accuracy is not suitable because of the impact of the least represented, but more important examples (where "Next DA New Topic" class is positive), is reduced when compared to that of the majority class (examples of "Next DA New Topic" class is negative) [8]. Therefore the metrics we use in imbalanced classification problems must consider the user preference bias towards the minority / positive class examples and should take into account the data distribution. From the confusion matrix there are numerous alternate yet commonly calculated and used metrics[8].

precision = TP / (TP + FP)

recall = TP / (TP + FN)

specificity = TN / (TN + FP)

false positive rate = FP / (TN + FP)

false negative rate = FN / (TP + FN)

negative predictive value = TN / (TN + FN)

Precision and recall (otherwise know as sensitivity or true positive rate) are two of the most frequently used metrics to assess a predictive model [10].

From the above calculation we can see that precision is the ratio of correct positive predictions to the the overall number of positive predictions [10]. For our below experiments, precision can be useful in understanding how often a trained model is introducing a topic at an appropriate time and compare the predictive power when training the model using baseline BoW data versus SAASA data (i.e. using precision we can show how often our below models are correctly identifying opportunities to introduce a new topic; alternatively what percent of introduced new topics are raised at an appropriate time). As we discussed in earlier Chapters, virtual agents introducing new topics at unexpected points in a dialogue is an issue preventing them from continuing longer cohesive dialogues.

Recall is the ratio of correct positive predictions to the overall number of positive examples in the dataset. In our below experiments, this can help us measure a trained model's efficiency at identifying points in which it is appropriate to introduce a new topic and how often these opportunities are missed when the model has been trained on baseline BoW data vs SAASA data (i.e. what percent of opportunities to introduce a new topic are missed by the model). Therefore in tracking precision and recall in our below experiments, we can see how much better a model is at identifying when a new topic can be introduced and demonstrate that SAASA data improves our ability to identify such opportunities for more cohesive and realistic human-virtual agent dialogues.

Given the calculations of precision versus recall, as one increases the other decreases. In practice, it is often that we have to choose between a high precision or a high recall and one is chosen based on each specific problem. In the case of the spam detection problem highlighted by Table 10, we would want to have high precision as we would want to avoid making mistakes in which we detect any legitimate messages as spam (and potentially putting important non-spam messages in the spam folder) [10]. We would be ready to tolerate a lower recall and having some spam messages in our inbox [10]. In the case of our below experiments, we would also ideally have a higher precision which would remove instances in which we have point in a dialogue where a virtual agent flouts conversation norms and introduces a new topic at an inappropriate point. However, we would not want such a low recall that a virtual agent rarely identifies points where a new topic could be introduced naturally (however as in the case of spam classifications, this would be more tolerable than lower precision). Therefore we can also use the F1 score to track success in the following experiments. F1 score (also known as F-measure) represents the harmonic mean of precision and recall:

$$F1 = 2 * (precision * recall) / (precision + recall) \text{ [43]}$$

The value of F1 score is ranged from zero to one, and high values of F-measure indicate high classification performance [43]. A good F1 score can vary based on different balance classification problems. As our aim is to demonstrate that SAASA data can help predict the "Next DA New Topic" class by improving a model's performance we will also be using F1 score as a comparative measure to a model trained on baseline data. Therefore in the following experiments, we will be comparing models trained on baseline data (BoW and TF-IDF vectors) vs SAASA data using precision, recall, and F1 score, with focus on optimizing precision.

7.1.2 Precision-Recall Curve and Area Under the Curve (AUC-PR)

In addition to the use of the above metrics for understanding the comparative performance of our models, precision-recall curve plots (PRC) can provide the viewer with an accurate prediction of future classification performance when dealing with imbalanced classification scenarios [40]. This is due to the fact that they evaluate the fraction of true positives among positive predictions. Commonly used measures of classifier performance in the phase of model construction are Area under the Receiver Operating Characteristics (ROC) curve (AUC) in addition to accuracy and error rate, sensitivity and specificity [40]. However, just like accuracy and error rate, ROC is ideal when dealing with roughly equal balance data sets but is not ideal when working with a class imbalanced dataset. The ROC can be misleading when applied in imbalanced classification scenarios [40]. Therefore we will focus on AUC-PR (Area under the Precision-Recall Curve), and not ROC AUC, in our evaluations below. When cross validating performance using AUC-PR in Experiment 1, we train multiple classifiers for the binary classification problem and get performance results for the baseline dataset performance vs the SAASA feature dataset performance at the default thresholds. Additionally, in Experiment 1 we also compare results for each classifier's optimal threshold for maximizing F1 score. We maximize for F1 score and not precision as often the maximum precision is where recall is zero, and as mentioned above while we focus on improving precision score we do not want to allow recall to decrease too much.

AUC-PR, as well as ROC AUC, is measured between 0 and 1 and the higher the AUC the better the classifier [10]. For ROC AUC used on a dataset that is balanced for each class, a classifier with an ROC AUC higher than 0.5 is better than a random classifier, if ROC AUC is lower than 0.5, then something is wrong with your model. A perfect classifier would have an AUC of 1 [10]. The value of AUC-PR for a random classifier is equal to the ratio of positive samples in a dataset with respect to all samples (a precision that is the ratio of positive examples in the dataset). In our examples, this random classifier would therefore achieve an AUC-PR of 0.3237 for DA level dataset and 0.2170 for Turn level dataset, therefore any AUC-PR we achieve above these respective values in our below experiments would indicate that the model performs better than a random classifier.

Additionally for ROC AUC, if your model behaves well, you obtain a good classifier by selecting the threshold values that gives recall close to 1 while keeping false positive rate near 0 [10]. For the AUC-PR, the model will aim have recall close to 1 as well as precision close to 1. We will outline the overall AUC-PR in each experiment following cross-validation to highlight model fit for predicting. In section 7.2 we'll discuss our chosen methods for cross-validating our experiments in more detail.

7.1.3 Best Threshold for Optimal F1 score and AUC-PR

As we are working with a class imbalanced data set it is important to stratify to ensure there are instances of the minority / positive class in each set. In the resulting AUC-PR curves output in each experiment, we include a line for "No Skill" to represent the "curve" for a random classifier as well as a marker for the threshold that achieves the best F1 score (from this we can see the highest possible precision and recall when harmonized into highest possible F1 score for model).

7.1.4 Success Metrics Summary for Binary Classification

When measuring success and comparing competing methods in the below experiments, we will focus on precision, recall, F1 score, and AUC-PR with particular focus on precision. We will look to see how baseline and SAASA based methods compare and which scores higher across all metrics with the hypothesis that the SAASA trained models will offer higher scores in all 4 metrics.

As precision or recall increases the other decreases which is why we will also compare F1 scores as well as it demonstrates harmonization between precision and recall. However, as with the above discussed spam example and based on discussions in earlier chapters regarding human-chatbot interaction we will aim to optimize for higher precision over recall, tracking F1 will help allow us to still gauge scores with a reasonable recall value (i.e. not highest precision where recall could equal 0). While the avoidance of as many FN predictions as possible is desirable (which a higher recall score would measure) we more so prefer to avoid FP predictions where a virtual agent would prompt a new topic at inappropriate times (i.e. would prefer a virtual agent that introduces a new topic less frequently but when it does more often introduces at appropriate times, and we are ok with a slightly high amount of misses "new topic" introduction opportunities). Therefore we will also aim to get highest F1 score possible where there is balance but also ensure that the balance of precision versus recall is not skewed towards a significantly higher recall score as we would more likely prefer the reverse in this context.

7.2 Cross Validation Methodology

7.2.1 Train, Validation and Split Datasets

Post processing by M4 but pre-implementation of each experiment the dataset is split into train, validation and test sets. This is done so using the the Python sci-kit learn *train_test_split()* function [32]. By using the random_state parameters and consistently setting it to 14 we get the same shuffle before splitting to keep shuffle split consistent across all experiments post M4 preparation for input into M5. The test is 15% of the original data size and validation set is then 15% of the remaining samples in the training set. The data is not explicitly stratified across each test set to allow for more realistic input test input, however the distribution of labels is similar across each dataset (see Tables 11 and 12 below).

Train, Validation and Test Dataset Label Distribution		
Dialogue Act Training Dataset Distribution		
"Next DA New Topic" feature = 1	2,892	32.10 %
"Next DA New Topic" feature = 0	6,118	67.90%
Total DA Count	9,010	-
Dialogue Act Validation Dataset Distribution		
"Next DA New Topic" feature = 1	656	33.95%
"Next DA New Topic" feature = 0	1,276	66.05%
Total DA Count	1,932	-
Dialogue Act Test Dataset Distribution		
"Next DA New Topic" feature = 1	619	32.04%
"Next DA New Topic" feature = 0	1,313	67.96%
Total DA Count	1,932	-

Table 11: Dialogue Act Training, Validation and Test Dataset Label Distribution

Train, Validation and Test Dataset Label Distribution		
Turn Feature Training Dataset Distribution		
“Next DA New Topic” feature = 1	919	21.43%
“Next DA New Topic” feature = 0	3,370	78.57%
Total Turn Count	4,289	-
Turn Feature Validation Dataset Distribution		
“Next DA New Topic” feature = 1	191	20.76%
“Next DA New Topic” feature = 0	729	79.24%
Total Turn Count	920	-
Turn Feature Test Dataset Distribution		
“Next DA New Topic” feature = 1	220	23.91%
“Next DA New Topic” feature = 0	700	76.09%
Total Turn Count	920	-

Table 12: Turn Feature Training, Validation and Test Dataset Label Distribution

7.2.2 GridSearch Cross-Validation using Stratified K-Fold Validation for XG-Boost in 1 and 2

In Experiment 1 (and 2), we use the Python sci-kit learn model_selection function *GridSearchCV* function [32]. This function allows for cross-validation of our model using different hyperparameter inputs while optimizing for a specific set of metrics. The implemented grid search CV will be set to evaluate performance based on multiple scores: accuracy, precision, recall and F1 score and ultimately the best_params_ (best parameters output) are the parameters needed to refit the estimator to maximize the F1 score (this will be considering the best estimator based on the grid search cross validation) [32].

For each method we train and test using 5 stratified folds ($k = 5$) with the fold splits. In K-fold validation, the dataset is randomly split in K mutually exclusive subsets (the folds) of approximately equal size [27]. For each partition i, the model is trained on the remaining K - 1 partitions, and is evaluated on partition i. In stratified K-fold validation, the folds are stratified so that they contain approximately the same proportions of labels as the original dataset [27]. As we are working with a class imbalance data set important to stratify to ensure there are instances of the minority / positive class in each fold. The final score is then the averages of the K scores obtained. In the below experiments scores are the resulting mean precision, recall, and F1 score.

An alternate option is stratified K-fold validation with shuffling consists of applying stratified K-fold validation multiple times, shuffling the data every time before splitting it K ways. This training results in evaluating $P\ x\ K$ models (where P is the number of iterations you use, and can be very expensive). The final score is the average of scores obtained at each run of K-fold validation [14]. Note that you end up training and evaluating P x K models (where P is the number of iterations you use). This is useful for when you have relatively little data available and you need to evaluate your model as precisely as possible. This method has been noted to be extremely helpful in competitions with small amounts of data [14]. While we are dealing with relatively small DA level and Turn level datasets our aim for these experiments is to demonstrate the value of including and/or prioritizing SAASA data for topic handling and not optimizing to create a scalable model for a competition or immediate productionalization, we will not take this additional and expensive shuffling step with P iterations.

For XGBoost gridsearchCV, will focus on optimizing F1 score by trying various values for the three principal hyperparameters for gradient boosting: number of trees, learning rate, and maximum depth (see Table 13 below for values used in cross-validation of XGBoost based experiments). In section 7.4.3 we go into more detail what they hyperparameters are and their impact on the XGBoost model. In Appendix see Tables 23 and 27 for Experiment 1 and 2 parameters that achieved the best F1 score during cross-validation for each classifier in each experiment.

Hyperparameter	Test Values		
number of trees	50	100	200
learning rate	0.01	0.03	0.1
maximum depth	3	6	12

Table 13: Grid Search Cross-Validation Hyperparameters for XGB Classifiers

7.2.3 Learning Curves for Precision and F1 Score

Learning curves help determine cross-validated training and test scores for different training set sizes. Learning curves help by indicating how much we would benefit from adding more training data and whether the estimator suffers more from a variance error or a bias error [32]. Using Python sci-kit learn model_selection *learning_curve* function, we will use *StratifiedKfold* cross-validation and plot the training versus test precision scores and F1 scores in a learning curve. This is also done using the entire SPAADIA dataset (returning to total corpora size before create train, val and test splits) the *StratifiedKfold* cross-validation generator splits the whole dataset k times in training and test data. The subsets of the training set with varying sizes are used to train the estimator and a score for each training subset size and the test set will be computed. Afterwards the scores are averaged over all k runs for each training subset size [32].

From the learning curve outputs of each experiment, we can understand if the method would benefit from more data, whether the BoW or SAASA dataset would require less or more data relative to each other in order to generalize better, and if the methods trained on each dataset suffers from variance error or bias error. Variance error and bias error is important to consider in this context as more flexible statistical methods generally have higher variance and result in less bias [25]. As the curves are plotted with the mean scores, variability is shown with shaded areas representing the standard deviation above and below the training and cross-validation score lines. If the model suffers from bias error, then there would likely be more variability around the training score curve however if the model suffers from variance error, there would likely be more variability around the cross validation score line. Therefore in a learning curve we would prefer to see more flexible methods with higher variance and lower bias (i.e. larger standard deviation around the cross-validation score versus the training score).

For example, in learning curve plots found in Figure 6 we see that for the naive Bayes learning curve on the left that both the validation score and the training score converge to a value that is quite low with increasing size of the training set therefore we probably would not benefit much from more training data [32]. The training score starts higher at the beginning and decreases and the cross-validation score is very low at the beginning and increases, this shape of the curve can be found in more complex datasets very often [32]. Additionally the variability around the training score curve is as varied as the cross-validation score implying a bias error. In contrast for small amounts of data, the SVM training score in the right learning curve is much greater than the validation score and it appears that adding more training samples will most likely improve and increase the method's generalization [32]. Additionally the variability is emphasized on the cross-validation score alone indicating variance error and not bias error.

We will do a similar analysis for each machine learning method used in the below experiments and outline based on the learning curve the impact and efficiency of using BoW versus SAASA datasets for predicting the "Next DA New Topic?" feature class.

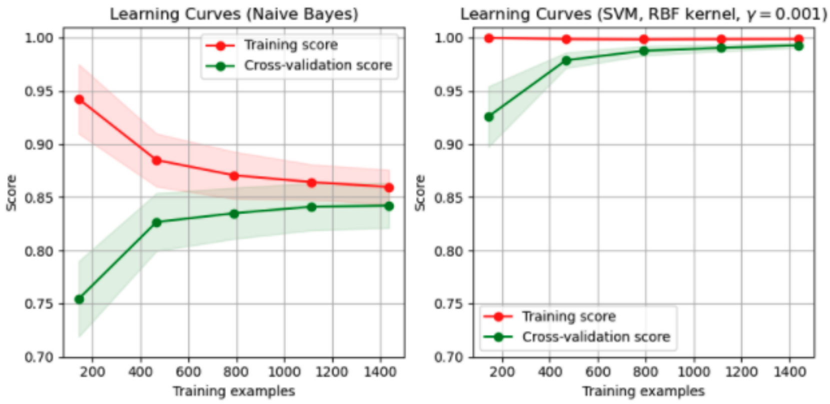


Figure 7: Example Learning Curves for Naive Bayes and SVM, Radial Basis Function Kernel [32]

7.3 Baselines for Experiments

For each of the below experiments, we will be using different baselines in addition to tracking the random classifier AUC-PR versus each experiment classifier. As mentioned above we will using AUC-PR to highlight classifier performance, with the additional performance comparison to a random classifier AUC-PR of 0.3237 for DA level SAASA data in Experiment 1 and 0.2170 for Turn level SAASA data in Experiment 2.

7.3.1 Experiment Baseline 1: Bag of Words

In order to test and demonstrate the improved performance model trained to predict "New DA New Topic" class using SAASA data, we use BoW as a baseline. BoW is a common way to convert a text into a feature vector [10]. In Experiment 1, where we look at DA level and Turn level data, we first take all DA feature texts in the training dataset, tokenize for each unique word (excluding stop words) and create a BoW vector for length 1,398 (1,563 words if we include stopwords). For each DA's BoW vector, the one-hot encoding method is used at the word level. The feature at position represents one of the 1,398 different words in the SPAADIA individual DAs and if the current DA has the specific word represented at this feature position it is 1, otherwise the feature is equal to 0 [10]. The "stopwords" English dictionary from the Sklearn feature extraction text Python package was used to remove commonly used words (such as "the", "a", "an", "in" that arguably do not carry as much semantic significance but whose frequency would impact results).

It is important to note that BoW is not an order-preserving tokenization

$$W(d, t) = TF(d, t) * \log \frac{N}{df(t)}$$

Figure 8: Calculation for TF-IDF of a Word [3]

method [14]. The tokens are generated and understood as a set, and not a sequence, and the general structure of the sentences is lost, therefore BoW tends to be used in shallow language-processing models rather than methods such as deep-learning models [14]. We will leave the use of sequence of character and words inputs for learning topic for a different study. Once we create a dataset with BoW vectors and the corresponding "Next DA New Topic" feature label it is included in a Pandas dataframe.

7.3.2 Experiment Baseline 2: Word TF-IDF

TF-IDF, like BoW, is a frequency-based text representation [22] and neither the order of the tokens (in this case words), or the semantics of the words, are captured. Inverse Document Frequency (IDF) was proposed as a method in conjunction with term frequency (TF) in order to lessen the effect of implicitly common words in a corpus [3]. IDF assign a higher weight to words (or tokens) with either high or low frequencies term in the document. The TF-IDF combination of TF and IDF is well known and the mathematical representation of the weight of a term in a document by TF-IDF is given below in Figure 8. Here N is the number of documents and *d f (t)* is the number of documents containing the term t in the corpus [3]. The first term in the equation in Figure 8 improved the recall while the second term improved the precision of the resulting word embedding/TF-IDF vector [3].

The goal of using TF-IDF is to decrease the impact of tokens that occur very frequently in a given corpus and that are hence empirically less informative than features that occur in a small fraction of the training corpus [32].

For this baseline vector, each position in the vector represents a word/-token in the DA vocabulary and if the input DA or Turn includes a specific term the TF-IDF weight is given at the position representing the specific token. As will BoW, "stopwords" are removed before the TF-IDF vectors are created. The vectors will be 1,398 in length and each TF-IDF vector will be added into a Pandas dataframe along with the corresponding "Next DA New Topic" feature label to create this baseline 2 dataset.

7.4 XGBoost for Topic Handling in Experiments 1 and 2

XGBoost is the machine learning algorithm used as the Topic Trainer classifier in Experiments 1 and 2 (see Table 1). In [13], XGBoost is proposed as a scalable end-to-end tree boosting algorithm. Authors in [31] describe XGBoost as being used widely by data scientists to achieve state-of-the-art results on many machine learning challenges and is considered one of the best performing algorithms available for supervised learning. It can be used for regression or classification problems [31]. It is preferred by data scientists because of high execution speed of core computation [31] and there has been particular interest in finance due to flexibility, scalability and explainable [35]. In the following sections, we will first focus on the type of learning algorithm XGBoost falls under, ensemble learning with boosting, then go into more detail regrading gradient tree boosting otherwise known as gradient boosting machines [13] before going into more detail regarding the XGBoost algorithm. However, even before jumping into ensemble learning with boosting, we touch on why we do not choose to experiment using BERT which is a machine learning technique for NLP [18].

7.4.1 Why not BERT?

It is important to note that we do not implement the increasingly popular Bidirectional Encoder Representations from Transformers (BERT) machine learning technique for NLP in our implementations. Unlike recent language representation models, BERT is designed to pre-train deep bidirectional representations form unlabeled text by jointly conditioning on both left and right context in all layers [18]. The pre-trained BERT model can be fine-turned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference without substantial task-specific modifications [18], sentiment analysis [24], text classification, text generation, and sequence labeling [19]. Prior to proposal of Transformers, such as the Transformers used in BERT architecture, NLP had been mostly training models from scratch for different NLP tasks [19]. BERT has yielded record breaking results through transfer learning where a language model is trained using the information learned from the language model of many other NLP tasks [19].

While Transformers and BERT have made significant improvements in creating new state-of-the-art results for NLP tasks, one common feature in studies built upon BERT architecture is that even in tuning the pre-trained models, very large datasets are used [19]. As previously mentioned we are currently working with the SPAADIA corpora, which yields a small number of samples for training especially when considering the Turn level dataset. In [19], authors conduct an experiment where they compare the performance of a bidirectional Long Short-Term Memory (LSTM) and BERT for intent classification using a small corpus. The size of the corpus is 23,703 with 15,101 of the samples used in training [19], a corpus size larger than our DA level dataset and more than double the size of our Turn level dataset. The experimental results show that for smaller datasets, BERT overfits and does not perform as well as a simple LSTM model [19]. The simple LSTM model can be trained in much less time than the pre-trained counterpart and potentially would be an interesting classifier to experiment with for our task in future research should significantly more data not be readily available.

In addition, this research is focused on understanding if speech act patterns in dialogue can help an NPC better identify when proposing a new topic versus continuing the current topic is appropriate. If we were focused solely on using DA utterance text, and not SAASA data or patterns, using BERT would be a smart consideration for creating a binary classifier however would again need a significantly larger data. Future work could include using BERT for predicting "New Topic" label in parallel to appropriate machine learning applied classification trained on SAASA data, or a combination of both input into an LSTM or BERT could be considered.

For instance in [18], authors point out how it is critical to use a document-level corpus rather than a shuffled sentence-level corpus for pre-training data in order to extract long continuous sequences. As described above our corpus fo dialogues (at both Turn level and DA level) do not have such lengthy sequences, in particulate the SAASA feature sequences are very short. Therefore we will keep pre-trained BERT for future research on how best to use SAASA data alone or in parallel with DA utterance text. We will however leave this for future work and therefore, for our work here, we select using XGBoost for our experimentation and will discuss this algorithm in more detail below.

7.4.2 Ensemble Learning and Boosting

Because of the simplicity of methods such as linear regression, logistic regression, decision trees, SVM, KNN, these classifiers are often unable to produce a model accurate enough for specific problems [10]. While one can try using deep neural networks, in practice deep neural networks require a significant amount of labeled data which you might not have therefore another approach is to boost the performance of simple learning algorithms by performing ensemble learning. Ensemble learning is a paradigm that instead of trying to learn one accurate model focuses on training a large number of low-accuracy models and then combining the predictions given by those weak model to obtain a high-accuracy meta-model [10].

Weak learners are learning algorithms that cannot learn complex models and are therefore fast at training and prediction time. The aforementioned low-accuracy model are usually learned by weak learners [10]. Decision tree learning algorithms are the most frequently used algorithms which often stop splitting the training set after just a few iterations. Author in [10] explains that the created trees are shallow and not particularly accurate but idea behind ensemble learning is that if trees are not identical and each tree is at least slightly better than random guessing, then we can obtain a high accuracy by combining a large number of such trees. The prediction for a specific sample input is based on the predictions of each weak model and are combined using some sort of weighted voting where vote weighting form depends on the algorithm [10]. In the next section we go into more detail regarding boosting, a principle method of ensemble tree learning.

7.4.3 Boosting

Bagging and boosting are two principle methods of ensemble learning. Bagging consists of creating many "copies" of the training data, with each copy is slightly different from another, and then applying the weak learner to each copy to obtain multiple weak models and then combine them [10]. A widely used and effective such algorithm is random forests. Boosting consists of using the original training data and the iterative creation of multiple models by using a weak learner. Each new model is different from the previous ones in the sense that the weak learner, the errors made by previous models will be "fixed" by building a new model. In Figure 9, we see a diagram of XGBoost tree being run and the residual error (previous prediction label minus current model prediction; for initial tree equals actual label minus prediction) being carried over to the newly created model for a regression classifier example. The final ensemble model is a certain combination of those multiple weak model built iteratively [10].

There are 3 main tuning parameters for boosting algorithms: number of trees, learning parameter (otherwise known as shrinkage parameter) and maximum depth (i.e. number of splits in each tree) [26]:

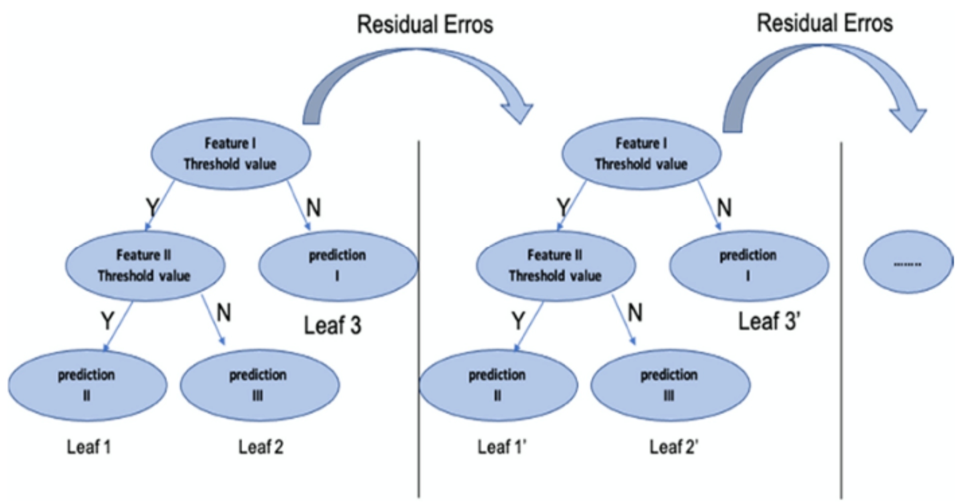


Figure 9: Schematic of XGBoost Trees [31]

Number of Trees: Unlike in bagging and random forests, boosting can overfit if the number of trees is too large, however such overfitting tends to occur slowly if at all [26]. Cross validation is needed in order to select the most optimal number of trees.

Learning rate: a small positive number that controls the rate at which boosting learns [26]. Typical values are 0.01 or 0.001 and the right choice can depend on the problem. A very small learning rate can require using a very large value number of trees in order to achieve good performance.

Maximum depth: controls the complexity of the boosted ensemble [26]. Maximum depth is the interaction depth and controls the interaction order of the boosted model since the number of splits can involve at most the same number of variables. Often maximum depth equals 1 works well, in which case the tree is a stump that consists of a single split. In this case, the boosted ensemble is fitting an additive model since each term involves only a single variable.

As mentioned above, will focus on optimizing F1 score by trying various values for the three principal hyperparameters for gradient boosting: number of trees, learning rate, and maximum depth (see Table 13 for values used in cross-validation of XGBoost based experiments). In Appendix Tables (e.g. Table **??** for Experiment 1) we also include the hyperparameters values that achieved the best F1 score during cross-validation for each classifier in each experiment.

```
XGBClassifier
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              eval_metric='aucpr', gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=4,
              num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

Figure 10: Default Values for XGBoost Algorithm and eval_metric = aucpr)

7.4.4 Gradient Tree Boosting

Gradient boosting is an effective ensemble learning algorithm based on the idea of boosting. Gradient boosting is one of the most powerful machine learning algorithms because it creates very accurate models and is capable of handling huge datasets with millions of examples and features [10]. This algorithm outperforms random forest in accuracy but because of its sequential nature can be significantly slower during training.

As with other boosting algorithms, three principal hyperparameters to tune in gradient boosting are the number of trees, the learning rate and the depth of trees as all three affect model accuracy [10]. The depth of trees also affects the speed of training and prediction (i.e. shorter trees mean faster training and prediction) [10]. For this reason, in our own experimentation, we will only be tuning these three hyperparameters described in section 7.4.3 during optimization and cross-validation.

Gradient boosting for classification is similar to what is described in the previous section however the steps are slightly different [10]. Similar to logistic regression, the predicton of the ensemble of decision trees is modelled using the sigmoid function and we apply the maximum likelihood principle by trying to find the right ensemble model that maximizes the likelihood.

7.4.5 XGBoost Classification

As described above XGBoost is a popular machine learning algorithm and is a scalable end-to-end tree boosting algorithm and can be used for both regression and classification problems [13]. XGBoost has desirable advantages of being flexible, explainable and scalable while also providing state-of-the-art performance in most supervised learning tasks [35]. It has made excellent improvements in the implementation level and it utilizes parallel learning, out-of-core computation and cache-aware access techniques to minimize the time complexity of learning, thus being applicable for very large datasets [41].

XGBoost has improved the traditional gradient boosting by adding up some efficient techniques to control overfitting, split finding and handling missing values in the training phase [41]. It differs significantly from other convex-margin based classifiers, such as Support Vector Machines (SVM), logistic regression, neural networks, in the way the model parameters are updated. In order to control the overfitting, the objective (minimization) function consists of two parts: loss function and regularization term (which controls the complexity of the model [41]. Here a second-order approximation is used to optimize the objective function and in each iteration the best tree is selected [41].

In addition, an approximate technique is used to split finding in each node of the tree. In this technique, for each feature all instances are sorted by the feature's value, then a linear search is done to find the best split along that feature. The best split is chose by comparison amongst the best split of all the features [41]. Additionally, after constructiong the tree, in each node the direction with maximum score is marked to make a default path on the tree for classification of data with missing values.

7.4.6 XGBoost Summary

Above we offer an overview of ensemble learning, describe the machine learning algorithm XGBoost, its advantages to other popular algorithms and the three different hyperparameters we tune in our experiments in order to optimize each classifier and to demonstrate the impact of incorporating SAASA data to create the proposed Topic Handler model (number of trees, learning rate and maximum tree depth).

In the next section, begin with Experiment 1, reiterate each XGBoost classifier and dataset for each subexperiment, then present and discuss the results.

7.5 Experiment 1 Implementation: Predicting "New Topic" feature class using Dialogue Act Level Dataset

In Chapter 2, there are various related areas of research discussed (including CA and dialogue grammars). In these areas of research, it is clear that patterns among dialogues are a clear indicator of what is currently going on and what is expected to come next in the following 'bit of talk' [11]. However given the twelve training features per DA for use when making the prediction label classification, we now have even more information than the initial idea of patterning just speech act data for topic handling. With the additional SAASA data, as oppose to just the speech act label, can the patterning of these features further optimize predicting the correct topic-related label for the next DA? In the following sections, we begin by trying to predict one label: "Next DA New Topic" class for a DA using five data splits in the simulated M4. We also use the Bag of Words (BoW) and TF-IDF methods on the DAs themselves as baselines for predicting "Next DA New Topic" class, then compare the classifier performance when trained using baseline vs SAASA data. As mentioned above, we will be measuring model performance by precision, recall and F1 score because we are working with a class imbalanced dataset. We will also compare classifier performance using AUC-PR.

Experiment 1 DA level Dataset	Binary Classifier to 'Predict Next DA New Topic' Feature	Input Features
Baseline 1	BoW of DA level Dataset	BoW feature vectors based on Dialogue Act/Feature 1 in Table 7
Baseline 2	TF-IDF of DA level Dataset	TF-IDF feature vectors based on Dialogue Act/Feature 1 in Table 7
Topic Trainer 1	DA level Dataset SAASA features minus DA	Feature 2 - 11 listed Table 7
Topic Trainer 2	DA level Dataset SAASA features Plus BoW DA	Feature 2 - 11 listed Table 7 and BoW DA Vectors
Topic Trainer 3	DA level Dataset SAASA features Plus TF-IDF DA	Feature 2 - 11 listed Table 7 and TF-IDF DA Vectors

Table 14: Module 5 Topic Trainer Experiment 1

7.5.1 Experiment 1 Imputers, Scalars, Encoders and Vocabulary Sizes Used

For Experiment 1 Topic Trainer 1, we use the MinMaxScalar for numeric features and the OneHotEncoder for categorical features and for 1 Topic Trainer 2 and 3 we use the StandardScalar for numeric features and the OneHotEncoder for categorical features. We select these based on predictions using the SAASA validation dataset using different combinations of scalars and encoders. MinMaxScalar and OneHotEncoder resulted in the best AUC-PR without optimization for Topic Trainer 1 whereas StandardScalar and OneHotEncoder resulted in the best AUCPR for Topic Trainers 2. See Table 22 in Appendix for complete results.

Imputers are used to set null numerical features to zero and any null categorical values are assigned 'missing'. Any new unknown categorical features introduced in the validation and test set, but were not found in the training set, are set to be ignored.

For the unstructured feature, DA vectors, used in the baseline datasets, Topic Trainer 2 and Topic Trainer 3, the vocabulary is limited to a maximum length value of 150. In the training dataset the vocabulary size for all DAs is 1,563. All English stop words are removed and text is converted to lowercase.

7.5.2 Experiment 1 Pre- and Post-Optimization Success Metrics Results and Discussion

Based on the pre-optimization results in Table 15 we see all classifiers, including the baseline classifiers, already achieve a higher AUC-PR than a random classier without optimization (discussed in section 7.1.2, a random classifier prediction on DA level dataset would achieve an AUC-PR of 0.324). Pre-optimization, classifiers Topic Trainer 1 and 3 achieve the highest precision score, with Topic Trainer 1 having the overall highest F1 score and Topic Trainer 3 having the highest AUC-PR. While Topic Trainer 2 achieves the highest recall pre-optimization, it has a lower precision and AUC-PR than Topic Trainers 1 and 3.

Following optimization using GridSearchCV, we apply the threshold of the best resulting classifier (based on results on Validation set; see Tables 24 and 25 in Appendix) and use same model to get the below results for predictions made on the test DA level dataset (see Table 16). Post-optimization, we see that Topic Trainer classifiers perform better than the Baseline classifiers, in particular we see consistent AUC-PR over 0.5 for Topic Trainer classifiers whereas Baseline classifiers remain below 0.5. Post-optimization, we also see a reduction in precision and increase in recall across all 5 classifiers. Baselines 1 and 2 achieve the highest recall but the lowest precision scores across the experiment results. As previously mentioned, for this context this is less desirable and we want to prioritize a higher precision (less false positives) over higher recall (less false negative) to reduce unwarranted topic change and flouting of previously discussed conversation norms.

As expected, the post-optimization F1 scores and AUC-PRs on the test DA level dataset is also lower compared to those achieved pre-optimization in Table 15. Based on the post-optimization results, while Topic Trainer 1 and 3 achieve the same precision score again, Topic Trainer 1 achieves the highest AUC-PR and second highest F1 score (by 0.002), making it the more promising of the two Topic Trainer classifiers. Highest precision (higher than Topic Trainers 1 and 3 by only 0.004) and F1 score are achieved by Topic Trainer 2 however achieved the lowest AUC-PR of the three Topic Trainers (lower that Topic Trainer 1 and 3 by 0.016 and 0.006 respectively), indicating it is potentially the least likely to accurately predict future classification on novel input datasets.

Experiment Dataset	Best Threshold Precision	Best Threshold Recall	Best Threshold F1	AUC-PR
Baseline 1	0.375	0.941	0.536	0.474
Baseline 2	0.374	0.893	0.527	0.473
Topic Trainer 1	0.412	0.870	0.559	0.565
Topic Trainer 2	0.404	0.883	0.554	0.563
Topic Trainer 3	0.412	0.832	0.551	0.579

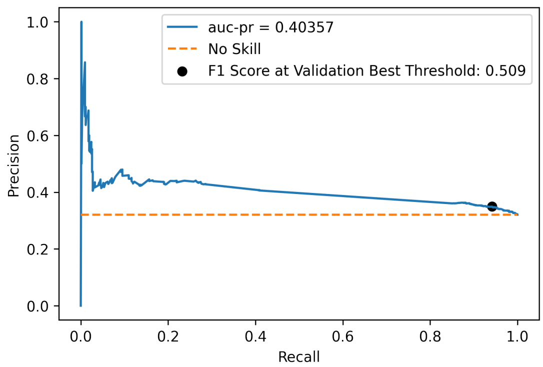
Table 15: Module 5 Topic Trainer Experiment 1 Results at Best Threshold Pre-Optimization on Validation Dataset

Experiment Dataset	Best Threshold Precision	Best Threshold Recall	Best Threshold F1	AUC-PR
Baseline 1	0.349	0.942	0.509	0.404
Baseline 2	0.344	0.939	0.503	0.424
Topic Trainer 1	0.374	0.939	0.535	0.553
Topic Trainer 2	0.380	0.918	0.537	0.537
Topic Trainer 3	0.374	0.918	0.531	0.543

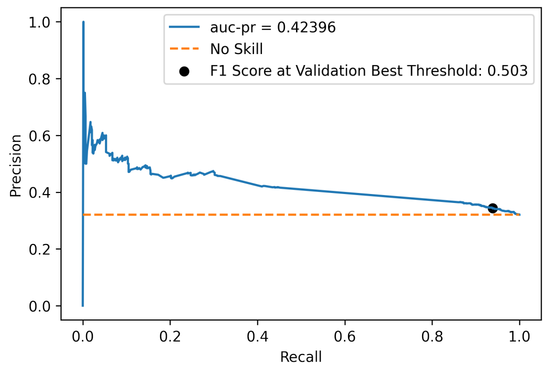
Table 16: Module 5 Topic Trainer Experiment 1 Results at Best Threshold Post-Optimization on Test Dataset

In Figure 12, Topic Trainer 1, 2 and 3 AUC-PR plots are shown. Topic Trainer 1, classifier based on SAASA data alone, achieves the highest AUC-PR score indicating they are most like to accurately predict future classifications on novel input datasets. When comparing the shape of Baseline AUC-PRs in Figure 11 to the Topic Trainer AUC-PRs in 12 we can see the stark different in variance from random classifier performance as well as relationship of best F1 score to precision and recall. For the Topic Trainer classifiers, these plots clearly demonstrate the improved ability for these to more accurately predict future classifications without sacrificing as much precision for improved recall and therefore improved F1 score.

Overall, based on the performance metrics and AUC-PR scores the best performing classifier it Topic Trainer 1. While it did not achieve the highest precision or F1, Topic Trainer 1 did achieve the second highest for each metric in addition to the second highest recall score and the highest AUC-PR. This indicates

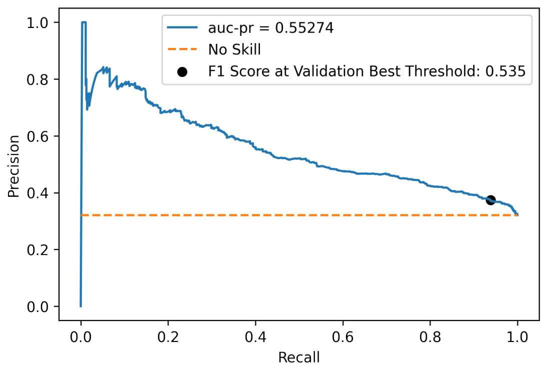


(a) Experiment 1 Baseline 1 AUC-PR

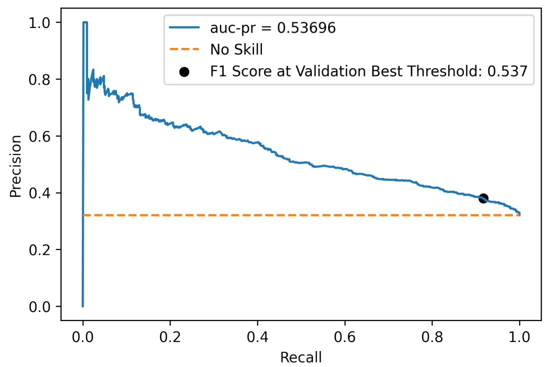


(b) Experiment 1 Baseline 2 AUC-PR

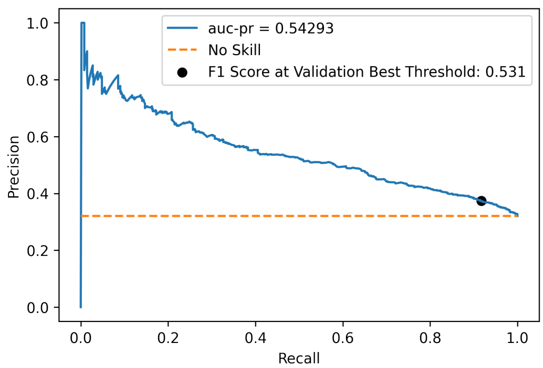
Figure 11: Experiment 1 Baseline Best Threshold for Optimal F1 score and AUC-PR for on Test Set



(a) Experiment 1 Topic Trainer 1 AUC-PR



(b) Experiment 1 Topic Trainer 2 AUC-PR



(c) Experiment 1 Topic Trainer 3 AUC-PR

Figure 12: Experiment 1 Topic Trainer Best Threshold for Optimal F1 score and AUC-PR on Test Set

the potential value of focusing on leveraging SAASA data for topic handling over utilizing DA tokens along.

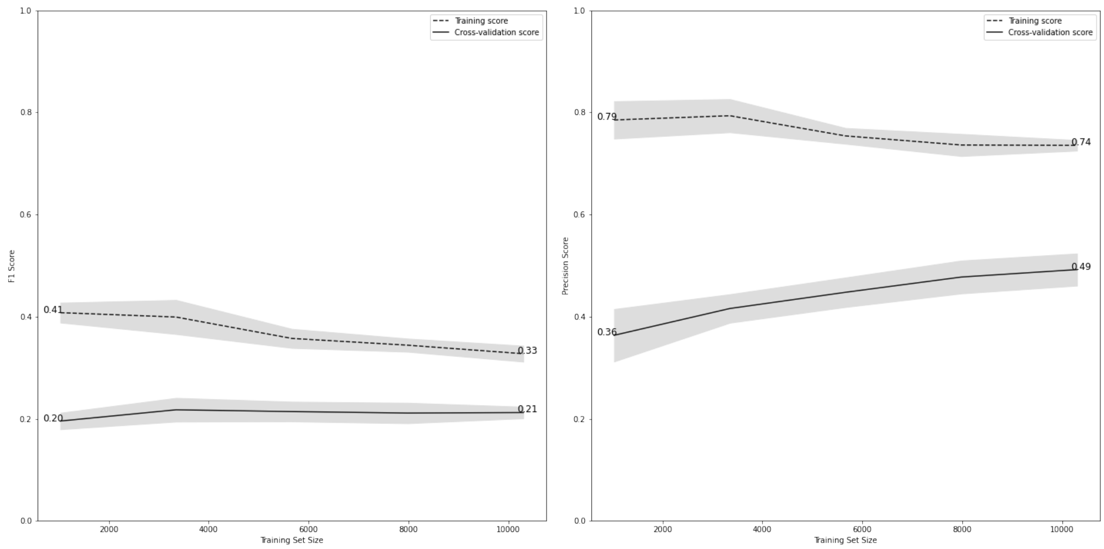
In the next section we review the resulting learning curves from Experiment 1 and analyze potential impact of the addition of more data on the classifiers.

7.5.3 Experiment 1 Learning Curves for Precision and F1 Score

In addition to the success metrics results demonstrating the potential power in utilizing SAASA data to help with topic handling, the learning curves also demonstrate the impact additional data would have on the Baseline classifiers versus the Topic Trainer classifiers in Experiment 1. In Figures 13 and 14 we see the learning curves for Baselines 1 and 2. In Figures 20a and 14a we see the training score for f1 starts higher at the beginning and decreases and the cross-validation score is low at the beginning and increases, this shape of the curve is often found in more complex datasets. As previously explained in section 7.2.3, when the F1 and precision training and cross-validation curves do not move towards convergence or when the curves begin to do so at a value quite low, this indicates that the classifier likely would not benefit or improve generalization with more training data. While the shape of the curves is common for more complex datasets the point of converge for both Baselines and metrics are still quite low (for F1 score below 0.45 and precision in Baseline 1 below 0.7; Baseline 2 precision curves do not show indications of convergence for this training dataset size). Therefore additional data for the baseline classifiers would likely not improve the performance in prediction "New Topic" feature class. Additionally, the variability around both the training and cross-validation curves in Figures 13 and 14 is implying a bias error.

Topic Trainers 2 and 3 f1 and precision training scores in Figures 16 and 17 have a similar downward trajectory as Baselines but at first achieves much higher scores. In contrast to the Baselines learning curves however, the training score curves for Topic Trainers 2 and 3 demonstrate much less variability and the cross-validation score curve demonstrate more which indicates a variance error over bias error.

In Figure 15, the training score is much greater than the validation score and appears that adding more training samples will most likely improve and increase the method's generalization. The f1 and precision training scores in Figure 15 for Topic Trainer 1 remains high through out and f1 and precision cross-validation scores slowly begin to increase indicating convergence, if occurs, would



(a) Baseline 1 F1 Learning Curve

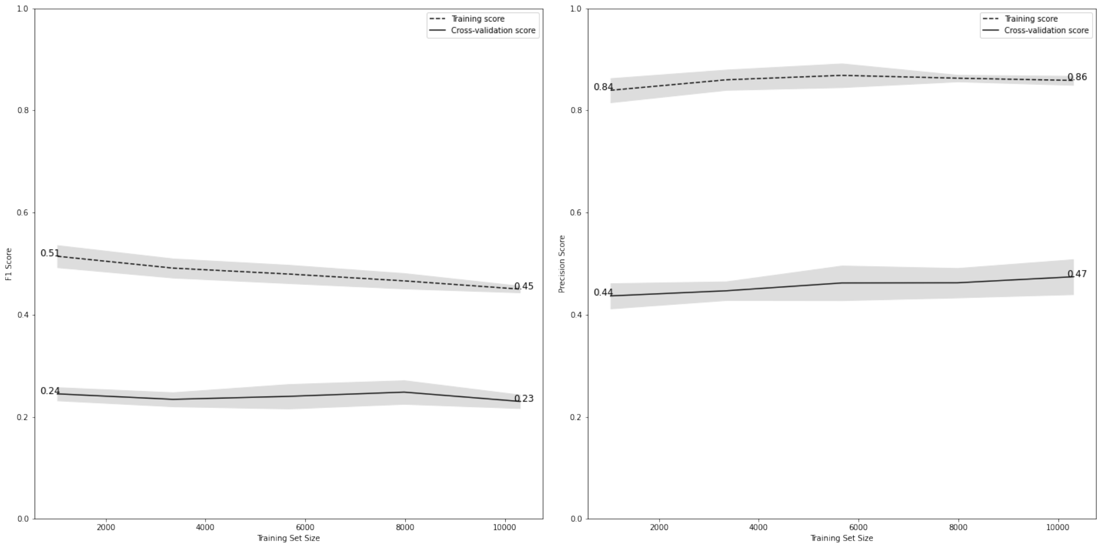
(b) Baseline 1 Precision Learning Curve

Figure 13: Learning Curves of Experiment 1 Baseline 1: BoW of Dialogue Act

occur at a high value. Additionally, the variability on the cross-validation score in Figures 15 indicates variance error and not bias error for Topic Trainer 1 in this experiment. And as with the previous section, where the classifiers were compared based on performance metrics alone, Topic Trainer 1 appears to have the most promising potential with the highest training score achieved and maintained as well as the validation score improved and increasing as more data tested.

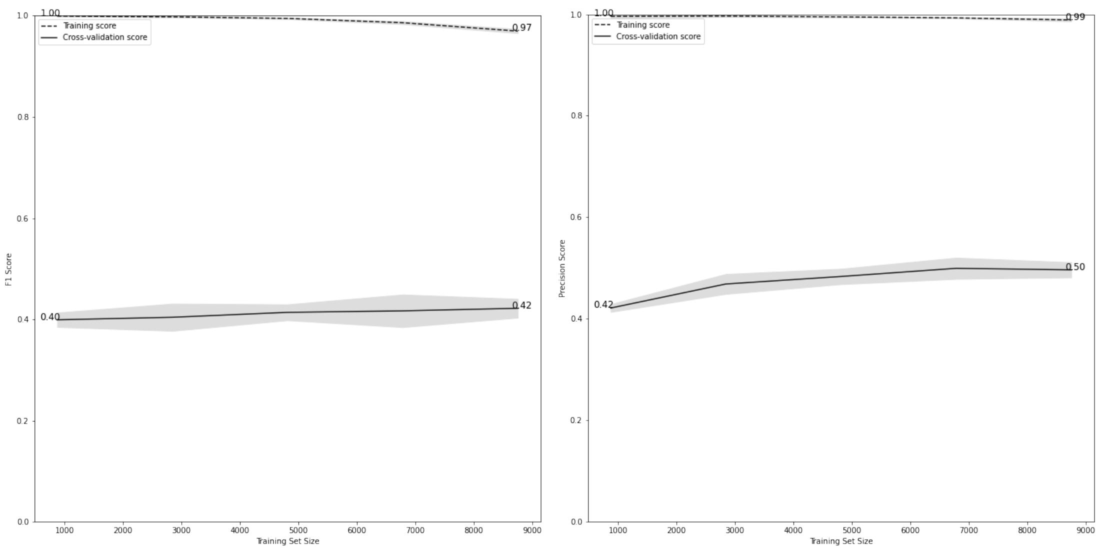
7.5.4 Experiment 1 Discussion and Summary

Results of Experiment 1 demonstrates use of SAASA data offers improved performance in predicting if the next DA should refer to a “New Topic”. Topic Trainer 1, which relied solely on SAASA data offered immediate performance improvements to the Baseline classifiers as well as the highest AUC-PR, indicating Topic Trainer 1 is the classifier most likely to accurately predict future “New Topic” feature class on novel input datasets, and most promising learning curves indicating Topic Trainer 1 as the classifier that would most likely to improve performance even more so give additional input data. Topic Trainers 2 and 3 also provided significant improvements on Baseline classifiers 1 and 2, with Topic Trainer 2 achieving the highest precision and F1 score, however they do not achieve as high AUC-PR scores and their learning curves do not indicate as much potential improvements in performance scores should additional novel data be available. Therefore based on the results of Experiment 1, Topic Trainer 1 is the best performing classifier for the M5 in the proposed Topic Handler Model, and scalable



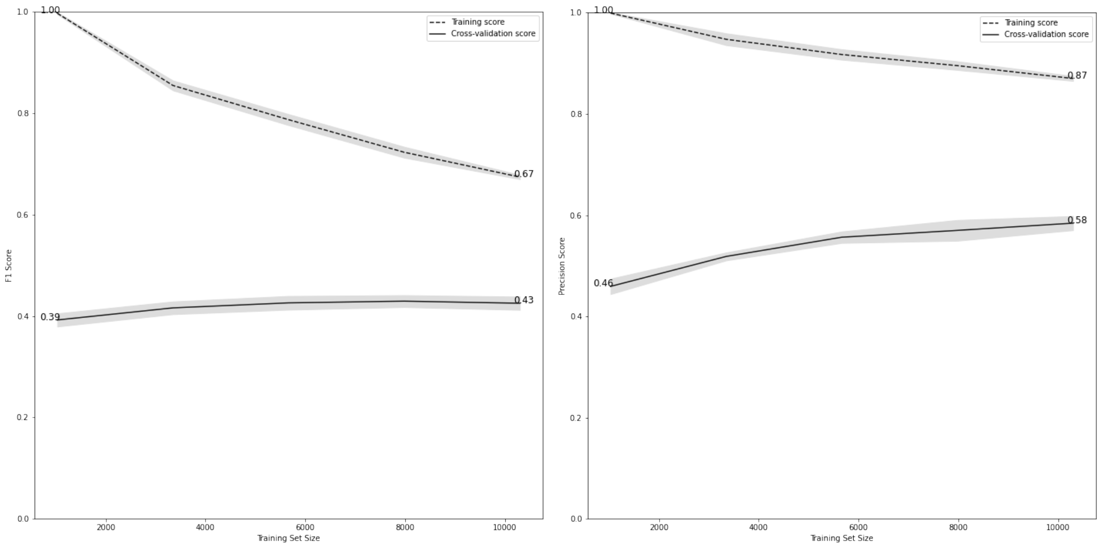
(a) Baseline 2 F1 Learning Curve (b) Baseline 2 Precision Learning Curve

Figure 14: Learning Curves of Experiment 1 Baseline 2: TF-IDF of Dialogue Act



(a) Topic Trainer 1 F1 Learning Curve (b) Topic Trainer 1 Precision Learning Curve

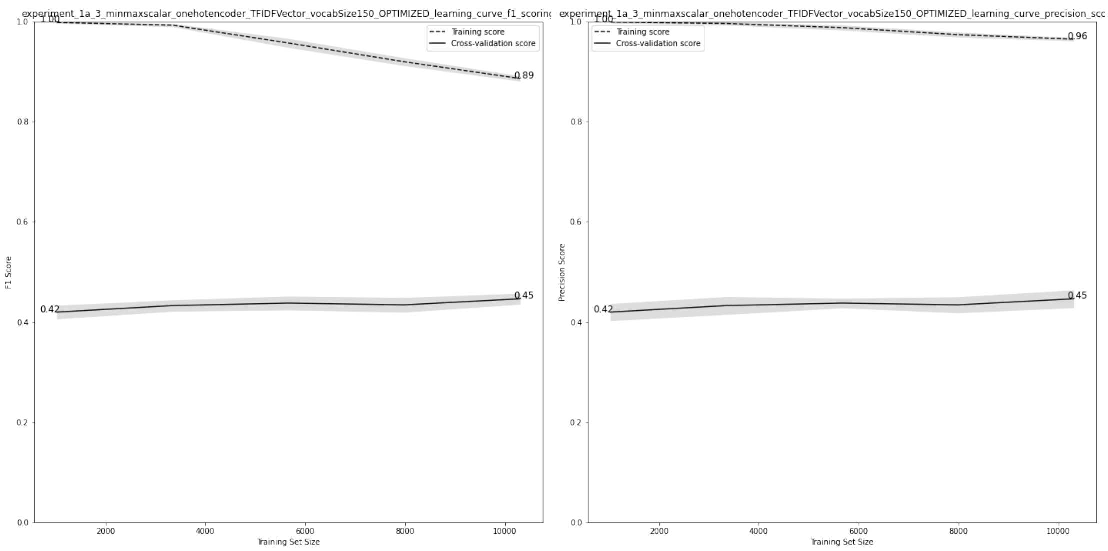
Figure 15: Learning Curves of Experiment 1 for Post-Optimized Topic Trainer 1



(a) Topic Trainer 2 F1 Learning Curve

(b) Topic Trainer 2 Precision Learning Curve

Figure 16: Learning Curves of Experiment 1 for Post-Optimized Topic Trainer 2



(a) Topic Trainer 3 F1 Learning Curve

(b) Topic Trainer 3 Precision Learning Curve

Figure 17: Learning Curves of Experiment 1 for Post-Optimized Topic Trainer 3

for future classifications based on DA level dataset.

In the next section, will take a look at utilizing the Turn level dataset as input into XGBoost classifiers for use as M5 in the proposed Topic Handler Model instead of DA level inputs.

7.6 Experiment 2 Implementation: Predicting "New Topic" feature class using Turn Level Dataset

In Experiment 2, we instead have the Turn level dataset as input into our classifiers with Baseline and Topic Trainers classifiers defined in Table 17. Features for this experiments are as defined in Table 8 and re-iterated here in Table 18. Significant difference is that majority of input features are each now represented as vectors as each Turns can include more than one DA. For each DA in a Turn, there is SAASA information available. Vectors are similar to those representative of BoW, where vector is the length of possible values for each DA and each position in vector represents a specific token. If token is present in Turn, position representing the token is 1. Additionally, just as in BoW and TF-IDF, the order of the tokens is not maintained.

In Table 17, we list the different datasets and classifiers utilized in this experiment. In addition to having binary count vectors for each of the non-numeric and non-DA based features (i.e. ITFS, ITLS, ITPS and ITMS in Table 18) we experiment with these vectors as input into a XGBoost classifier as TF-IDF vectors and included the results in the Appendix in Table 30 as well as post-optimization AUC-PR curve in Figure 25 and learning curves in Figure 26. As this is an additional classifier not directly comparable to Experiment 1 or outside initial proposed remit have left outside of this focus and for further investigation in future work.

Experiment 2 Turn level Dataset	Binary Classifier to 'Predict Next DA New Topic' Feature	Input Features
Baseline 1	BoW of Turn level Dataset	BoW feature vectors based on Dialogue Act/Feature 1 in Table 7
Baseline 2	TF-IDF of Turn level Dataset	TF-IDF feature vectors based on Dialogue Act/Feature 1 in Table 7
Topic Trainer 1	Turn level Dataset SAASA features minus DA	Feature 2 - 11 listed Table 7
Topic Trainer 2	Turn level Dataset SAASA features Plus BoW Turn DA(s)	Feature 2 - 11 listed Table 7 and BoW Turn DA(s) Vectors
Topic Trainer 3	Turn level Dataset SAASA features Plus TF-IDF Turn DA(s)	Feature 2 - 11 listed Table 7 and TF-IDF Turn DA(s) Vectors

Table 17: Module 5 Topic Trainer Experiment 2

	In-Turn Input Feature	Example of one Turn with 3 DAs
1	In-Turn Dialogue Acts *In-Turn Dialogue Act 1:* "ok" *In-Turn Dialogue Act 2:* "what about a a change of planes or a quick stop-over?" *In-Turn Dialogue Act 3:* "or one stop maybe in either Portland or Seattle and continuing on down"	**In-Turn DA:** "ok what about a a change of planes or a quick stop over or one stop maybe in either Portland or Seattle and continuing on down"
2	Turn Count	37
3	In-Turn Speech Act Form Sequence (ITFS)	dm q-wh decl
4	In-Turn Speech Act Label Sequence (ITLS)	acknowledge suggest reqInfo
5	In-Turn Polarity Sequence (ITPS)	None positive positive
6	In-Turn Mode Sequence (ITMS)	tag alternative-closed-query alternative-opinion-query

Table 18: Turn level Dataset Training Features and examples

7.6.1 Experiment 2 Scalars and Vocabulary Sizes Used

In Experiment 2 there is just 1 numerical feature (Turn Count) and no categorical features as the features are now each input vectors as discussed above. For Topic Trainer 1 we will use MinMaxScalar for the numerical feature but for Topic Trainer 2 and 3 we will use StandardScalar for the Turn Count feature. These scalars are selected based on scalar pre-optimization prediction performance on each classifier validation set. MinMaxScalar results in the best pre-optimization AUC-PR for Topic Trainer 1 whereas StandardScalar resulted in the best pre-optimization AUC-PR for Topic Trainers 2 and 3. See Table 26 in Appendix for specific results of these tests.

7.6.2 Experiment 2 Pre- and Post-Optimization Success Metrics Results and Discussion

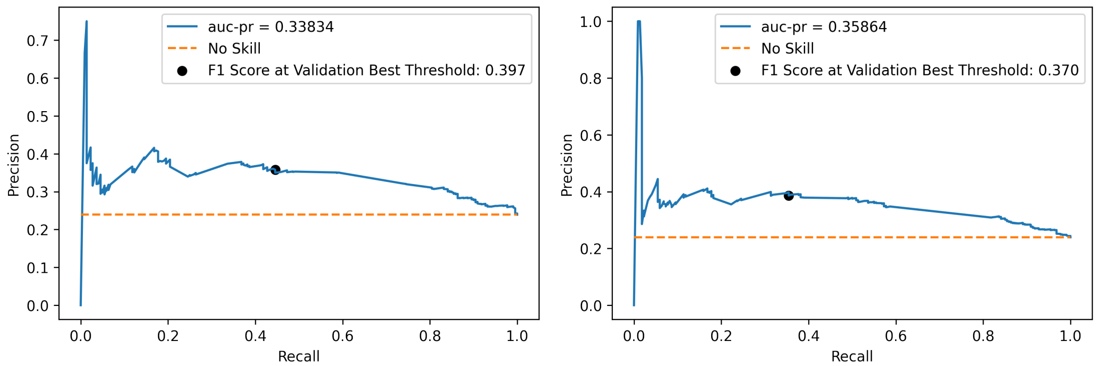
For each of the classifiers in Experiment 2, they perform better than a random classifier with AUC-PR scores over 0.2170 both pre- and post- optimization. Overall performance metric scores however do not perform as well as the results in Experiment 1. This could be attributed to the reduced number of examples in the Turn level dataset but could also be attributed to suitability of the Turn level dataset as input features into XGBoost.

Baseline 2 achieves the best precision score in pre- and post-optimization but achieves significantly lower F1 score and AUC-PR than the Topic Trainer classifiers. When looking at post-optimization results in Table 20, Baselines 1 and 2 achieve the highest Precision values however also achieve the lowest F1 scores and AUC-PR.

The highest post-optimization AUC-PR achieved by Topic Trainers 1 and 2 and these classifiers are the only ones that achieve an AUC-PR over 0.4. Topic Trainer 1 achieves the highest F1 score (pre- and post-optimization) and highest post-optimization precision as well as the second highest recall and AUC-PR (less 0.013 and 0.003 respectively when compared to Topic Trainer 2). Topic Trainer 1 post-optimization precision and F1 score are both 0.012 higher than Topic Trainer 2 and thus, similar to the Experiment 1 findings, overall Topic Trainer 1 is the best option when taking into consideration the 2 performance metrics.

Experiment Dataset	Best Threshold Precision	Best Threshold Recall	Best Threshold F1	AUC-PR
Baseline 1	0.331	0.618	0.431	0.339
Baseline 2	0.368	0.524	0.432	0.331
Topic Trainer 1	0.346	0.691	0.462	0.365
Topic Trainer 2	0.356	0.618	0.452	0.370
Topic Trainer 3	0.306	0.770	0.438	0.378

Table 19: Module 5 Topic Trainer Experiment 2 Results at Best Threshold Pre-Optimization on Validation Dataset



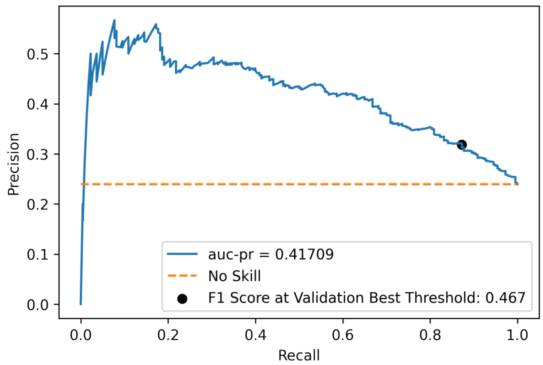
(a) Experiment 2 Baseline 1 AUC-PR (b) Experiment 2 Baseline 2 AUC-PR

Figure 18: Experiment 2 Baseline Best Threshold for Optimal F1 score and AUC-PR for on Test Set

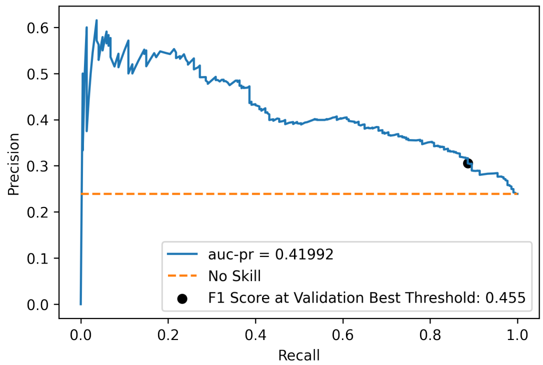
Experiment Dataset	Best Threshold Precision	Best Threshold Recall	Best Threshold F1	AUC-PR
Baseline 1	0.358	0.445	0.397	0.338
Baseline 2	0.386	0.355	0.370	0.359
Topic Trainer 1	0.318	0.873	0.467	0.417
Topic Trainer 2	0.306	0.886	0.455	0.420
Topic Trainer 3	0.315	0.873	0.463	0.396

Table 20: Module 5 Topic Trainer Experiment 2 Results at Best Threshold Post-Optimization on Test Dataset

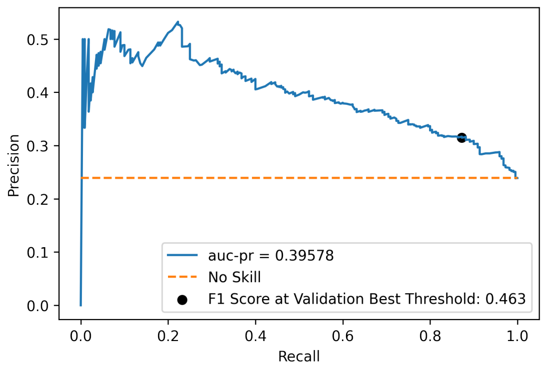
In Figure 12, Topic Trainer 1, 2 and 3 AUC-PR plots are shown. Topic Trainer 2 and 3 are the two classifiers achieved the highest AUC-PR scores indicating they are most like to accurately predict future classifications on novel input datasets. When comparing the shape Baseline AUC-PRs in Figure 11 to the Topic Trainer AUC-PRs in 12 we can see the stark different in variance from random classifier performance as well as relationship of best F1 score to precision and recall. For the Topic Trainer classifiers, these plots clearly demonstrate the improved ability for these to more accurately predict future classifications without sacrificing as much precision for improved recall. And while Topic Trainer 1 classifier, based only on use of SAASA data, performs better than both baselines, from these results we can also conclude that a classifier will succeed even more so if SAASA data is used in conjunction with DA text representation vectors as done in Topic Trainers 2 and 3.



(a) Experiment 2 Topic Trainer 1 AUC-PR



(b) Experiment 2 Topic Trainer 2 AUC-PR



(c) Experiment 2 Topic Trainer 3 AUC-PR

Figure 19: Experiment 2 Topic Trainer Best Threshold for Optimal F1 score and AUC-PR on Test Set

7.6.3 Experiment 2 Learning Curves for Precision and F1 Score

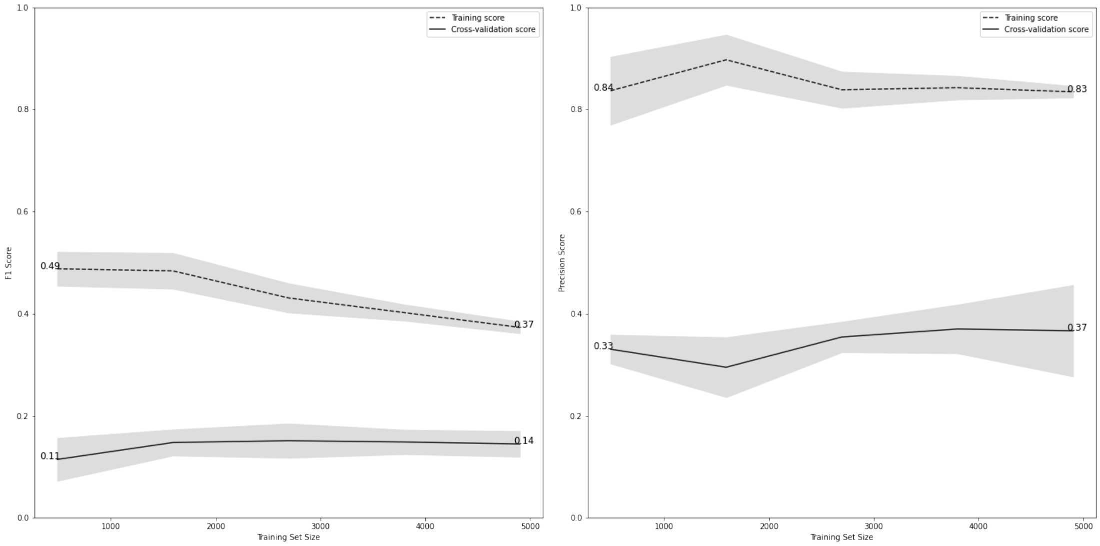
In addition to the success metrics results demonstrating the potential power in utilizing SAASA data to help with topic handling, the learning curves also demonstrate the impact additional data would have on the Baseline classifiers versus the Topic Trainer classifiers in Experiment 2. In Figures 20 and 21 we see the learning curves for Baselines 1 and 2. For Baseline 1, the F1 score curve has a downward turn and significant variability indicating bias error. The precision training and cross-validation curves start to move in an upward projection but then appears to plateau, with both curves showing significant variability also indicating bias error. Baseline 2 learnings curves behave similarly with indications of significant bias error but with a better starting and final precision and F1 score in the training curves. The learning curves for Baselines 1 and 2 do not imply there would be much improved performance or generalization with additional training data, in particular when compared to Topic Trainer learning curves.

Looking at the Topic Trainer learning curves in Figures 22, 23 and 24, we see all training score curves decline from a high score for both F1 and precision. As discussed in 7.2.3, this shape is typical of more complex datasets. The training curves still remain relatively high across all Topic Trainers, especially when compared to the Baseline learning curves, and all cross-validation curves move upwards, without a plateau, suggesting that the addition of more training data would most likely improve each of the Topic Trainers and increase their generalization. The emphasized variability on cross-validation score across all Topic Trainer learning curves indicates variance error and not bias error.

Therefore the learning curves display that Topic Trainer classifiers are more likely to improve and increase generalization with more training data whereas Baseline classifiers are shown to have signifiance bias error as well as less indication performance would improve with additional data. Overall Topic Trainer 2 maintains the highest training scores for F1 and precision, with Topic Trainer 1 and 3 maintaining relatively high scores as well.

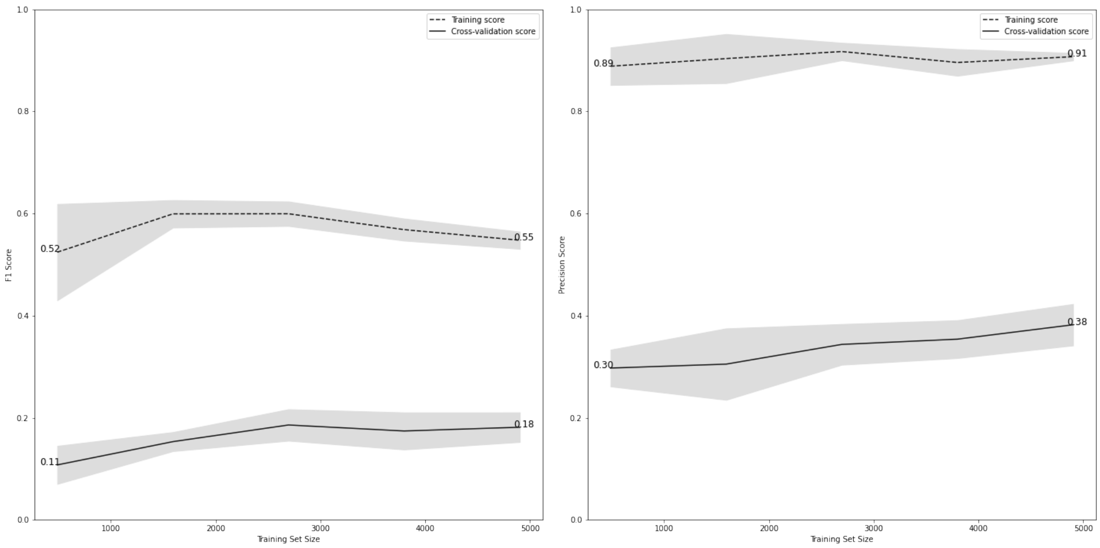
7.6.4 Experiment 2 Discussion and Summary

Based on Experiment 2 results, Topic Trainer 1 and 2 are the most promising classifiers for use in M5. Topic Trainer 1 has the highest F1 score and precision, along with the second highest AUC-PR and recall, and Topic Trainer 2 has the highest AUC-PR and recall. The Topic Trainer learning curves perform quite



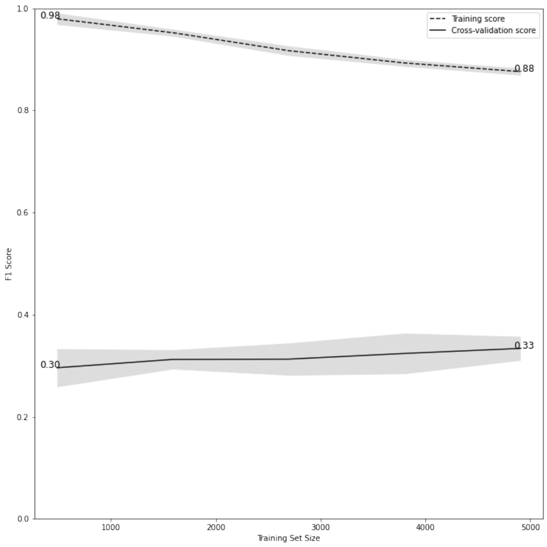
(a) Baseline 1 F1 Learning Curve (b) Baseline 1 Precision Learning Curve

Figure 20: Learning Curves of Experiment 2 Baseline 1: BoW of Dialogue Act

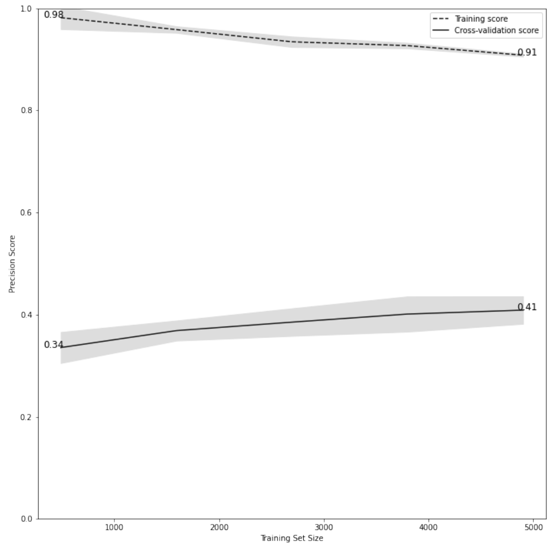


(a) Baseline 2 F1 Learning Curve (b) Baseline 2 Precision Learning Curve

Figure 21: Learning Curves of Experiment 2 Baseline 2: TF-IDF of Dialogue Act

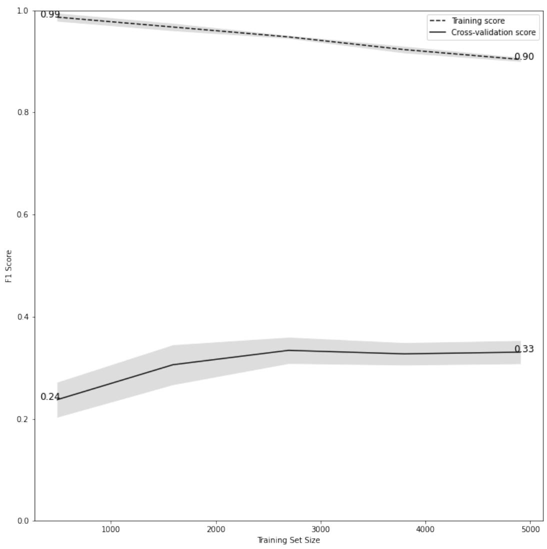


(a) Topic Trainer 1 F1 Learning Curve

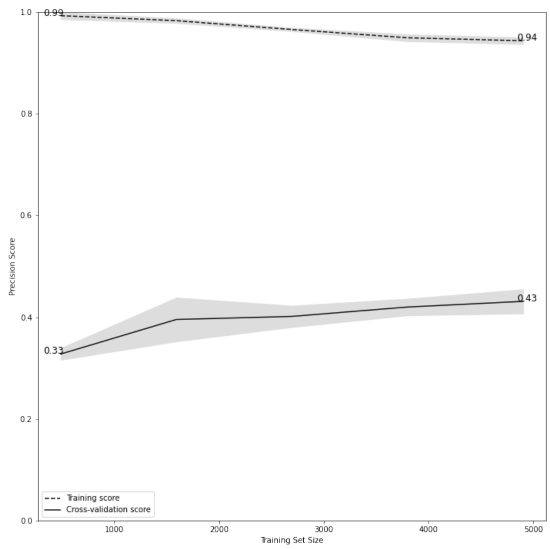


(b) Topic Trainer 1 Precision Learning Curve

Figure 22: Learning Curves of Experiment 2 for Topic Trainer 1

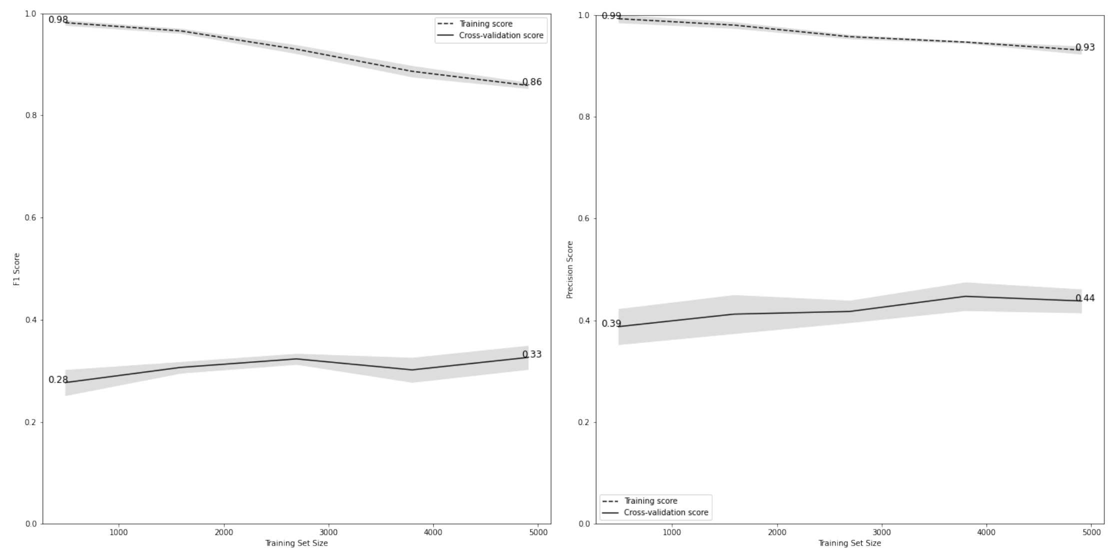


(a) Topic Trainer 2 F1 Learning Curve



(b) Topic Trainer 2 Precision Learning Curve

Figure 23: Learning Curves of Experiment 2 for Topic Trainer 2



(a) Topic Trainer 3 F1 Learning Curve (b) Topic Trainer 3 Precision Learning Curve

Figure 24: Learning Curves of Experiment 2 for Topic Trainer 3

similarly with Topic Trainer 2 achieving a slightly higher F1 and precision final score for training and cross-validation score than Topic Trainer 1 and 3. While the Topic Trainer 2 learning curve results are slightly higher, as well as its AUC-PR score, Topic Trainer 1 would still be the best option for future implementation as it achieves the highest F1 and precision. As previously mentioned we would prioritize higher precision.

The Baselines in this experiment achieve higher precision scores but then have significantly lower recall, and subsequently achieve lower F1 and AUC-PR scores as well as dissatisfying learning curves. Therefore we still favour Topic Trainer 1 for future use based on these results.

7.7 Proof of Concept Experimentation Summary

In both Experiments 1 and 2, we see classifiers trained using SAASA data improves performance in predicting if the next DA should be a “New Topic” (see Table 16 and Table 20 for summary fo Experiment results).

In Experiment 1, where classifiers are trained using the DA level dataset, we see that Topic Trainer 1, the classifier trained only on SAASA data, not only performed the best for this task but this classifier’s AUC-PR score and learning curves also demonstrate that of the five classifiers in Experiment 1 Topic Trainer 1 is the best candidate for future implementations.

In Experiment 2, where classifiers are trained on the Turn level dataset, we see from the results that Topic Trainer 2, the classifier trained on both SAASA and DA text as input features, achieved the most promising AUC-PR score and achieved the best performing learning curve suggesting this classifier has the most potential for accurate future predictions. The Topic Trainer 2 AUC-PR and learning curve performance however is only slightly higher than Topic Trainer 1's performance (AUC-PR for Topic Trainer 1 = 0.417 versus AUC-PR for Topic Trainer 2 = 0.420). Additionally Topic Trainer 1, the classifier trained only on SAASA data, achieved the highest post-optimization F1 score and higher precision score of the Topic Trainer classifiers in experiment 2 (Baseline 2 achieved a higher precision however also resulted in the lowest recall, F1 score as well as second lower AUC-PR and unpromising learning curves).

As mentioned above, when comparing the results of Experiments 1 and 2, we see that the classifiers in Experiment 1 perform significantly better than the classifiers trained and tested in Experiment 2. We believe this can be attributed to the significantly smaller size of samples in the Turn level dataset and the classifiers in Experiment 1 benefiting from training on a larger DA level dataset (see Table 9).

Topic Trainer 1, which leverages only SAASA data and no DA textual data, performs relatively well in both Experiments 1 and 2. This finding clearly demonstrates a potential benefit for using SAASA data for topic handling in human-computer dialogue, or even an advantage in focusing on SAASA data alone for the task of topic handling. Unlike Topic Trainer 2, Topic Trainer 1 learning curves in both Experiments 1 and 2 maintain high training scores for F1 and precision. Topic Trainer 1 is a clear example of how SAASA data can not only improve performance in identifying points to propose a new topic but also when used alone can outperform classification using DA data alone.

8 Discussion and Conclusion

In this thesis, we aimed to contribute to ongoing research in the field of human-computer dialogue and help move closer to the goal of having more realistic human-computer dialogue. In particular we aimed to address the current challenge of topic handling in human-computer conversation and the struggle for existing chatbots, virtual agents, and NPCs to engage in cohesive conversations containing multiple topics and transition between topics in a natural way. We chose to address this problem by asking the question: "Can speech act patterns in dialogue help an NPC better identify when proposing a new topic versus continuing the current topic is appropriate?". In order to answer this question, we reviewed speech act theory, conversation analysis, past and current chatbot implementations and dialogue systems and then used this knowledge to propose a Topic Handler model whose framework is designed to be compatible with dialogue systems and frameworks alike. We focused on creating this Topic Handler system within the context of use for NPCs as they require the most realism for immersive experience whereas other virtual agents (such as e-commerce chatbots and virtual assistants) tend to be focused on task-related dialogues (however we are not limiting potential improvements our research can inform on to just NPCs).

Similar to advanced automated chatbots, that generally use machine learning coupled with NLP within the domain of AI, we propose this Topic Handler model leverage machine learning classifiers to predict when a response should continue the current topic or if this is an appropriate time to propose a new topic of conversation. In this manner we also demonstrate how SAASA data can impact the performance of this task and highlight what combination DA and/or SAASA data yields the best results.

In this proof of concept however we were limited by a relatively small and imbalanced dataset. Taking this into account we selected performance metrics accordingly (including learning curves for each classifier) in order to ensure that the testing of each classifier could be correctly compared. Following the experimental results, we successfully demonstrate not only that SAASA data improves the performance of XGBoost binary classifiers for this task but also that, for both the DA level and Turn level datasets, utilizing the SAASA data alone, without the DA utterance text, as input features resulted in better performance results for this task as well as better generalising classifiers for future predictions.

In the following subsections we reiterate the main contributions of this thesis, then discuss potential next steps and work for this research.

8.1 Main Contributions

Below we reiterate the main contributions of the thesis:

1. A Topic Handler model for human-computer dialogue leveraging linguistic methods and theories. The proposed framework is designed in such a way that is flexible and compatible with third party dialogue systems. In particular, we demonstrate the Topic Handler model's compatibility with the work in [36] which proposes a comprehensive model for conversation handling.

2. A proof of concept for main modules in framework by implementing:

   a) Feature engineering of dialogue act and semantic related data for input in machine learning module

   b) Machine learning experimentation to demonstrate the positive impact of using speech act and semantic annotation data on identifying appropriate locations for an NPC/virtual agent to propose a new topic in task-oriented dialogues

   c) Machine learning experimentation to show potential architecture for effective implementation of proposed Topic Handler model

To our knowledge this is the only framework that leverages linguistic methods and theories for topic handling in human-computer dialogue systems. Our experimentation shows that this framework is particularly well suited for the task-oriented genre however we believe it to be equally applicable to other genres and will establish this in experimentation in future work.

8.2 Future Work

For future work, there are two main areas we would choose to explore next however would require more SAASA annotated data:

1. Derive additional topic labels for a multi-label classifier that would offer even more precise topic handling by the Topic Handler model.

2. Test performance of additional algorithms for use in M5. In particular, test performance when using LSTM and BERT neural networks that maintain sequence/order of each feature within the Turn level dataset.

In the following Subsections below, we discuss these areas in more detail.

8.2.1 Additional Prediction Labels for Multi-Label Classifier

Additional predictive labels could be derived from above features described in Chapter 5 and allow for even more detailed predictive outputs. Due to limited volume of data in our corpus, we did not do this in our above experimentation; however if a larger dataset was readily available, the additional derived prediction labels listed in Table 21, in **bold**, could be used to create a multi-label classifier that could allow for more cohesive discussions with NPCs across longer dialogue exchanges. In this instance, instead of only the "Next DA New Topic" label being predicted by M5, once M5 assesses the patterns found in the M4, the module will output whether or not the next turn's initial dialogue act topic would instead be:

1) a new topic

1.1) a new topic that is related to the current topic

1.2) the reintroduction of a previously discussed in conversation

1.3) the reintroduction of a related topic to the current topic that was already discussed in the current conversation

2) the last DA of the current topic (i.e. the initial DA wraps up the current topic)

Options 1.2, 1.3 and 1.4 above are a subset of the 'new topic' option as in this model we define any change in topic in the next turn as a new topic. The Topic Handler then outputs its choice to M6.

Prediction Feature	Description
Next DA New Topic (y/n flag)	A topic is only annotated yes/1 when a new topic is introduced that differs from the previous dialogue act's topic.
Next DA Last Speech act in Topic (y/n flag)	If the dialogue act is the last dialogue act for the current topic, this feature is flagged as yes/1.
Next DA Revisited Topic Reintroduction (y/n flag)	As topics are introduced in each conversation, they are logged in a list called "topic base", if the new topic introduced in the dialogue act is in the "topic base", this feature is flagged as yes/1. This topic however does not necessarily relate to the current topic.
Next DA Related Topic of Current Topic Introduced (y/n flag)	A topic is related to another topic if the topics have overlapping 'sub-topics'. Sub-topics are derived from topics by tokenizing the topics, separating tokens by '-', and are logged in a list "topic base breakout" (i.e a sub topic base) as each dialogue act and topic are processed. If the introduced topic has an overlapping sub-topic with the current topic of the previous dialogue act, then this feature is flagged as yes/1. Example of related topics are "to-number-location-journey-from" and "journey-day", with sub-topic "journey" relating the two topics. If these were the topics of two subsequent dialogue acts, this feature would be flagged as yes/1.
Next DA Revisited Related Topic Reintroduction (y/n flag)	If new topic relates to previously discussed topics, this feature is flagged as yes/1. In other words, if sub-topics of dialogue act's topic have been previously discussed (i.e. has an overlapping sub-topic with a previously introduced topic, this feature is flagged as yes/1; if new topic is already in topic base, this feature as well as the above Revisited Topic Reintroduction would be flagged as yes/1).

Table 21: Derived Topic Labels Set 2 (i.e. Additional Prediction Labels)

8.2.2 Testing Neural Network Based M5 Topic Trainer to Predict on Turn Level Dataset

In future work, we could also explore performance of additional machine learning algorithms. In particular, we would like to understand how algorithms such as LSTM and BERT could improve the performance of identifying points at which new topics could be introduced. Such algorithms could more easily allow for a multi-label classifier for M5 and are able to maintain and leverage sequence patterning of SAASA data in the Turn level dataset. As previously mentioned in Section 7.4.1, these algorithms would require a significantly larger dataset for training and therefore require additional annotated and labeled samples to avoid overfitting.

8.3 Conclusion

In this thesis, we aim to contribute to ongoing research in the field of human-computer dialogue and help move closer to the goal of having more realistic human-computer dialogue. We address the current challenge of topic handling in human-computer conversation by proposing a Topic Handler model that is compatible with existing and future dialogue systems. The proposed Topic Handler model builds off of previously proposed dialogue grammars and system architecture and is based on speech act theory and conversation analysis.

The framework of the Topic Handler contains six modules, two of which we use to demonstrate proof of concept through experimentation. By employing feature engineering of existing dialogue act corpora and using this data in machine learning experimentation, we successfully demonstrate not only that SAASA data improves the performance of XGBoost binary classifiers for the task of topic handling, particularly for task-oriented dialogues, but also shows how the proposed Topic Handler model can provide needed inputs to assist in topic handling when used in parallel with a dialogue system. Further research and investigation, such as mentioned in the previous section, could further help contribute to the goal of topic handling and a successful implementation of this Topic Handler model in existing dialogue systems at scale.

9 Appendix

9.1 Example trainline35.xml dialogue

Listing 1: trainline35.xml from SPAADIA corpus

```xml
<?xml version="1.0"?>
<dialogue corpus="trainline" lang="en" id="35">
<turn n="1" speaker="A">
<frag n="1" sp-act="greet" mode="greet-opening">
good afternoon <punc type="stop" />
</frag>
<frag n="2" sp-act="refer" polarity="positive" mode="frag">
Virgin Trainlines <punc type="level" />
</frag>
<frag n="3" sp-act="identifySelf" polarity="positive" mode="intro-partial-decl">
Sandra speaking <punc type="stop" />
</frag>
</turn>
<turn n="2" speaker="B">
<dm n="4" sp-act="hesitate">
em
</dm>
<frag n="5" sp-act="greet" mode="greet-opening">
hi <punc type="stop" />
</frag>
<dm n="6" sp-act="hesitate">
em
</dm>
<decl n="7" sp-act="state" polarity="positive" topic="enum-number" mode="report-decl">
i've just been in touch with the 0 3 4 5 1 2 1 9 6 2 number <punc type="stop" />
</decl>
<decl n="8" sp-act="state" polarity="positive" topic="fare-number-location-enum" mode="report-decl">
and we've just established em two train times on the 19 pounds going through Rugby Super Saver Virgin <punc type="stop" />
</decl>
</turn>
<turn n="3" speaker="A">
<frag n="9" sp-act="reqInfo" polarity="positive" topic="journey" mode="query">
travelling from <punc type="query" />
</frag>
```

```
</turn>
<turn n="4" speaker="B">
<frag n="10" sp-act="answer-refer" polarity="positive" topic="to-location" mode="frag">
Euston to erm <pause /> er Macclesfield <punc type="level" />
</frag>
<frag n="11" sp-act="elab-refer" polarity="positive" topic="to-location-from" mode="decl">
and then back from Wilmslow to Euston <punc type="stop" />
</frag>
<decl n="12" sp-act="state" polarity="positive" topic="location" mode="exists-decl">
and i've got all the times here and everything <punc type="stop" />
</decl>
</turn>
<turn n="5" speaker="A">
<dm n="13" sp-act="acknowledge">
right <pause />
</dm>
<dm n="14" sp-act="init">
now <punct type="level" />
</dm>
<q-yn n="15" sp-act="reqInfo" polarity="positive" topic="creditcard" mode="alternative-closed-query">
<pause /> do you hold a current debit or credit card <punc type="query" />
</q-yn>
</turn>
<turn n="6" speaker="B">
<decl n="16" sp-act="answer-state" polarity="positive" mode="decl">
i do <punc type="stop" />
</decl>
</turn>
<turn n="7" speaker="A">
<q-wh n="17" sp-act="reqInfo" polarity="positive" topic="journey-date" mode="open-query">
and what date is it you're travelling <punc type="query" />
</q-wh>
</turn>
<turn n="8" speaker="B">
<decl n="18" sp-act="answer-state" polarity="positive" mode="decl">
the 17th <punc type="stop" />
</decl>
```

```
<frag n="19" sp-act="elab-refer" polarity="positive" topic="month-date-day" mode="partial-decl">
for er Thursd... Saturday the 17th of October <punc type="stop" />
</frag>
</turn>
<turn n="9" speaker="A">
<decl n="20" sp-act="reqInfo" polarity="positive" topic="number" mode="query">
and it's for 2 <punc type="query" />
</decl>
</turn>
<turn n="10" speaker="B">
<decl n="21" sp-act="answer-state" polarity="positive" topic="number" mode="decl">
it's for 1 <punc type="stop" />
</decl>
</turn>
<turn n="11" speaker="A">
<frag n="22" sp-act="echo-refer" polarity="positive" topic="number" mode="partial-decl">
for 1 <punc type="stop" />
</frag>
</turn>
<turn n="12" speaker="B">
<yes n="23" sp-act="acknowledge">
yes <punc type="stop" />
</yes>
</turn>
<turn n="13" speaker="A">
<dm n="24" sp-act="init">
<pause length="2s" /> now <punct type="level" />
</dm>
<decl n="25" sp-act="stateCondition" polarity="positive" topic="fare-journey" mode="condition-frag">
if you're actually buying a return ticket <punc type="level" />
</decl>
<decl n="26" sp-act="stateConstraint" polarity="positive" topic="fare" mode="constrain-decl">
you must return from the station <backchannel content="yes" /> that you get off at the destination <punc type="stop" />
</decl>
</turn>
<turn n="14" speaker="B">
<dm n="27" sp-act="exclaim">
oh
</dm>
<dm n="28" sp-act="acknowledge">
```

```
right <punc type="stop" />
</dm>
</turn>
<turn n="15" speaker="A">
<decl n="29" sp-act="confirm-expressImPossibility" polarity="negative" topic="to-location-from" mode="poss2-decl">
you can't go from Euston to Macclesfield then Wilmslow to Euston <punc type="stop" />
</decl>
</turn>
<turn n="16" speaker="B">
<dm n="30" sp-act="exclaim">
<overlap type="start" /> oh <punc type="stop" />
</dm>
</turn>
<turn n="17" speaker="A">
<decl n="31" sp-act="suggest" polarity="positive" topic="fare" mode="condition-poss2-predict-poss1-decl">
what <overlap type="end" /> we could try and do is see if you could get you Singles to see if it's going to work out to the same price <punc type="stop" />
</decl>
</turn>
<turn n="18" speaker="B">
<dm n="32" sp-act="acknowledge" mode="tag">
ok <punc type="stop" />
</dm>
</turn>
<turn n="19" speaker="A">
<decl n="33" sp-act="state" polarity="positive" topic="journey" mode="decl">
which will enable you use that journey that way <punc type="stop" />
</decl>
</turn>
<turn n="20" speaker="B">
<dm n="34" sp-act="acknowledge" mode="tag">
ok <punc type="stop" />
</dm>
</turn>
<turn n="21" speaker="A">
<dm n="35" sp-act="init">
now <punct type="level" />
</dm>
<q-wh n="36" sp-act="reqDirect" polarity="positive" topic="time-day" mode="preference2-open-query">
```

```
what time on the Saturday do you want to depart <punc type="query" />
</q-wh>
</turn>
<turn n="22" speaker="B">
<frag n="37" sp-act="direct-refer" polarity="positive" mode="decl">
10:30 <punc type="stop" />
</frag>
<decl n="38" sp-act="elab-stateReason" polarity="positive" topic="fare-enum" mode="report-reason-decl">
<pause length="4s" /> cos i was quoted 19 pounds <punc type="stop" />
</decl>
<decl n="39" sp-act="approve" polarity="positive" mode="decl">
which is <overlap type="start" /> great <punc type="stop" />
</decl>
</turn>
<turn n="23" speaker="A">
<frag n="40" sp-act="reqConfirm" polarity="positive" topic="fare" mode="query">
the <overlap type="end" /> return fare <punc type="query" />
</frag>
</turn>
<turn n="24" speaker="B">
<yes n="41" sp-act="confirm-acknowledge">
<overlap type="start" /> yes <punc type="stop" />
</yes>
</turn>
<turn n="25" speaker="A">
<dm n="42" sp-act="init">
now <punct type="level" />
</dm>
<decl n="43" sp-act="state" polarity="negative" topic="time-day" mode="decl">
<overlap type="end" /> the trains don't run at half past the hour on a Saturday <punc type="stop" />
</decl>
</turn>
<turn n="26" speaker="B">
<dm n="44" sp-act="acknowledge">
right <punc type="stop" />
</dm>
</turn>
<turn n="27" speaker="A">
<decl n="45" sp-act="state" polarity="positive" topic="time-enum" mode="decl">
```

```xml
it's 10 to the hour <punc type="stop" />
</decl>
<decl n="46" sp-act="state" polarity="positive" mode="decl">
so it's 10:50 <punc type="stop" />
</decl>
</turn>
<turn n="28" speaker="B">
<dm n="47" sp-act="acknowledge">
right <punc type="stop" />
</dm>
</turn>
<turn n="29" speaker="A">
<decl n="48" sp-act="state" polarity="negative" topic="fare-availability-location" mode="exists-decl">
and there's not any availability there for the Virgin Value fare <punc type="stop" />
</decl>
</turn>
<turn n="30" speaker="B">
<q-yn n="49" sp-act="reqModal-abandon" polarity="positive" mode="tag-closed-abandon">
would it <punc type="incomplete" />
</q-yn>
<q-yn n="50" sp-act="reqInfo" polarity="positive" topic="number-time" mode="exists-closed-query-disflu">
do they do they have an earlier one where there is a Value f... <pause /> thing <punc type="query" />
</q-yn>
</turn>
<turn n="31" speaker="A">
<frag n="51" sp-act="answer-refer" polarity="positive" topic="time-enum" mode="decl">
<pause length="4s" /> 9:45 in the morning <punc type="stop" />
</frag>
</turn>
<turn n="32" speaker="B">
<dm n="52" sp-act="exclaim">
oh
</dm>
<frag n="53" sp-act="reqInfo" polarity="positive" topic="time" mode="query">
<pause /> anything a bit later <punc type="query" />
</frag>
<decl n="54" sp-act="stateReason" polarity="positive" topic="day" mode="reason-decl-disflu">
<pause /> cos it it's anything that's that's possible at all at that day <punc type="stop" />
```

```
</decl>
</turn>
<turn n="33" speaker="A">
<decl n="55" sp-act="answer-state" polarity="positive" topic="location-enum-departure" mode="frag">
13:50 departing Euston <backchannel content="yeah" /> <punc type="level" />
</decl>
<frag n="56" sp-act="elab-state" polarity="positive" topic="time-arrival" mode="decl">
arriving in Macclesfield at 14:03 <punc type="stop" />
</frag>
</turn>
<turn n="34" speaker="B">
<dm n="57" sp-act="acknowledge" mode="tag">
ok <punc type="stop" />
</dm>
</turn>
<turn n="35" speaker="A">
<dm n="58" sp-act="apologise" mode="regret">
sorry <punc type="stop" />
</dm>
<frag n="59" sp-act="correctSelf" polarity="positive" mode="decl">
16:03 <punc type="stop" />
</frag>
</turn>
<turn n="36" speaker="B">
<dm n="60" sp-act="acknowledge">
right <punc type="stop" />
</dm>
<dm n="61" sp-act="acknowledge" mode="tag">
ok <punc type="stop" />
</dm>
<dm n="62" sp-act="init">
<pause length="2s" /> well <punct type="level" />
</dm>
<imp n="63" sp-act="suggest" polarity="positive" mode="decl">
let's go with that then to for the <unclear length="8 syllables" /> <punc type="stop" />
</imp>
<decl n="64" sp-act="reqConfirm" polarity="positive" mode="query">
<pause length="3s" /> that's Macclesfield <punc type="query" />
</decl>
</turn>
<turn n="37" speaker="A">
<dm n="65" sp-act="confirm-acknowledge" mode="tag">
ok <punc type="stop" />
```

```
</dm>
</turn>
<turn n="38" speaker="B">
<yes n="66" sp-act="acknowledge">
yeah <punc type="stop" />
</yes>
</turn>
<turn n="39" speaker="A">
<decl n="67" sp-act="state" polarity="positive" topic="fare-enum"
    mode="decl">
and that's 9 pounds 50 <punc type="stop" />
</decl>
<dm n="68" sp-act="init">
now <punct type="level" />
</dm>
<decl n="69" sp-act="stateIntent-hold" polarity="positive" topic="
    fare-verify" mode="intent-report-hold-decl">
i'm just going to check <pause /> your return <punc type="stop" />
</decl>
</turn>
<turn n="40" speaker="B">
<dm n="70" sp-act="acknowledge" mode="tag">
ok <punc type="stop" />
</dm>
</turn>
<turn n="41" speaker="A">
<frag n="71" sp-act="reqConfirm" polarity="positive" topic="to-
    location-from" mode="partial-query">
from Wilmslow to Euston <punc type="query" />
</frag>
</turn>
<turn n="42" speaker="B">
<yes n="72" sp-act="confirm-acknowledge">
yeah <punc type="stop" />
</yes>
</turn>
<turn n="43" speaker="A">
<q-wh n="73" sp-act="reqInfo" polarity="positive" topic="date-
    return" mode="open-query">
what date is it you're returning <punc type="query" />
</q-wh>
</turn>
<turn n="44" speaker="B">
<frag n="74" sp-act="answer-refer" polarity="positive" topic="day"
    mode="decl-disflu">
Su... Sunday the 18th <punc type="stop" />
</frag>
```

```
</turn>
<turn n=”45” speaker=”A”>
<frag n=”75” sp-act=”reqInfo” polarity=”positive” topic=”time-departure” mode=”open-query”>
<pause length=”2s” /> departing on what time around <punc type=”query” />
</frag>
</turn>
<turn n=”46” speaker=”B”>
<dm n=”76” sp-act=”hesitate”>
erm
</dm>
<frag n=”77” sp-act=”answer-refer” polarity=”positive” topic=”time-enum” mode=”alternative-decl”>
you kn... after sort of like 4 o’clockish or something <punc type=”stop” />
</frag>
<decl n=”78” sp-act=”elab-state” polarity=”positive” topic=”availability” mode=”decl”>
<pause /> whatever’s available really <punc type=”stop” />
</decl>
</turn>
<turn n=”47” speaker=”A”>
<dm n=”79” sp-act=”init”>
<pause length=”14s” /> now <punct type=”level” />
</dm>
<decl n=”80” sp-act=”expressRegret” polarity=”positive” topic=”to-location” mode=”regret-frag”>
i’m afraid on the Wilmslow to Euston route <punc type=”level” />
</decl>
<decl n=”81” sp-act=”state” polarity=”negative” topic=”fare” mode=”exists-decl”>
there’s not any Virgin Value fares <punc type=”stop” />
</decl>
</turn>
<turn n=”48” speaker=”B”>
<frag n=”82” sp-act=”reqConfirm” polarity=”positive” mode=”partial-query”>
at all <punc type=”query” />
</frag>
</turn>
<turn n=”49” speaker=”A”>
<no n=”83” sp-act=”confirm-negate”>
no <punc type=”stop” />
</no>
</turn>
<turn n=”50” speaker=”B”>
```

```
<q-wh n="84" sp-act="suggest-reqInfo" polarity="positive" mode="closed-query">
what about erm <pause /> Macclesfield <pause /> back <punc type="query" />
</q-wh>
</turn>
<turn n="51" speaker="A">
<decl n="85" sp-act="expressRegret" polarity="negative" topic="availability-day" mode="regret-exists-decl">
<pause length="22s" /> i'm afraid there's not any available for the Sunday <punc type="stop" />
</decl>
</turn>
<turn n="52" speaker="B">
<frag n="86" sp-act="reqConfirm" polarity="negative" mode="query">
nothing <punc type="query" />
</frag>
</turn>
<turn n="53" speaker="A">
<no n="87" sp-act="confirm-negate">
no <punc type="stop" />
</no>
<decl n="88" sp-act="elab-state" polarity="positive" mode="exists-decl">
there's engineering works going on at the moment <punc type="stop" />
</decl>
<decl n="89" sp-act="stateConstraint" polarity="positive" topic="to-location-from" mode="constrain-decl">
so you need to get <pause /> the bus from Macclesfield to Wilmslow then Wilmslow to Euston <punc type="stop" />
</decl>
<decl n="90" sp-act="state" polarity="negative" topic="fare" mode="exists-constrain-decl">
but there's not any allocation for the Virgin Value fare <punc type="stop" />
</decl>
</turn>
<turn n="54" speaker="B">
<dm n="91" sp-act="exclaim">
ah
</dm>
<q-wh n="92" sp-act="reqInfo" polarity="positive" topic="fare" mode="tag-poss1-query">
<pause /> so what's the cheapest fare i could do it <punc type="query" />
</q-wh>
```

```
</turn>
<turn n="55" speaker="A">
<frag n="93" sp-act="stateIntent-hold" polarity="positive" topic="verify" mode="benefit-decl">
<pause /> just check that for you <punc type="stop" />
</frag>
</turn>
<turn n="56" speaker="B">
<dm n="94" sp-act="phatic">
<pause length="3s" /> i mean
</dm>
<decl n="95" sp-act="expressOpinion" polarity="negative" mode="alternative-condition-decl">
it doesn't it really doesn't matter whether i go into Macclesfield or Wilmslow <punc type="stop" />
</decl>
</turn>
<turn n="57" speaker="A">
<q-yn n="96" sp-act="reqInfo-abandon" polarity="positive" topic="enum" mode="closed-interruption">
<pause length="3s" /> but is the 13:50 train still suitable on the <punc type="incomplete" />
</q-yn>
</turn>
<turn n="58" speaker="B">
<yes n="97" sp-act="answer-acknowledge">
<pause length="2s" /> yeah <punc type="stop" />
</yes>
<frag n="98" sp-act="elab-abandon" polarity="positive" mode="abandon">
every... <punc type="incomplete" />
</frag>
<yes n="99" sp-act="acknowledge">
yeah <punc type="stop" />
</yes>
<dm n="100" sp-act="approve">
fine <punc type="stop" />
</dm>
</turn>
<turn n="59" speaker="A">
<frag n="101" sp-act="refer" polarity="positive" topic="day" mode="frag">
Saturday <punc type="level" />
</frag>
<decl n="102" sp-act="hold" polarity="positive" topic="verify-day" mode="hold-decl">
```

```
i 'm just checking Sunday <overlap type="start" /> now <punc type="stop" />
</decl>
</turn>
<turn n="60" speaker="B">
<frag n="103" sp-act="thank" polarity="positive" mode="thank-decl">
thank you <overlap type="end" /> <punc type="stop" />
</frag>
</turn>
<turn n="61" speaker="A">
<dm n="104" sp-act="init">
<pause length="10s" /> now <punct type="level" />
</dm>
<frag n="105" sp-act="refer" polarity="positive" topic="day" mode="partial-frag">
on the Sunday <backchannel content="m" /> <punc type="level" />
</frag>
<decl n="106" sp-act="state" polarity="positive" mode="exists-decl">
there 's a train <punc type="stop" />
</decl>
<dm n="107" sp-act="init">
well <punct type="level" />
</dm>
<dm n="108" sp-act="apologise" mode="regret">
sorry <punc type="stop" />
</dm>
<decl n="109" sp-act="refer" polarity="positive" topic="time-from" mode="decl">
the bus from Macclesfield at 15:55 <punc type="stop" />
</decl>
</turn>
<turn n="62" speaker="B">
<yes n="110" sp-act="acknowledge">
yeah <punc type="stop" />
</yes>
</turn>
<turn n="63" speaker="A">
<decl n="111" sp-act="state" polarity="positive" topic="arrival" mode="decl">
you arrive in Wilmslow for 16:25 <punc type="stop" />
</decl>
<decl n="112" sp-act="state" polarity="positive" topic="time-from" mode="decl">
<pause /> then it 's a train from Wilmslow at 16:46 <punc type="stop" />
```

```
</decl>
</turn>
<turn n="64" speaker="B">
<dm n="113" sp-act="exclaim">
oh
</dm>
<dm n="114" sp-act="apologise">
sorry <punc type="stop" />
</dm>
<imp n="115" sp-act="hold" polarity="positive" mode="hold-decl">
hold the line <punc type="stop" />
</imp>
</turn>
<turn n="65" speaker="A">
<yes n="116" sp-act="acknowledge">
yeah <punc type="stop" />
</yes>
</turn>
<turn n="66" speaker="B">
<decl n="117" sp-act="state" polarity="positive" mode="decl">
<pause length="14s" /> i 'm a receptionist <vocal content="laughter" /> <punc type="stop" />
</decl>
</turn>
<turn n="67" speaker="A">
<dm n="118" sp-act="init">
now <punct type="level" />
</dm>
<q-yn n="119" sp-act="reqInfo" polarity="positive" topic="railcard" mode="closed-query">
do you have a railcard at all <punc type="query" />
</q-yn>
</turn>
<turn n="68" speaker="B">
<dm n="120" sp-act="hesitate">
erm
</dm>
<no n="121" sp-act="answer-negate">
no <punc type="stop" />
</no>
</turn>
<turn n="69" speaker="A">
<no n="122" sp-act="echo-negate">
no <punc type="stop" />
</no>
<dm n="123" sp-act="hesitate">
erm
```

```
</dm>
<dm n="124" sp-act="init">
well <punct type="level" />
</dm>
<decl n="125" sp-act="predict" polarity="positive" topic="fare-enum-availability" mode="decl">
the cheapest fare available is going to be the Apex at 26 pounds 50 return <punc type="stop" />
</decl>
<frag n="126" sp-act="refer" polarity="positive" topic="fare" mode="frag">
<pause /> and on the return <punc type="level" />
</frag>
<frag n="127" sp-act="state" polarity="positive" mode="report-frag">
like i was saying <punc type="level" />
</frag>
<decl n="128" sp-act="state" polarity="positive" topic="to-from-departure" mode="decl">
<pause /> it's a bus from Macclesfield to Wilmslow departing 15:55 <punc type="stop" />
</decl>
</turn>
<turn n="70" speaker="B">
<decl n="129" sp-act="approve" polarity="positive" mode="reassurance-decl">
that's fine <punc type="stop" />
</decl>
</turn>
<turn n="71" speaker="A">
<decl n="130" sp-act="state" polarity="positive" topic="time-from" mode="frag">
then it's a train from Wilmslow at 16:46 <punc type="level" />
</decl>
</turn>
<turn n="72" speaker="B">
<yes n="131" sp-act="acknowledge">
yes <punc type="stop" />
</yes>
</turn>
<turn n="73" speaker="A">
<frag n="132" sp-act="state" polarity="positive" topic="location-time-arrival" mode="decl">
arriving in Euston at 19:13 <punc type="stop" />
</frag>
</turn>
<turn n="74" speaker="B">
```

```
<dm n="133" sp-act="acknowledge" mode="tag">
ok <punc type="stop" />
</dm>
</turn>
<turn n="75" speaker="A">
<decl n="134" sp-act="state" polarity="positive" topic="time-day"
    mode="decl">
and that's going out on the Saturday the 17th at 13:50 <punc type=
    "stop" />
</decl>
</turn>
<turn n="76" speaker="B">
<dm n="135" sp-act="init">
now <punct type="level" />
</dm>
<q-yn n="136" sp-act="reqOpt" polarity="positive" topic="time-day"
    mode="closed-query">
can i go earlier on the Saturday <punc type="query" />
</q-yn>
<imp n="137" sp-act="hold" polarity="positive" mode="hold-decl">
<pause /> just hold the line <punc type="stop" />
</imp>
<dm n="138" sp-act="apologise" mode="regret">
sorry <punc type="stop" />
</dm>
</turn>
<turn n="77" speaker="A">
<yes n="139" sp-act="stateOpt-acknowledge">
yeah <punc type="stop" />
</yes>
<dm n="140" sp-act="agree" mode="reassurance-tag">
<pause /> that's ok <punc type="stop" />
</dm>
<decl n="141" sp-act="state" polarity="positive" topic="time-enum"
    mode="decl">
it's the 7:45 in the morning <punc type="stop" />
</decl>
<decl n="142" sp-act="state" polarity="positive" topic="
    availability" mode="decl">
it's the best available <punc type="stop" />
</decl>
</turn>
<turn n="78" speaker="B">
<frag n="143" sp-act="reqInfo" polarity="negative" topic="time-
    enum" mode="exists-constrain-query">
but abouts there's nothing at 8 o'clockish <punc type="query" />
</frag>
```

```
</turn>
<turn n="79" speaker="A">
<no n="144" sp-act="answer-negate">
no <punc type="stop" />
</no>
<decl n="145" sp-act="elab-state" polarity="positive" topic="availability" mode="decl">
they've all now being booked up <punc type="stop" />
</decl>
</turn>
<turn n="80" speaker="B">
<frag n="146" sp-act="reqConfirm" polarity="positive" mode="query">
07:45 <punc type="query" />
</frag>
</turn>
<turn n="81" speaker="A">
<yes n="147" sp-act="confirm-acknowledge">
<overlap type="start" /> yes <punc type="stop" />
</yes>
</turn>
<turn n="82" speaker="B">
<dm n="148" sp-act="exclaim">
oh God <overlap type="end" /> <punc type="stop" />
</dm>
<q-wh n="149" sp-act="reqInfo" polarity="" topic="" mode="">
<vocal content="laughter" /> what
</q-wh>
<decl n="150" sp-act="reqInfo" polarity="positive" topic="number" mode="query">
even to to any one of those stations <punc type="query" />
</decl>
</turn>
<turn n="83" speaker="A">
<frag n="151" sp-act="answer-echo-refer" polarity="positive" topic="number" mode="decl">
even to any one of those stations <punc type="stop" />
</frag>
</turn>
<turn n="84" speaker="B">
<dm n="152" sp-act="exclaim">
oh
</dm>
<no n="153" sp-act="negate-exclaim">
no <punc type="stop" />
</no>
```

```
<decl n="154" sp-act="reqConfirm" polarity="positive" topic="fare-enum" mode="query">
<pause /> and that would be 26 pounds 50 <punc type="query" />
</decl>
</turn>
<turn n="85" speaker="A">
<dm n="155" sp-act="confirm-acknowledge">
aha <punc type="stop" />
</dm>
</turn>
<turn n="86" speaker="B">
<q-yn n="156" sp-act="reqInfo-abandon" polarity="positive" topic="location" mode="exists-closed-abandon">
is there anything a little <punc type="incomplete" />
</q-yn>
<q-yn n="157" sp-act="reqInfo" polarity="positive" topic="availability-location" mode="exists-closed-query">
is there anything available more expensively <punc type="query" />
</q-yn>
<decl n="158" sp-act="stateCondition" polarity="positive" mode="condition-decl">
if you see what i mean <punc type="stop" />
</decl>
<decl n="159" sp-act="reqInfo" polarity="positive" topic="time" mode="query-query">
<pause /> that's at the right time what <punc type="query" />
</decl>
</turn>
<turn n="87" speaker="A">
<dm n="160" sp-act="init">
well <punct type="level" />
</dm>
<frag n="161" sp-act="answer-refer" polarity="positive" topic="enum" mode="partial-decl">
on the 10:50 train <punc type="stop" />
</frag>
</turn>
<turn n="88" speaker="B">
<yes n="162" sp-act="acknowledge">
yeah <punc type="stop" />
</yes>
</turn>
<turn n="89" speaker="A">
<frag n="163" sp-act="state" polarity="positive" topic="time-arrival" mode="decl">
arriving at 13:30 <punc type="stop" />
</frag>
```

```
<decl n=”164” sp−act=”state” polarity=”positive” topic=”day” mode=”decl”>
that’s on Saturday the 17th <punc type=”stop” />
</decl>
</turn>
<turn n=”90” speaker=”B”>
<yes n=”165” sp−act=”answer−acknowledge”>
yeah <punc type=”stop” />
</yes>
</turn>
<turn n=”91” speaker=”A”>
<decl n=”166” sp−act=”state” polarity=”positive” topic=”fare−availability” mode=”decl”>
the Super Advance is available <punc type=”stop” />
</decl>
<decl n=”167” sp−act=”state” polarity=”positive” topic=”fare−enum” mode=”decl”>
and that’s 30 pounds return <punc type=”stop” />
</decl>
</turn>
<turn n=”92” speaker=”B”>
<dm n=”168” sp−act=”exclaim”>
oh
</dm>
<decl n=”169” sp−act=”approve−reqConfirm” polarity=”negative” mode=”tag−query”>
that’s not bad is it <punc type=”query” />
</decl>
</turn>
<turn n=”93” speaker=”A”>
<decl n=”170” sp−act=”confirm−state” polarity=”positive” topic=”enum−fare−location” mode=”decl”>
that’s for an extra 3 pounds 50 <overlap type=”start” /> there <punc type=”stop” />
</decl>
</turn>
<turn n=”94” speaker=”B”>
<dm n=”171” sp−act=”init”>
well <punct type=”level” />
</dm>
<yes n=”172” sp−act=”acknowledge”>
<overlap type=”end” /> yeah <punc type=”stop” />
</yes>
<decl n=”173” sp−act=”approve” polarity=”positive” mode=”reassurance−decl”>
that’s fine <punc type=”stop” />
</decl>
```

```
<dm n="174" sp-act="hesitate">
erm <punc type="stop" />
</dm>
</turn>
<turn n="95" speaker="A">
<decl n="175" sp-act="predict" polarity="positive" topic="enum-day" mode="predict-decl">
and you'll be able to get the 10:50 going out on the <pause /> Saturday the 17th <punc type="stop" />
</decl>
</turn>
<turn n="96" speaker="B">
<dm n="176" sp-act="acknowledge" mode="tag">
ok <punc type="stop" />
</dm>
</turn>
<turn n="97" speaker="A">
<decl n="177" sp-act="state" polarity="positive" topic="time-enum-arrival" mode="decl">
and you arrive at 13:03 in Macclesfield <punc type="stop" />
</decl>
</turn>
<turn n="98" speaker="B">
<decl n="178" sp-act="reqConfirm" polarity="positive" mode="query">
that's thir... 13:03 <punc type="query" />
</decl>
</turn>
<turn n="99" speaker="A">
<yes n="179" sp-act="confirm-acknowledge">
yes <punc type="stop" />
</yes>
<frag n="180" sp-act="elab-refer" polarity="positive" mode="decl">
13:03 <punc type="stop" />
</frag>
</turn>
<turn n="100" speaker="B">
<dm n="181" sp-act="acknowledge" mode="tag">
ok <punc type="stop" />
</dm>
</turn>
<turn n="101" speaker="A">
<frag n="182" sp-act="state" polarity="positive" topic="return" mode="frag">
and returning <punc type="level" />
</frag>
<decl n="183" sp-act="state" polarity="positive" mode="decl">
```

```
it's still the 15:55 <punc type="stop" />
</decl>
</turn>
<turn n="102" speaker="B">
<yes n="184" sp-act="acknowledge">
yeah <punc type="stop" />
</yes>
</turn>
<turn n="103" speaker="A">
<frag n="185" sp-act="refer" polarity="positive" topic="to-from" mode="partial-decl">
from Macclesfield to Wilmslow <punc type="stop" />
</frag>
<decl n="186" sp-act="state" polarity="positive" topic="to-location-from-enum" mode="frag">
and it's the 16:46 from Wilmslow to Euston <punc type="level" />
</decl>
<frag n="187" sp-act="state" polarity="positive" topic="time-arrival" mode="decl">
<pause /> arriving at 19:13 <punc type="stop" />
</frag>
</turn>
<turn n="104" speaker="B">
<decl n="188" sp-act="approve" polarity="positive" mode="decl">
<pause /> that sounds ideal <punc type="stop" />
</decl>
</turn>
<turn n="105" speaker="A">
<q-yn n="189" sp-act="reqDirect" polarity="positive" topic="booking" mode="benefit-preference2-closed-query">
do you want me to book that for you <punc type="query" />
</q-yn>
</turn>
<turn n="106" speaker="B">
<yes n="190" sp-act="direct-acknowledge">
yeah <punc type="stop" />
</yes>
<q-yn n="191" sp-act="reqOpt" polarity="positive" mode="closed-query">
le... can i just sorry just get a pen <punc type="query" />
</q-yn>
<frag n="192" sp-act="hold" polarity="positive" mode="decl">
just going to phone my sister <punc type="stop" />
</frag>
<frag n="193" sp-act="stateIntent-hold" polarity="positive" topic="verify" mode="reassurance-tag-decl">
just check that ev... that's alright <punc type="stop" />
```

```
</frag>
<decl n="194" sp-act="stateReason" polarity="positive" mode="intent-reason-decl">
cos that's who i'm going to go and stay with <punc type="stop" />
</decl>
<frag n="195" sp-act="refer" polarity="positive" topic="enum" mode="decl">
6 em 2 6 6 6 6 8 <punc type="stop" />
</frag>
<decl n="196" sp-act="expressPossibility-abandon" polarity="positive" mode="abandon-poss1-abandon">
and then i can just ca... <punc type="incomplete" />
</decl>
<dm n="197" sp-act="init">
you know <punc type="stop" />
</dm>
<decl n="198" sp-act="predict" polarity="positive" mode="predict-decl-disflu">
then it's then it'll be great <punc type="stop" />
</decl>
<decl n="199" sp-act="expressPossibility" polarity="positive" mode="poss1-decl">
then i can just <overlap type="start" /> <unclear length="6 syllables" /> <punc type="stop" />
</decl>
</turn>
<turn n="107" speaker="A">
<dm n="200" sp-act="init">
now <punct type="level" />
</dm>
<q-yn n="201" sp-act="reqDirect" polarity="positive" topic="booking" mode="preference2-closed-query">
<overlap type="end" /> do you want me to book the seat just now <punc type="query" />
</q-yn>
<q-yn n="202" sp-act="reqInfo" polarity="positive" topic="seat" mode="alternative-closed-query">
is it smoking or non-smoking <punc type="query" />
</q-yn>
</turn>
<turn n="108" speaker="B">
<frag n="203" sp-act="direct-refer" polarity="positive" topic="seat" mode="poss1-condition-decl">
non-smoking window if i could <punc type="stop" />
</frag>
<frag n="204" sp-act="elab-refer" polarity="positive" topic="journey-seat" mode="decl">
```

```
sort of facing you know the <overlap type="start" /> the direction of travel <punc type="stop" />
</frag>
</turn>
<turn n="109" speaker="A">
<decl n="205" sp-act="predict" polarity="positive" mode="intent-predict-benefit-decl">
i'm sure we'll find that for you <overlap type="end" /> <punc type="stop" />
</decl>
</turn>
<turn n="110" speaker="B">
<frag n="206" sp-act="thank" polarity="positive" mode="thank-decl">
thank you very much <punc type="stop" />
</frag>
<address n="207" sp-act="refer" polarity="positive" mode="decl">
<pause length="15s" /> Wella <punc type="stop" />
</address>
<frag n="208" sp-act="greet" mode="greet-opening">
hi <punc type="stop" />
</frag>
<decl n="209" sp-act="identifySelf" polarity="positive" mode="intro-decl">
it's me <punc type="stop" />
</decl>
<decl n="210" sp-act="state-abandon" polarity="positive" mode="interruption-exists-decl">
<pause /> i've got a ti... <punc type="stop" />
</decl>
</turn>
<turn n="111" speaker="A">
<dm n="211" sp-act="init">
so <punc type="level" />
</dm>
<decl n="212" sp-act="reqInfo" polarity="positive" mode="query">
you want me to go ahead <punc type="query" />
</decl>
</turn>
<turn n="112" speaker="B">
<yes n="213" sp-act="answer-acknowledge">
yes <punc type="stop" />
</yes>
</turn>
<turn n="113" speaker="A">
<dm n="214" sp-act="init">
so <punc type="level" />
```

```
</dm>
<decl n="215" sp-act="stateCondition" polarity="negative" mode="condition-frag">
if your sister calls you back and says that she doesn't want it <punc type="level" />
</decl>
<decl n="216" sp-act="state" polarity="positive" topic="time-enum-fare-refund-journey-day-cancel" mode="condition-decl">
you get a full refund less 5 pounds <backchannel content="ok" /> if you cancel by 2 o'clock the day before the outward <pause /> journey <punc type="stop" />
</decl>
</turn>
<turn n="114" speaker="B">
<yes n="217" sp-act="acknowledge">
yeah <punc type="stop" />
</yes>
</turn>
<turn n="115" speaker="A">
<decl n="218" sp-act="stateCondition" polarity="positive" topic="time-enum" mode="condition-frag">
if it's after the 2 o'clock deadline <punc type="level" />
</decl>
<frag n="219" sp-act="refer" polarity="positive" topic="time-departure" mode="constrain-frag">
but before departure time of the train <punc type="level" />
</frag>
<decl n="220" sp-act="state" polarity="positive" topic="enum-fare-refund" mode="decl">
it's 50 percent refund less 5 pounds <punc type="stop" />
</decl>
</turn>
<turn n="116" speaker="B">
<dm n="221" sp-act="acknowledge" mode="tag">
ok <punc type="stop" />
</dm>
</turn>
<turn n="117" speaker="A">
<decl n="222" sp-act="stateCondition" polarity="positive" topic="journey" mode="constrain-condition-frag">
and if you needed to change any of the times of travel <backchannel content="yeah" /> <punc type="level" />
</decl>
<decl n="223" sp-act="stateConstraint" polarity="positive" topic="time-enum-day" mode="constrain-decl">
you must do so by 2 o'clock the day before again <backchannel content="ok" /> <punc type="stop" />
```

```
</decl>
<frag n="224" sp-act="refer" polarity="positive" topic="fare-availability-enum" mode="constrain-decl">
subject to availability and 5 pounds charge <punc type="stop" />
</frag>
</turn>
<turn n="118" speaker="B">
<dm n="225" sp-act="approve">
great <punc type="stop" />
</dm>
</turn>
<turn n="119" speaker="A">
<dm n="226" sp-act="init">
now <punct type="level" />
</dm>
<frag n="227" sp-act="state" polarity="positive" topic="month-date-day" mode="frag">
going out on Saturday the 17th <backchannel content="yeah" /> of October <punc type="level" />
</frag>
<frag n="228" sp-act="refer" polarity="positive" topic="enum" mode="frag">
the 10:50 train <punc type="level" />
</frag>
<decl n="229" sp-act="expressPossibility" polarity="positive" mode="poss1-frag">
i can get a window seat <punc type="level" />
</decl>
<decl n="230" sp-act="state" polarity="positive" topic="journey-time" mode="constrain-decl">
but it's backward travel at a table <punc type="stop" />
</decl>
<decl n="231" sp-act="state" polarity="positive" topic="seat" mode="constrain-decl">
<pause /> but it is non-smoking <punc type="stop" />
</decl>
</turn>
<turn n="120" speaker="B">
<decl n="232" sp-act="approve" polarity="positive" mode="reassurance-decl">
that's fine <punc type="stop" />
</decl>
</turn>
<turn n="121" speaker="A">
<frag n="233" sp-act="refer" polarity="positive" topic="fare" mode="frag">
on the return on the 18th <punc type="level" />
```

```
</frag>
<decl n="234" sp-act="state" polarity="positive" topic="seat-from-enum" mode="decl">
the train from Wilmslow from 15:46 is forward facing at the window non-smoking <punc type="stop" />
</decl>
</turn>
<turn n="122" speaker="B">
<dm n="235" sp-act="approve">
great <punc type="stop" />
</dm>
</turn>
<turn n="123" speaker="A">
<dm n="236" sp-act="init">
now <punct type="level" />
</dm>
<decl n="237" sp-act="state" polarity="positive" topic="fare-enum" mode="decl">
the total cost is 30 pounds Return <punc type="stop" />
</decl>
</turn>
<turn n="124" speaker="B">
<dm n="238" sp-act="acknowledge" mode="tag">
ok <punc type="stop" />
</dm>
</turn>
<turn n="125" speaker="A">
<decl n="239" sp-act="reqConfirm" polarity="positive" topic="creditcard" mode="report-query">
and you said it's yourself as the credit card holder <punc type="query" />
</decl>
</turn>
<turn n="126" speaker="B">
<yes n="240" sp-act="confirm-acknowledge">
yeah <punc type="stop" />
</yes>
</turn>
<turn n="127" speaker="A">
<frag n="241" sp-act="direct-refer" polarity="positive" topic="name" mode="request-query">
and your surname please <punc type="query" />
</frag>
</turn>
<turn n="128" speaker="B">
<frag n="242" sp-act="refer" polarity="positive" mode="decl">
<anonym type="name" /> <punc type="stop" />
```

```
</frag>
<frag n="243" sp-act="refer" polarity="positive" mode="decl">
<anonym type="letter" /> <anonym type="letter" /> <anonym type="letter" /> <anonym type="letter" /> <anonym type="letter" /> <anonym type="letter" /> <anonym type="letter" /> <punc type="stop" />
</frag>
<dm n="244" sp-act="apologise" mode="regret">
sorry <punc type="stop" />
</dm>
<imp n="245" sp-act="hold" polarity="positive" mode="hold-decl">
just hold the line <punc type="stop" />
</imp>
<decl n="246" sp-act="expressOpinion" polarity="positive" mode="opinion-decl">
i think this is her <overlap type="start" /> <punc type="stop" />
</decl>
<imp n="247" sp-act="hold" polarity="positive" mode="hold-decl">
hold on <punc type="stop" />
</imp>
</turn>
<turn n="129" speaker="A">
<dm n="248" sp-act="acknowledge" mode="tag">
ok <overlap type="end" /> <punc type="stop" />
</dm>
</turn>
<turn n="130" speaker="B">
<frag n="249" sp-act="state" polarity="positive" mode="decl">
<pause length="2s" /> gone <punc type="stop" />
</frag>
<dm n="250" sp-act="exclaim">
ha
</dm>
<dm n="251" sp-act="acknowledge">
ok <punc type="stop" />
</dm>
<dm n="252" sp-act="apologise" mode="regret">
sorry <punc type="stop" />
</dm>
<imp n="253" sp-act="suggest" polarity="positive" mode="decl">
let's go for it <punc type="stop" />
</imp>
</turn>
<turn n="131" speaker="A">
<dm n="254" sp-act="confirm-acknowledge" mode="tag">
ok then <punc type="stop" />
</dm>
```

```
<dm n="255" sp-act="init">
now <punct type="level" />
</dm>
<frag n="256" sp-act="reqInfo" polarity="positive" topic="name" mode="query">
your initial <punc type="query" />
</frag>
</turn>
<turn n="132" speaker="B">
<frag n="257" sp-act="answer-refer" polarity="positive" mode="decl">
<anonym type="letter" /> <punc type="stop" />
</frag>
</turn>
<turn n="133" speaker="A">
<frag n="258" sp-act="reqInfo" polarity="positive" topic="name" mode="query">
and your title <punc type="query" />
</frag>
</turn>
<turn n="134" speaker="B">
<frag n="259" sp-act="answer-refer" polarity="positive" mode="decl">
Miss <punc type="stop" />
</frag>
</turn>
<turn n="135" speaker="A">
<dm n="260" sp-act="init">
<pause /> now <punct type="level" />
</dm>
<q-yn n="261" sp-act="direct-reqInfo" polarity="positive" topic="telephone-number" mode="request-query">
could you give me your contact telephone number please <punc type="query" />
</q-yn>
</turn>
<turn n="136" speaker="B">
<yes n="262" sp-act="acknowledge">
yes <punc type="stop" />
</yes>
<frag n="263" sp-act="refer" polarity="positive" topic="enum" mode="frag">
0 1 9 5 <punc type="level" />
</frag>
</turn>
<turn n="137" speaker="A">
<dm n="264" sp-act="acknowledge">
```

```
aha <punc type="stop" />
</dm>
</turn>
<turn n="138" speaker="B">
<frag n="265" sp-act="refer" polarity="positive" topic="enum" mode="frag">
9 6 8 <punc type="level" />
</frag>
</turn>
<turn n="139" speaker="A">
<dm n="266" sp-act="acknowledge">
aha <punc type="stop" />
</dm>
</turn>
<turn n="140" speaker="B">
<frag n="267" sp-act="refer" polarity="positive" topic="enum" mode="decl">
5 1 6 5 <punc type="stop" />
</frag>
</turn>
<turn n="141" speaker="A">
<frag n="268" sp-act="reqInfo" polarity="positive" topic="creditcard-address" mode="query">
and your postcode to where your credit card is registered to <punc type="query" />
</frag>
</turn>
<turn n="142" speaker="B">
<dm n="269" sp-act="hesitate">
erm
</dm>
<frag n="270" sp-act="answer-refer" polarity="positive" topic="enum" mode="frag">
<anonym type="letter" /> <anonym type="letter" /> 1 <punc type="level" />
</frag>
</turn>
<turn n="143" speaker="A">
<dm n="271" sp-act="acknowledge">
<pause /> aha <punc type="stop" />
</dm>
</turn>
<turn n="144" speaker="B">
<frag n="272" sp-act="refer" polarity="positive" topic="enum" mode="decl">
8 <anonym type="letter" /> for <anonym type="alpha" /> <anonym type="letter" /> for <anonym type="alpha" /> <punc type="stop"
```

```
/>
</frag>
</turn>
<turn n="145" speaker="A">
<frag n="273" sp-act="direct-refer" polarity="positive" topic="address" mode="request-query">
and that address for me please <punc type="query" />
</frag>
</turn>
<turn n="146" speaker="B">
<frag n="274" sp-act="refer" polarity="positive" mode="frag">
Flat 6 <punc type="level" />
</frag>
<frag n="275" sp-act="refer" polarity="positive" topic="enum" mode="frag">
<pause /> 168 New <anonym type="street" /> Street <punc type="level" />
</frag>
<frag n="276" sp-act="refer" polarity="positive" topic="location-enum" mode="decl">
London <anonym type="letter" /> <anonym type="letter" /> 1 8 <anonym type="letter" /> <anonym type="letter" /> <punc type="stop" />
</frag>
</turn>
<turn n="147" speaker="A">
<dm n="277" sp-act="init">
now <punct type="level" />
</dm>
<decl n="278" sp-act="predict" polarity="positive" topic="journey" mode="predict-decl">
<pause /> that's where the ticket will be posted out to <punc type="stop" />
</decl>
</turn>
<turn n="148" speaker="B">
<dm n="279" sp-act="approve">
great <punc type="stop" />
</dm>
</turn>
<turn n="149" speaker="A">
<q-wh n="280" sp-act="reqInfo" polarity="positive" topic="creditcard" mode="open-query">
and what type of credit is that you hold <punc type="query" />
</q-wh>
</turn>
<turn n="150" speaker="B">
```

```
<q-yn n="281" sp-act="reqInfo" polarity="positive" mode="alternative-closed-query">
do you take Diner's or American Express or anything like that <punc type="query" />
</q-yn>
</turn>
<turn n="151" speaker="A">
<yes n="282" sp-act="answer-acknowledge">
yes <punc type="stop" />
</yes>
<decl n="283" sp-act="elab-state" polarity="positive" mode="decl">
we do <punc type="stop" />
</decl>
</turn>
<turn n="152" speaker="B">
<q-yn n="284" sp-act="reqOpt-direct" polarity="positive" mode="request-closed-query">
can i put it on my Diner's then please <punc type="query" />
</q-yn>
</turn>
<turn n="153" speaker="A">
<yes n="285" sp-act="stateOpt-acknowledge">
yeah <punc type="stop" />
</yes>
<decl n="286" sp-act="elab-stateCondition" polarity="positive" topic="number" mode="condition-decl">
if you give me the number <punc type="stop" />
</decl>
</turn>
<turn n="154" speaker="B">
<yes n="287" sp-act="acknowledge">
yep <punc type="stop" />
</yes>
<decl n="288" sp-act="state" polarity="positive" topic="enum" mode="frag">
it is 8 fo... 8 1 6 1 <punc type="level" />
</decl>
</turn>
<turn n="155" speaker="A">
<frag n="289" sp-act="echo-refer" polarity="positive" topic="enum" mode="frag">
8 1 6 1 <punc type="level" />
</frag>
</turn>
<turn n="156" speaker="B">
<frag n="290" sp-act="refer" polarity="positive" topic="enum" mode="frag">
```

```
2 2 6 1 <punc type="level" />
</frag>
</turn>
<turn n="157" speaker="A">
<frag n="291" sp-act="echo-refer" polarity="positive" topic="enum"
    mode="frag">
2 2 6 1 <punc type="level" />
</frag>
</turn>
<turn n="158" speaker="B">
<frag n="292" sp-act="refer" polarity="positive" topic="enum" mode
    ="frag">
1 8 <punc type="level" />
</frag>
</turn>
<turn n="159" speaker="A">
<frag n="293" sp-act="echo-refer" polarity="positive" topic="enum"
    mode="frag">
1 8 <punc type="level" />
</frag>
</turn>
<turn n="160" speaker="B">
<frag n="294" sp-act="refer" polarity="positive" topic="enum" mode
    ="decl">
2 2 6 6 <punc type="stop" />
</frag>
</turn>
<turn n="161" speaker="A">
<frag n="295" sp-act="echo-refer" polarity="positive" topic="enum"
    mode="decl">
2 2 6 6 <punc type="stop" />
</frag>
</turn>
<turn n="162" speaker="B">
<frag n="296" sp-act="refer" polarity="positive" topic="enum-
    creditcard" mode="decl">
expiry 1 <pause /> 99 <punc type="stop" />
</frag>
</turn>
<turn n="163" speaker="A">
<dm n="297" sp-act="init">
<pause /> now <punct type="level" />
</dm>
<decl n="298" sp-act="state" polarity="positive" topic="booking-
    number-journey" mode="decl">
this is your booking reference number that the ticket has been
    booked under <punc type="stop" />
```

```
</decl>
</turn>
<turn n="164" speaker="B">
<yes n="299" sp-act="acknowledge">
yeah <punc type="stop" />
</yes>
</turn>
<turn n="165" speaker="A">
<q-yn n="300" sp-act="reqInfo" polarity="positive" mode="closed-query">
do you have a pen handy <punc type="query" />
</q-yn>
</turn>
<turn n="166" speaker="B">
<yes n="301" sp-act="answer-acknowledge">
yeah <punc type="stop" />
</yes>
</turn>
<turn n="167" speaker="A">
<decl n="302" sp-act="state" polarity="positive" mode="frag">
it's <anonym type="letter" /> for <anonym type="alpha" /> <anonym type="letter" /> for <anonym type="alpha" /> <punc type="level" />
</decl>
</turn>
<turn n="168" speaker="B">
<yes n="303" sp-act="acknowledge">
yeah <punc type="stop" />
</yes>
</turn>
<turn n="169" speaker="A">
<frag n="304" sp-act="refer" polarity="positive" mode="decl">
<anonym type="letter" /> for <anonym type="alpha" /> <anonym type="letter" /> for <anonym type="alpha" /> <punc type="stop" />
</frag>
</turn>
<turn n="170" speaker="B">
<yes n="305" sp-act="acknowledge">
yeah <punc type="stop" />
</yes>
</turn>
<turn n="171" speaker="A">
<frag n="306" sp-act="refer" polarity="positive" topic="enum" mode="decl">
9 1 5 <punc type="stop" />
</frag>
</turn>
```

```
<turn n="172" speaker="B">
<dm n="307" sp-act="approve">
<overlap type="start" /> great <punc type="stop" />
</dm>
</turn>
<turn n="173" speaker="A">
<frag n="308" sp-act="refer" polarity="positive" mode="partial-decl">
<anonym type="letter" /> <overlap type="end" /> for <anonym type="alpha" /> <punc type="stop" />
</frag>
</turn>
<turn n="174" speaker="B">
<dm n="309" sp-act="approve">
great <punc type="stop" />
</dm>
</turn>
<turn n="175" speaker="A">
<decl n="310" sp-act="predict" polarity="positive" topic="journey-time" mode="predict-decl">
and the ticket will be posted out first class to you this afternoon <punc type="stop" />
</decl>
<frag n="311" sp-act="reqConfirm" polarity="positive" mode="tag-query">
ok <punc type="query" />
</frag>
<address n="312" sp-act="refer">
Miss <overlap type="start" /> <anonym type="name" /> <punc type="stop" />
</address>
</turn>
<turn n="176" speaker="B">
<frag n="313" sp-act="confirm-thank" polarity="positive" mode="thank-decl">
thank you <overlap type="end" /> <punc type="stop" />
</frag>
<decl n="314" sp-act="elab-approve" polarity="positive" mode="decl">
that's great <punc type="stop" />
</decl>
<dm n="315" sp-act="init">
so <punc type="level" />
</dm>
<q-yn n="316" sp-act="reqOpt" polarity="positive" topic="number-time" mode="closed-query">
can we just go over it one more time <punc type="query" />
```

```
</q-yn>
</turn>
<turn n="177" speaker="A">
<yes n="317" sp-act="stateOpt-acknowledge">
yes <punc type="stop" />
</yes>
</turn>
<turn n="178" speaker="B">
<decl n="318" sp-act="state" polarity="positive" mode="decl">
that's the 10:50 <punc type="stop" />
</decl>
<frag n="319" sp-act="state" polarity="positive" topic="to-number-day" mode="decl">
5 O <backchannel content="yep" /> on Saturday the 17th to Macclesfield <punc type="stop" />
</frag>
<frag n="320" sp-act="reqConfirm" polarity="positive" topic="time" mode="query">
getting on at 13:03 <punc type="query" />
</frag>
</turn>
<turn n="179" speaker="A">
<decl n="321" sp-act="confirm-state" polarity="positive" mode="decl">
that's great <punc type="stop" />
</decl>
</turn>
<turn n="180" speaker="B">
<frag n="322" sp-act="refer" polarity="positive" topic="day" mode="partial-frag">
and then on Sunday the 18th <punc type="level" />
</frag>
<dm n="323" sp-act="hesitate">
erm
</dm>
<frag n="324" sp-act="refer" polarity="positive" mode="frag">
15:55 <punc type="level" />
</frag>
<imp n="325" sp-act="direct" polarity="positive" topic="to-from" mode="frag">
take a coach to from Macclesfield to Wilmslow <punc type="level" />
</imp>
<imp n="326" sp-act="direct" polarity="positive" topic="time" mode="decl">
get on the train at 16:25 <punc type="stop" />
</imp>
```

```
<decl n=”327” sp-act=”state” polarity=”positive” topic=”location-time” mode=”decl”>
and it gets into London at 19:13 <punc type=”stop” />
</decl>
<dm n=”328” sp-act=”init”>
<pause /> now <punct type=”level” />
</dm>
<q-wh n=”329” sp-act=”suggest-reqInfo” polarity=”negative” mode=”alternative-condition-closed-query”>
what if the coach Ma... is is not running properly or something <punc type=”query” />
</q-wh>
<decl n=”330” sp-act=”reqInfo” polarity=”positive” mode=”query”>
what happens <punc type=”query” />
</decl>
</turn>
<turn n=”181” speaker=”A”>
<dm n=”331” sp-act=”init”>
well <punct type=”level” />
</dm>
<decl n=”332” sp-act=”answer-refer” polarity=”positive” topic=”time-from” mode=”frag”>
the train from Wilmslow at 16:46 <punc type=”level” />
</decl>
<decl n=”333” sp-act=”elab-state” polarity=”positive” topic=”time” mode=”decl”>
you get in at 16:25 <punc type=”stop” />
</decl>
<decl n=”334” sp-act=”stateCondition” polarity=”positive” topic=”time-enum” mode=”benefit-exists-decl”>
so there is 21 minutes for you <punc type=”stop” />
</decl>
<decl n=”335” sp-act=”state” polarity=”positive” mode=”decl”>
t... the buses that we've put on is specifically for <pause /> connecting with the trains <punc type=”stop” />
</decl>
</turn>
<turn n=”182” speaker=”B”>
<yes n=”336” sp-act=”acknowledge”>
yeah <punc type=”stop” />
</yes>
</turn>
<turn n=”183” speaker=”A”>
<decl n=”337” sp-act=”state” polarity=”positive” topic=”problem-location” mode=”condition-frag”>
so if there was any problems with connections <punc type=”level” />
```

```
</decl>
<decl n="338" sp-act="stateCondition" polarity="positive" mode="reason-condition-frag">
then if it was due to the coach <punc type="level" />
</decl>
<decl n="339" sp-act="state" polarity="positive" topic="availability" mode="decl">
then it would <unclear length="6 syllables" /> to try and get you on the next available service <punc type="stop" />
</decl>
</turn>
<turn n="184" speaker="B">
<dm n="340" sp-act="acknowledge" mode="tag">
ok <punc type="stop" />
</dm>
</turn>
<turn n="185" speaker="A">
<decl n="341" sp-act="stateReason" polarity="positive" mode="reason-decl">
cos the buses are from each train station <event type="tape cuts off" /> <punc type="stop" />
</decl>
</turn>
</dialogue>
```

9.2 Experiment 1: Additional Results

Dataset	Scalar	Imputer	AUC-PR on Validation Set
DA Dataset SAASA features minus DA	MinMaxScalar	OneHotEncoder	0.55643
DA Dataset SAASA features minus DA	MinMaxScalar	TargetEncoder	0.52083
DA Dataset SAASA features minus DA	StandardScalar	OneHotEncoder	**0.55644**
DA Dataset SAASA features minus DA	StandardScalar	TargetEncoder	0.52083
DA or Turn level Dataset SAASA features Plus BoW DA	MinMaxScalar	OneHotEncoder	**0.56015**
DA or Turn level Dataset SAASA features Plus BoW DA	MinMaxScalar	TargetEncoder	0.52075
DA or Turn level Dataset SAASA features Plus BoW DA	StandardScalar	OneHotEncoder	0.55998
DA or Turn level Dataset SAASA features Plus BoW DA	StandardScalar	TargetEncoder	0.51973

Table 22: Validation Set Results for Best Imputers, Scalars, and Encoders for Experiment 1 (Pre-Optimization)

Experiment Dataset	Number of Trees	Learning Rate	Maximum Depth
Baseline 1	200	0.01	12
Baseline 2	200	0.01	12
Topic Trainer 1	200	0.01	12
Topic Trainer 2	200	0.01	12
Topic Trainer 3	200	0.01	12

Table 23: Best Hyperparameters when Optimizing for F1 score in Grid Search Cross-Validation for 1 XGBoost Classifier

Experiment Dataset	Best Threshold Precision	Best Threshold Recall	Best Threshold F1	AUC-PR
Baseline 1	0.382	0.896	0.535	0.480
Baseline 2	0.377	0.881	0.528	0.451
Topic Trainer 1	0.435	0.750	0.550	0.535
Topic Trainer 2	0.403	0.860	0.548	0.564
Topic Trainer 3	0.421	0.782	0.547	0.553

Table 24: Module 5 Topic Trainer Experiment 1 Results at Best Threshold Post-Optimization on Validation Dataset

Experiment Dataset	Best Threshold	Best Threshold F1
Baseline 1	0.216	0.535
Baseline 2	0.188	0.528
Topic Trainer 1	0.157	0.535
Topic Trainer 2	0.170	0.548
Topic Trainer 3	0.167	0.547

Table 25: Module 5 Topic Trainer Experiment 1 Best Threshold and F1 Score based on Validation Dataset

9.3 Experiment 2: Additional Results

Dataset	Scalar	AUC-PR on Validation Set
Turn Dataset SAASA features minus Turn level DA(s)	MinMaxScalar	**0.35516**
Turn Dataset SAASA features minus Turn level DA(s)	StandardScalar	0.0.35151
Turn level Dataset SAASA features Plus BoW Turn level DA(s)	MinMaxScalar	0.37019
Turn level Dataset SAASA features Plus BoW Turn level DA(s)	StandardScalar	**0.37621**
Turn level Dataset SAASA features Plus TF-IDF Turn level DA(s)	MinMaxScalar	0.35358
Turn level Dataset SAASA features Plus TF-IDF Turn level DA(s)	StandardScalar	**0.37913**

Table 26: Validation Set Results for Best Imputers, Scalars, and Encoders for Experiment 2 (Pre-Optimization)

Experiment Dataset	Number of Trees	Learning Rate	Maximum Depth
Baseline 1	200	0.01	12
Baseline 2	200	0.01	12
Topic Trainer 1	200	0.01	12
Topic Trainer 2	200	0.01	12
Topic Trainer 3	200	0.01	12

Table 27: Best Hyperparameters when Optimizing for F1 score in Grid Search Cross-Validation for 2 XGBoost Classifier

Experiment Dataset	Best Threshold Precision	Best Threshold Recall	Best Threshold F1	AUC-PR
Baseline 1	0.329	0.560	0.415	0.323
Baseline 2	0.362	0.513	0.424	0.322
Topic Trainer 1	0.322	0.670	0.435	0.334
Topic Trainer 2	0.321	0.707	0.442	0.365
Topic Trainer 3	0.324	0.717	0.446	0.376

Table 28: Module 5 Topic Trainer Experiment 2 Results at Best Threshold Post-Optimization on Validation Dataset

Experiment Dataset	Best Threshold	Best Threshold F1
Baseline 1	0.248	0.415
Baseline 2	0.303	0.424
Topic Trainer 1	0.091	0.435
Topic Trainer 2	0.081	0.442
Topic Trainer 3	0.102	0.446

Table 29: Module 5 Topic Trainer Experiment 2 Best Threshold and F1 Score based on Validation Dataset

9.3.1 Experiment 2 Topic Trainer 4 Results: Classifier Trained on TF-IDF vector representation of all Turn Level Dataset Features

Below we see the results of a Topic Trainer 4 in which all Turn level dataset features (listed in Table 8) are represented using TF-IDF vectors. Like the other Topic Trainers in Experiment 2, the hyperparameters that yielded the best results are number of tree = 200, learning rate = 0.01 and maximum depth = 12. Unilke results in Tables 19 and 20, here we see pre-optimization results yield a precision of 0.522 whereas for other Topic Trainers in Experiment 2 these values remained between 0.3 and 0.37. Post-optimization precision value at the best threshold is low at 0.29 when compared to other Topic Trainer precision scores remained above 0.3. To compensate here, recall is the only Topic Trainer in Experiment 2 that achieve over 0.9 (0.932) resulting in an F1 score of 0.442 and the second highest AUC-PR (0.419 see Figure 25; second only to the Topic Trainer 2 AUC-PR score of 0.420).

Looking at Topic Trainer 4 learning curves in Figure 26, we see training score curves decline from a high score for both F1 and precision (similar to what was see in Figures 22, 23 and 24 above. As previously mentioned, this is a typical shape of more complex datasets. The training curves still remain relatively high across when compared to the Baseline learning curves. The cross-validation curves move upwards suggesting the addition of more training data would most likely improve the performance and improve the generalization of Topic Trainer 4. The emphasized variability on the cross-validation score also indicates variance error and not bias error.

Given above results, best performing Topic Trainer in Experiment 2 would still be Topic Trainer 1. While Topic Trainer 4 demonstrates on of the highest AUC-PR post-optimization scores and results in promising learning curves the precision is significantly low compared to the other classifiers, including the baseline classifiers. As previously mentioned, in this context we optimize for F1 in order to have a balance in precision and recall however prefer a higher precision over recall as we would prefer the risk of more false negatives than more false positives (i.e. sudden changes in topics where not actually appropriate).

Topic Trainer 4	Precision	Recall	F1	AUC-PR
Pre-Optimization on Validation Dataset	0.522	0.1888	0.277	0.386
Best Threshold Post-Optimization Test Dataset	0.290	0.932	0.442	0.419

Table 30: Module 5 Topic Trainer 4 Experiment 2 Results for Classifier Trained on TF-IDF vector representation on all Turn Level Feature sequences

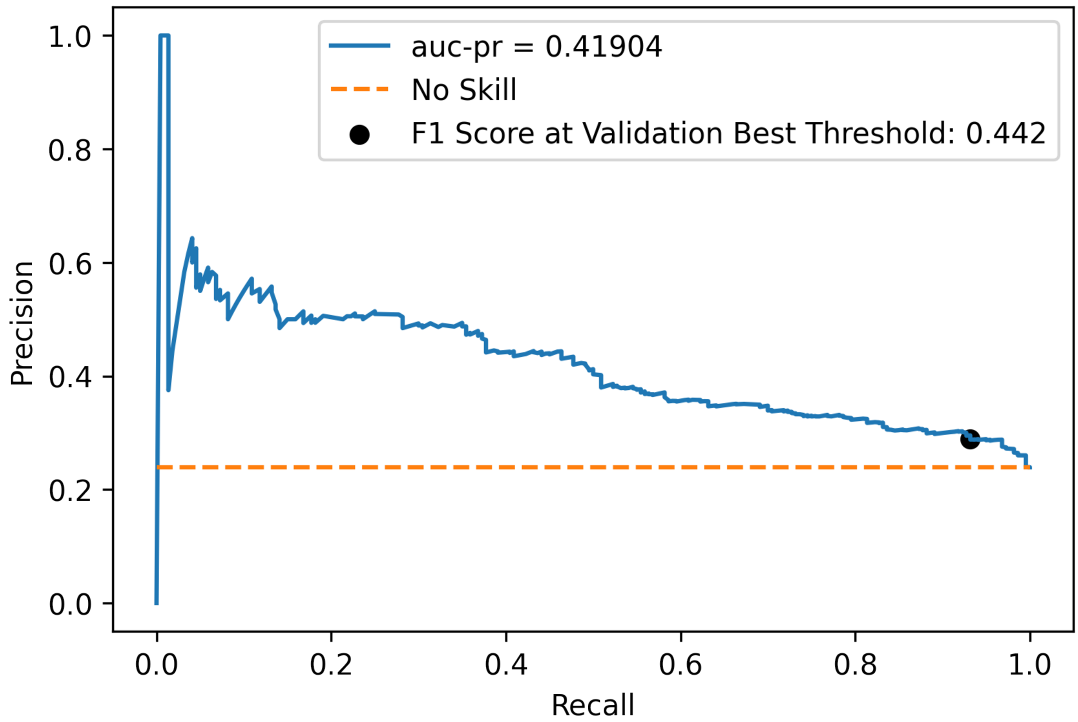
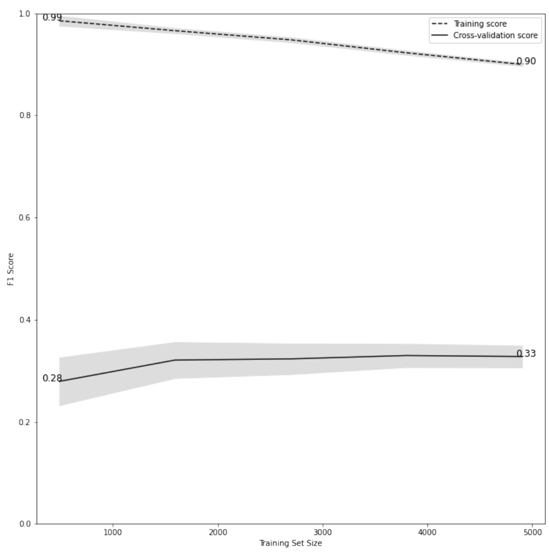
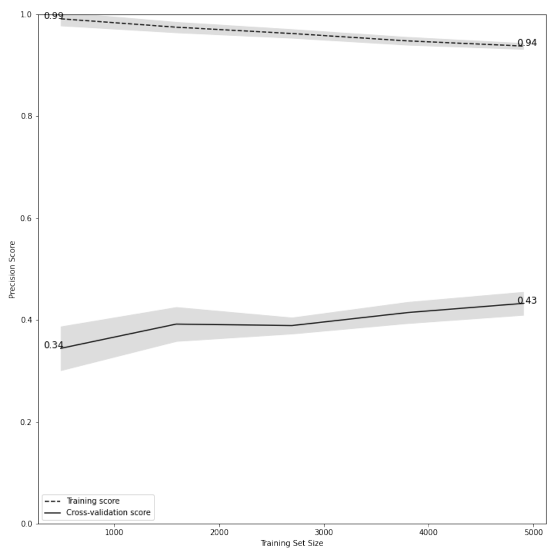


Figure 25: Experiment 2 Topic Trainer 4 Classifier Trained on TF-IDF vector representation on all Turn Level Feature sequences, Best Threshold for Optimal F1 score and AUC-PR for on Test Set



(a) Experiment 2 TF-IDF for All Features Classifier F1 Learning Curve



(b) Experiment 2 TF-IDF for All Features ClassifierPrecision Learning Curve

Figure 26: Learning Curves of Experiment 2 Topic Trainer 4: Classifier Trained on TF-IDF vector representation on all Turn Level Feature sequences

10 Bibliography

References

[1] The society for the study of artificial intelligence and simulation of behaviour events.

[2] E. Adams. *Fundamentals of Game Design.* New Riders Publishing, 2013.

[3] O. S. Ajose-Ismail B.M., Abimbola O.V. Performance analysis of different word embedding models for text classification. *International Journal of Scientific Research and Engineering Development*, 3(6):1016–1020, Nov-Dec 2020.

[4] T. Anderson, M. Blank, B. Daniels, and D. Lebling. *Zork*, 1980.

[5] S. Aydin and H. Jurgensen. Dialogues modelled by cooperating grammars. *Applications of Mathematics in Engineering and Economics'27*, pages 631 – 637, 2002.

[6] S. Aydin and H. Jurgensen. Parallel versus sequential grammar systems for modelling dialogues. pages 596 – 600, 2003.

[7] S. Aydin, H. Jurgensen, and L. Robbins. Dialogues as co-operating grammars. *Journal of Automata, Languages and Combinatories 6*, 4:395–410, 2001.

[8] P. Branco, L. Torgo, and R. P. Ribeiro. A survey of predictive modelling under imbalanced distributions, May 2015. Laboratory of Aritificial Intelligence and Decision Support (LIAAD) - INESC TEC and University of Porto.

[9] M. Buckland. *Programming Game AI by Example.* Wordware Publishing, Inc, 2005.

[10] A. Burkov. *The Hundred-Page Machine Learning Book.* Andriy Burkov, author@themlbook.com, 2019.

[11] D. Cameron. *Working with Spoken Discourse.* SAGE Publications, 2001.

[12] C. Chakrabarti. *Artificial Conversations for Chatter Bots Using Knowledge Representation, Learning, and Pragmatics.* PhD thesis, University of New Mexico, Albuquerque, New Mexico, May 2014.

[13] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge*

Discovery and Data Mining, pages 785–794, San Francisco, CA, USA, 2016. Association of Computing Machinery, ACM.

[14] F. Chollet. *Deep Learning with Python.* Manning Publications Co., 20 Baldwin Road, PO Box 81, Shelter Island, NY 11964, 2018.

[15] P. Cohen. *Survey of the State of the Art in Human Language Technology*, chapter Discourse and Dialogue, Section 3: Dialogue Modelling, pages 204–210. Cambridge University Press and Giardini, 1997.

[16] J. Dassow, G. Paun, and G. Rozenberg. Grammar systems. *Handbook of Formal Languages*, 2:155 – 213, 1997.

[17] R. Davies, N. Dewell, and C. Harvey. A framework for interactive, autonomous and semantic dialogue generation in games. In *Proceedings of the 1st Games Technology Summit*, 2021.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[19] A. Ezen-Can. A comparison of lstm and bert for small corpus, September 2020.

[20] M. Fisel. Machine learning techniques in dialogue act recognition. *Estonian Papers Applied Linguistics*, 2007.

[21] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera. A review on ensembles for the class imbalance problem: Bagging- , boosting, and hybrid-based approaches. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART C: APPLICATIONS AND REVIEWS*, 42(4):463 – 484, July 2012.

[22] I. Gunasekara and I. Nejadgholi. A review of standard text classification practices for multi-label toxicity identification of online content. pages 21–25, October 2018.

[23] C. Haustein. Implementierung von dialogen unter einsatz von mcd-grammatiksystemen. Master's thesis, Institut für Informatik der Universität Potsdam, Potsdam, November 2004.

[24] M. Heidari and J. H. Jones. Using bert to extract topic-independent sentiment features for social media bot detection. IEEE Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), October 2020.

[25] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning with Applications in R.* Springer, 2013.

[26] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning with Applications in R.* Number 8. Springer Science and Business Media, New York, 2017.

[27] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence (IJCAI).* Computer Science Department, Stanford University, 1995.

[28] G. Leech and M. Weisser. The spaadia annotation scheme. February 2014.

[29] X. Li, L. Mou, R. Yan, and M. Zhang. Stalematebreaker: A proactive content-introducing approach to automatic human-computer conversation. In *25th International Joint Conference on Artificial Intelligence*, 2016.

[30] D. Livingstone. Turing's test and believable ai in games. *Computers in Entertainment (CIE)*, 4(1), 2006.

[31] A. I. A. Osman, A. N. Ahmed, M. F. Chow, Y. F. Huang, and A. El-Shafie. *Extreme gradient boosting (Xgboost) model to predict the groundwater levels in Selangor Malaysia.* Elsevier B.V., 2020.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, and V. Michel. Scikit-learn: Machine learning in python. *JMLR 12*, pages 2825 – 2830, 2011.

[33] C. R. Perrault and J. Allen. A plan-based analysis of indirect speech acts. *American Journal of Computational Linguistics*, 6(3):167 – 182, 1980.

[34] Proceedings of the 7th Workshop on Asian Language Resources. *Annotating Dialogue Acts to Construct Dialogue Systems for Consulting*, volume ACL-IJCNLP 2009, pages 32 - 39, Suntec, Singapore, 6-7 August 2009 2009. ACL and AFNLP.

[35] S. Ravichandran, D. Khurana, B. Venkatesh, and N. U. Edakunni. Fairxgboost: Fairness-aware classification in xgboost. In *KDD Workshop on Machine Learning in Finance '20, August 24, 2020.*, New York, NY, USA, 2020. Association of Computing Machinery, ACM.

[36] C. M. Rose. Realistic dialogue engine for video games. Masters thesis, The University of Western Ontario, Department of Computer Science, 2015.

[37] S. Russell and P. Norvig. *Artificial Intelligences: A Modern Approach.* Prentice Hall, 2010.

[38] J. Searle. Speech acts: An essay in the philosophy of language. Master's thesis, Cambrideg, Cambridge University Press, 1969.

[39] P. Suta, X. Lan, B. Wu, P. Mongkolnam, and J. H. Chan. An overview of machine learning in chatbots. *International Journal of Mechanical Engineering and Robotics Research*, 9(4):502 – 510, April 2020.

[40] M. R. T. Saito. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets, 2015.

[41] J. Tanha, Y. Abdi, N. Samadi, N. Razzaghi, and M. Asadpour. *Boosting methods for multi-class imbalanced data classification: an experimental review*, volume 7 of *Journal of Big Data*. SpringerOpen, 2020.

[42] P. Tero. Creating better npcs, March 2013.

[43] A. Tharwat. Classification assessment methods, 2018.

[44] A. M. Turing. Computing machinery and intelligence. *Mind LIX*, pages 433–460, 1950.

[45] F. J. ValverdeAlbacete and C. Pelaez-Moreno. 100% classification accuracy considered harmful: The normalized information transfer factor explains the accuracy paradox, January 2014.

[46] G. Vaszil. *Investigations on Parallel Communicating Grammar Systems*. PhD thesis, Eotvos University, Budapest, Hungary, 2000.

[47] M. Weisser. Tools, ideas & resources for linguistics.

[48] M. Weisser. Spaac speech-act annotation scheme. 2003.

[49] M. Weisser. *Manual for the Dialogue Annotation and Research Tool (DART)*, May 2015.

[50] M. Weisser. Doing corpus pragmatics in dart 2.0 - new and improved ways. Master's thesis, Guangdong University of Foreign Studies, China, 2017.

[51] K. Wheeler. Representing game dialogue as expressions in first-order logic. Master's thesis, University of Western Ontario, 2013.

11 Curriculum Vitae

Name:	Lareina Gail Perez Milambiling
Post-Secondary Education and Degrees:	The University of Western Ontario London, Ontario, Canada 2011 - 2013 M.A. in Linguistics Supervisor: Dr. Ileana Paul — Second Reader: Dr. Robert Stainton The University of Western Ontario London, Ontario, Canada 2007 - 2011 Honors B.A in Linguistics & Sociocultural Anthropology
Honours and Awards:	Mathematics of Information Technology & Complex Systems (MITACS) Internship Grant 2015-2016 Natural Sciences and Engineering Research Council (NSERC) CGS M 2014-2015 Linguistic Society of America Institute Fellowship 2013 Global Opportunities Award 2013 Dean's Honor List 2010 - 2011
Related Diplomas and Certificates:	Continuing Studies at Western London, Ontario, Canada 2013 - 2014 Diploma in Computer Science Linguistics Institute on Universality and Variability Ann Arbor, Michigan, USA 2013 Certificate

Related Work Experience:	Teaching Assistant The University of Western Ontario 2011 - 2016 Business Intelligence & Analytics Intern Big Blue Bubble Inc 2015 - 2016 Google Summer of Code Contributor OpenCog Foundation 2014 Research Assistant The University of Western Ontario 2011 - 2014

Publications:

L. Milambiling, M. Katchabaw, and D. Slogar, "Integrating Social and Textual Analytics into Game Analytics." *Data Analytics Applications in Gaming and Entertainment*, edited by Günter Wallner, 1st Edition, Auerbach Publications, 2019, pp. 141 - 168

K. Cortes, L. Milambiling and I. Paul, "Definiteness without D: The case of ang and ng in Tagalog" in *Canadian Journal of Linguistics 60 (3): 361 - 390*, 2015 November

Non-Refereed Contributions:

K. Cortes, L. Milambiling and I. Paul, "Does Tagalog have determiners?" in *2012 Canadian Linguistic Association Conference Proceedings*, 2012

L. Milambiling, "Null Object Constructions In Tagalog?" in *2011 Canadian Linguistic Association Conference Proceedings*, 2011