Australian National University

Computer Science Honours Thesis

Event Extraction and Temporal Ordering towards Narrative Model Generation

William James

Bachelor of Advanced Computing (Honours)

supervised by

Dr. Hanna Suominen

Dr. Patrik Haslum

October 2017

Abstract

Narrative generation is the process of automatically generating narrative stories (fictional or not) through a narrative model. With the advances in machine learning and natural language processing techniques, narrative generators have been able to create more creative and interesting text in terms of narrative contents and story telling techniques. These systems have applications in a range of domains including education and patient recovery. Narrative generation models are traditionally planning problems and uses a schema to represent the narrative world. This schema contains states and actions with preconditions and postconditions and the sequence of actions that lead to one or more goal states (dependent on the author's goal) is considered the skeleton for the narrative story. Many modern systems then employ heuristics to measure and improve storytelling aspects such as narrative suspense. However, these schemas are usually defined manually, requiring significant user input, or otherwise can only represent simplistic events and generate simple narratives. Here we explore the building of a system that can automatically generate these schemas to be used within a narrative model. We present the initial stages of such a system, pertaining steps of lexical processing, event extraction, and temporal event ordering. Our event extraction is performed by a rule-based system that can obtain informative triples accurately to our task without the need of training data. Determining the temporal ordering of extracted events is then done through multiple multilayer perceptron networks using features obtained through statistical and rule-based techniques. Our temporal ordering achieves accuracies of above 70 per cent for both event-event pairs and time-event pairs on Timebank and Opinion corpera. This pipeline presents the initial steps of a system to be able to automatically create states and actions with preconditions and postconditions which will form the narrative schema. We discuss several limitations of the techniques used and present potential improvements and future work towards the complete narrative generator system.

Keywords: *Information Extraction*, *Natural Language Generation*, *Lexical Semantics*, *Temporal Reasoning*, *Supervised Learning by Classification*, *Neural Networks*

Contents

---

1 Introduction

Natural Language Processing (NLP) has had significant progress in the last decade. Much of this is attributed to advances in machine learning; fields such as part-of-speech tagging (identify verbs, nouns, etc.) (Toutanova, et. al., 2003), named entity recognition (identifying names, organisations, time phrases, etc.) (Finkel, et al., 2005) and sentiment analysis (identifying opinions expressed in text) (Santos & Gatti, 2014) have been able to achieve highly accurate results through statistical or neural models. These improvements have found their way into our everyday life whether it be personalized advertisements based on our social media interests or the latest speech recognition software in our phone. One aspect of NLP that is less prevalent, however, is narrative generation, the concept of using a computational model to automatically generate narratives, fictional or not.

This does not mean narrative generation is any less valuable to society, narrative generation can have unique applications in fields such as education (e.g., aiding tourists through educational narratives (Hecht et al., 2007)), health care (e.g., generating narratives to aid patient recovery (Hall & Powell, 2011)) or entertainment (e.g., generating unique character narratives in a virtual environment (Riedl & Young, 2003)). In its most simple form, narratives need to be cohesive in terms of narrative flow, the state of the world it tells and causal relations between narrative events. Being able to do this in a creative and interesting way is a hard to define task (Gervás, 2009). Traditionally, narrative generation is a planning problem where the world is represented by sets of states and actions (with preconditions and postconditions) and the sequence of actions that leads to this goal from the initial state is the basis of the generated narrative (Meehan, 1977; Lebowitz, 1983; Turner, 1993; Riedl, 2004; Gervás, 2009; Young et al. , 2013). In this sense, the initial state would be considered the start of the narrative and the goal state is the end and the actions describe changes to the world within the narrative. The narrative schema that gives these possible actions and states is, however, usually manually defined by the user or developer and as such cannot generate complex and wide narratives without significant user input (Meehan, 1977; Lebowitz, 1983; Turner, 1993; Riedl, 2004).

The goal of this research is to investigate if these narrative schemas can be automatically constructed. We aim to work towards having a narrative generation model that can grow over time through text mining and can generate increasing complex and scalable narratives without user intervention. This research is not concerned with the actual complex narrative generation phase of the system as this can be considered a separate research project. Instead we are focused on identifying states and actions for text that could lead to this narrative generation goal.

Current research in narrative generation is more focused on improving the creative writing and structure of narratives generated from manually defined schemas (Porteous et al., 2013; Cheong & Young, 2006; Doust & Piwek, 2017) rather than exploring how these schemas are created. Our research attempts to explore this latter idea and employ current

NLP techniques in working towards a narrative model that learns new actions and states through analysing text. Two main NLP fields that can be used to work towards this goal is triple extraction and event ordering.

Triple extraction is the process of identifying entities and their relations in text (Angeli et.al, 2015). For instance, from *"Obama was born in Honolulu, Hawaii"* the relation triple (*"Obama"*, born in, *"Honolulu, Hawaii"*) can be extracted, indicating a relation between entities within the text. A range of techniques is used to effectively extract these triples, often combining semantic reasoning and statistical modeling (Chiticariu et al., 2013). Triple extraction is valuable in the goal of automatically extracting narrative models as it allows us to identify entities and relations in text, becoming the basis for the states and actions.

Event ordering is the concept of determining the ordering of identified events within text. Given two extracted events, event ordering is concerned with identifying the type of ordering between them (before, simultaneous, after, etc.) (Mirza, 2014). Methods usually involve training a statistical model to identify ordering based on manually annotated training data and a wide range of features (Chambers et al., 2007; Mirza, 2014; Derczynski & Gaizauskas, 2010; Mirroshandel & Ghassem-Sami, 2012). Event orderings can be used to identify the preconditions and post conditions of the identified actions extracted from text. For instance, if an action always takes place before another, then it is likely to be a precondition for that event and thus would require that type of ordering within the narrative generation. Using this with the triple extraction can allow us to identify actions and states present in text and determine how they are related, providing the building blocks for an automated narrative model builder.

This honour's thesis in Computer Science discusses these concepts and how we approach them in working towards a narrative generation model. The outline of the thesis is as follows: First, we discuss related work in narrative generation to provide an overview of the domain and the problem we are looking at (Section 2). Then, we discuss our overall approach to the project, providing an overview of the system pipeline and approaches used (Section 3). We then present the lexical processing step of the pipeline and discuss the approaches used and motivation behind them (Section 4). We then discuss the event extraction step of the pipeline, presenting related work in the field and illustrate our rule-based approach along with its evaluation (Section 5). We then discuss the event ordering step of the pipeline, presenting related work in the field and then discussing our neural network approach accompanied by its evaluation (Section 6). Finally, we conclude with a discussion of our findings and potential future directions for this work (Section 7).

2 Related Word: Narrative Generation

Narrative generation, the process of using algorithms to generate digital stories, has been around for a while. Dating back to 1973 with Klein's Novel Writer (Klein et al. 1973) there has been interest in this concept. Klein's Novel Writer generates murder mysteries by taking world descriptions as input, containing characters and their traits, which are then used to determine character roles. A set of rules are then used to generate sequences of events which constructed the story. While this is very limited model, it demonstrates how algorithms can be used to generate cohesive stories through character profiles as inputs.

Lebowitz's Universe (1989) generates scripts for TV soap operas. Much of Universe is focused on using complex data structures to represent the characters of the show, becoming the first storytelling system to devote most its attention to modeling characters. Universe takes story goals as input from the user and uses these to model the actions of the stories, keeping a precedence graph to record how the author goals and plot events relate to one another. This uses a planning model structure consisting of actions with preconditions and postconditions that acted upon the state of the world. The system introduces new plot goals and fragments through generalization existing ones, allowing unique narratives to be

generated, demonstrating a form of learning new narrative sequences. The narratives are, however, limited to that of the soap opera genre.

Pérez y Pérez's Mexica (1999) further develops this idea of learning from past narrative sequences by building a set of narrative schemas from sets of previous stories. The state when actions are used in previous stories depict rules involving their engagement, and these rules are used to indicate whether they can be used in the plot of a new narrative. The system reflects on the plot at different stages, checking coherency, novelty and interest, and then uses these scores compared to previous generated stories to choose new plot points and actions. In this sense, the system is able to learn from past generated stories and use this to mould the structure of new ones.

Riedl's Fabulist (2004) further develops the interest of a character focused approach similar to Lebowitz's Universe. In Fabulist characters have individual motives and goals and these are used to drive the narrative. The *intent-driven partial order causal link* (IPOCL) reasons about these defined character goals to produce causally coherent goals, introducing an element of believability towards the character's actions. This system uses a planning approach to generate the narratives where the user gives various inputs representing the world and the planner then modifies these to meet the required character goals.

A problem with the mentioned narrative generators is that they assume structured input from the user and can not generate narratives outside their domain. Klein's (1973) system is limited to weekend murder mysteries, Universe (Lebowitz, 2989) is limited to TV Soap Operas and Mexica (Pérez y Pérez, 1999) is limited to stories set in early Mexico, and they all require a set structure for the world involving characters, character traits, story goals, etc. In 2009, however, Chambers and Jurafsky set to develop methods of learning these actions and event sequences from text without any user input. This work focused on learning script-like information about the world without pre-defined frames, roles or rules, which could then become the basis for narrative generation. This is done through creating single narrative chains containing events and roles which can then be stringed together to create a cohesive narrative. These chains are identified by finding single words in the input text that indicate a relationship between two actors, where the most observed actors would be considered best fitted to that action. This extraction was accurate with an error rate of 28 per cent (majority of which were due to parsing errors), but a limitation of this approach is the learned actions had simplistic structure and may miss important causal information in the text.

Porteous, Charles and Cavazza (2013) further worked on developing effective narrative models focused on character profiles where the generated story is based on relationships between characters. Their narrative model brakes down relationships into several broad categories (affective, romantic or default) which are based off real relationship ontologies. A planning model is then defined with pre- and post-condition actions and an initial and goal state, all of which rely on these relationship definitions. Character relationships then evolve over the course of the narrative, providing progression and character development. Thus, a wide range of narratives are generated where shared action sequences occurred infrequently among narratives (20 per cent of 800 instances). This approach like many of the previous approaches, however, requires a predefined input and as such has little scalability.

Kiciman and Richardson (2015) took a similar approach to Chambers and Jurafsky (2009) in learning actions, but with less focus on a narrative setting. Their work focuses on using social media to information about common actions such as running a marathon or getting a pet and establish the type of outcomes that could be responsible from them. The system looks at social media feeds (mainly Twitter) and texted mined for events of interest, e.g. *"enrolled in a marathon"*. Once a timeline with the keyword is identified they then search through other events of the same user that has some type of connection to the event of interest, compiling lists of negative or positive experiences and those possibly contributing to the achievement (or failure) of the event goal. The purpose of this is to be able to use this information to aid decision making based on the experience of others. While this approach

is less narrative focused, it demonstrates a form of extracting events and relating them to others and could possibly be adapted to a narrative planning structure.

From this previous work, it is evident that narrative generation has had some interest for a good amount of time. As discussed in the many examples, narrative generation has traditionally been approached as planning model consisting of states with actions and their associated pre- and post-conditions and this has been shown to be an effective strategy. Though producing planning structures usually requires structured user input and those that try otherwise can only produce simplistic structures. Through this thesis, we explore using various NLP techniques to develop complex planning structures which could later be used to develop narratives. The work of Chambers and Jurafsky (2009) demonstrates that these types of structures can be effectively learned and Kiciman and Richardson (2015) show how events can be linked to one another in text. The techniques that are used in both works, and many more discussed later, can be adapted and used to provide insight towards developing the goal of a scalable narrative generation planning model.

3 Approach

The approach developed to address the project problem has three main steps, lexical processing, event extraction and event ordering, as illustrated in Figure 1.

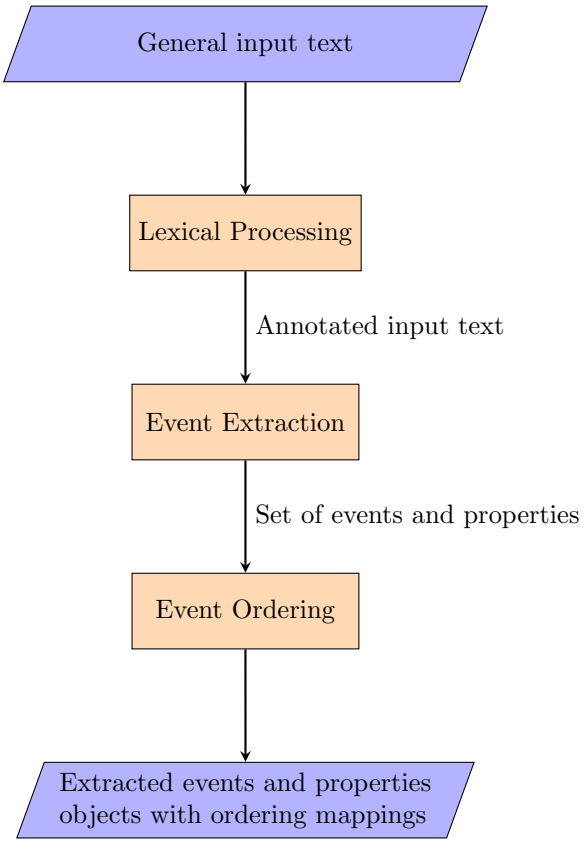

Figure 1: Basic pipeline approach.

The overall process of this pipeline is to take unstructured textual input and provide all extracted events and properties along with their temporal ordering. The goal would be to then use these extractions and ordering to develop narrative planning models. While at this stage it is not a completed model, these processes are crucial in working towards the goal; a possible area for future research.

This current pipeline is sequential and each step relies only on the output on the ones preceding it, meaning that each step can be used and evaluated independently from the ones after it. This, however, also means that performance of the later steps are dependent on the performance of the steps preceding, possibly causing incorrect results. This can be seen in some of the evaluation where inconsistencies in the lexical parsing of the first stage causes inconsistencies in the later parts.

3.1 Lexical Processing

The lexical processing stage of the pipeline does all the main pre-processing aspects that the later processes rely on. This takes unstructured text as input (e.g., news articles) and outputs the same text which has been processed and annotated with additional semantic information. This primarily uses the Stanford CoreNLP *application programming interface* (API), a set of NLP automated annotators developed at Stanford University (Manning et. al., 2014). Annotators used in lexical processing are *lemmatization annotator*, *part-of-speech tagger* (Toutanova et al., 2003), *named entity recognizer* (Finkel et. al., 2005), *coreference resolution system* (Recasens et al., 2013; Lee et al., 2013; Lee et al., 2011; Raghunathan et al., 2010), *dependency parser* (Chen & Manning, 2014) and *constituency parser* (Klein & Manning, 2002). These are used to convert the text into a more structured format for the later parts of the pipeline. The Stanford Temporal Tagger SUTime (Chang & Manning, 2012) is also used to convert time phrases in text to a comparable value. While these libraries have capabilities for multiple different languages, this research is focused on using English text. Additionally, the text inputs are assumed to be grammatically correct as otherwise parsing results will likely be erroneous, likely affecting accuracy in later steps. The input of this step is the original text to be processed and the output is that same text with additional annotated information. For example, a sentence consisting of "*The brown dog plays with his toy"* is passed through this step and is annotated with information similar to Figure 2 below.

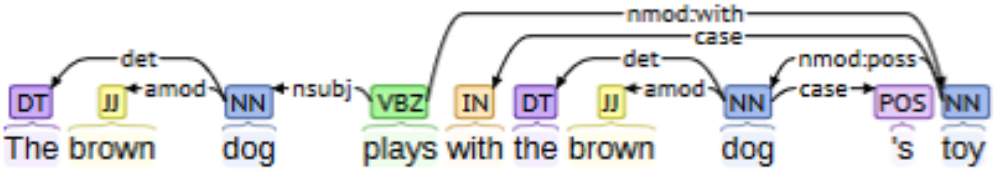

Figure 2: Lexical processing output on example.

The boxes above each word in Figure 2 indicate the part of speech that is associated with the token. For example, *"dog"* has an *NN* part-of-speech indicating a nominal noun and *"the"* has a *DT* part-of-speech indicating a determiner. The arrows between two tokens indicate the typed dependency between them with the specific relation being indicated on the link. For example, *"dog"* is the *nsubj* child of *"plays"* indicating it is a nominal subject of the *"plays"* token. *"Dog"* would be consider the dependant of the relation with *"plays"* being the governor. The specification of these labels come from Universal Dependencies Specification [1] which is a set of consistent treebank annotations for different tasks. In the above example, it is also evident that the step replaced the *"his"* pronoun with the noun it refers to (*"brown dog"*). This is due to coreference resolution which indicates which nouns and pronouns refer to the same object, allowing them to be replaced, making the text more consistent.

[1] http://universaldependencies.org/.

3.2 Event Extraction

The event extraction stage of the pipeline identifies events and properties from the pre-processed text. This stage takes the output of the lexical processing stage as input which represents the original input text in a structured format with additional semantic information annotated. Extractions made are in the form of triples: **{Entity}** ***Relation*** **{Entity}**, indicating relations between entities in text (Mausam et al., 2012). For example, {John} *plays* {Tennis} or {John} *works at* {a coffee shop} are both valid extractions that could be identified from text.

The extraction method used is semantic rule-based where a set of manually defined rules look at the dependency relations between tokens and their structure in the text to identify likely entities and relations based on the lexical processing. As the lexical processing stage provides the original unstructured text in the form of structured dependency trees, rules can then be used to easily identify relations between entities. The extractions identify the core information and framework of the narrative planning model that we are working towards. Additional information identified for the extractions include lexical semantic information such as tense or grammatical aspect which are used as features in the event ordering step of the pipeline. These additional features are similarly identified through manually defined semantic rules.

3.3 Event Ordering

The event ordering step of the pipeline involves identifying the ordering types between event-event pairs and time-event pairs. This takes the output of the two previous stages as input and uses these to identify the ordering between the extracted events. The type of ordering follows the widely accepted TimeML 1.2.1 specification (Saurí et al., 2006) for temporal information and relations between events in text. The ordering is identified through four neural networks: one to detect if ordering exists between event-event pairs, one to detect the type of ordering between event-event pairs, and two other networks similarly for time-event pairs.

These networks were trained and evaluated on the Timebank 1.2 (James et al., 2006) and AQUAINT TimeML (Verhagen & Moszkowicz, 2008) (here after referred to as the Opinion Corpus) corpera, using the values identified in the lexical processing and event extraction steps to build features for the data. During training, the datasets from these corpera are passed through the networks, causing the networks to adjust their weights and learn what configuration best represents the data. Training occurred on 70 per cent randomly selected data rows and the networks are then tested on the remaining 30 per cent. During testing, the test data is passed through the networks and the outcome is compared to the expected, producing an precision score for the network's classification. Testing indicates how well the networks perform on unseen pieces of data.

A transitive ordering algorithm is then applied to increase the number of orderings between events using transitive properties. For example, checking whether an ordering relation between events E1 and E2 and an ordering relation between E2 and E3 can imply ordering relation between E1 and E3. The output of this stage is then the identified ordering between extracted events which can then be used to identify likely preconditions and postconditions for actions in the narrative planning model (an area for future research).

3.4 Evaluation

Evaluation is important in determining the validity and effectiveness of approaches used (Stubbs & Pustejovsky, 2012). Evaluation is used on both the event extraction step and the temporal event ordering step. Evaluation is absent for the lexical processing as the majority

of techniques used come from existing research and libraries (Stanford CoreNLP) and thus is not our own work.

While the evaluation of these two steps assess different goals and make different measurements, the fundamentals are similar. We have a set of test data which we pass through each step and compare it to a ground truth. These ground truths or gold standard represent what the expected "correct" output of the step would be (Stubbs & Pustejovsky, 2012). For the event extraction step, the data set used is the set of Hansard Parliamentary speeches (Australian House of Representatives, 2017) and the ground truth is what the researchers interpret as being valid extractions. The event ordering step on the other hand uses the Timebank 1.2 corpus (James et al., 2006) and the Opinion corpus (Verhagen & Moszkowicz, 2008) as data with the ground truth being the expected results labeled in these corpera. Both of the data corpus and exact evaluation method is further explained in their respective part. While the exact evaluation output of either step will represent slightly different measures, they generally indicated the effectiveness of the approaches used. This helps identify both the strengths and the limitations inherent within the system.

We also apply significance testing on the results in both the event extraction evaluation and temporal event ordering evaluation. This is to determine whether the results we are comparing to are significantly different to the results we produce. For both evaluations we use a two-tailed t-test on the evaluation scores to detect significance. In both cases this is done using a critical value of 0.05 to determine if we reject the null hypothesis (the results are not significantly different). The null hypothesis is rejected if the found t-score is used to produce a p-value lower than the critical value (Haslam & McGarty, 2003). This allows us to draw conclusions based on concrete statistical methods and not based just on human intuition.

4 Lexical Processing

The lexical processing step of the pipeline focuses on preparing and annotating the input text for the later steps of the pipeline. This takes unstructured text as input and outputs the text in a more structured format annotated with additional semantic information. The text inputs are also assumed to be grammatically correct as otherwise incorrect parsing would likely occur and cause errors in subsequent pipeline parts. At the core of the lexical processing is the use of the Stanford CoreNLP API, a Java API implementation of many state of the art NLP annotators and processes developed by Stanford researchers (Manning et al., 2014). These annotators indicate additional semantic information, such as part of speech information or named entity information, which is inferred from the raw input text. The output is then the text with all additional information indicated by the annotators.

4.1 Stanford CoreNLP Annotators

Stanford CoreNLP offers a large range of automated annotators. The following six annotators are used in this research:

First, **MorphaAnnotator** is used for lemmatization as follows: Lemmatization is the process of identifying the base version of each token with aspects like plurality or tense being absent (Manning et.al, 2008. For example, the lemmatized version of *"walked"* would be *"walk"* and the lemmatized version of *"cars"* would be *"car"*. The aspects that are removed are considered inflectional endings. This improves the matching and comparisons of words due to the reduce state space and is necessary for subprocesses used later in the pipeline which rely on the basic versions of words.

Second, **POSTaggerAnnotation** is used for part-of-speech tagging as follows: It identifies the part-of-speech of each token in terms of token tense and their syntactic function

(Toutanova et al., 2003). For example, the verb *"runs"* in the context of *"John runs"* would have a part-of-speech of VBZ, indicating it is a verb, in the 3rd person, and has present tense. These part-of-speech annotations are defined by Universal Dependencies Specification [1]. These are used in later processes to identify entities, actions and modifiers based on semantic structure within the input text.

Third, **NERClassifierCombiner** is used for named entity recognition as follows: It identifies the named entities (such as a person or location), numerical entities (such as money or per centages) and temporal entities (such as dates or time) (Finkel et al., 2005). This information is used to aid the identification of entities and phrases present in the text.

Fourth, **ParseAnnotator** is used for constituency parsing as follows: It identifies sentence structure split into grammatical phrases and token types (Klein & Manning, 2012). For instance, a constituency parse of *"The dog played"* can be seen in Figure 3 below where *"the dog"* is a noun phrase and ''*played"* is a verb phrase. This parsing is used in later processes that rely on tree like semantic structure to identify grammatical aspects such verb types or phrase relations.

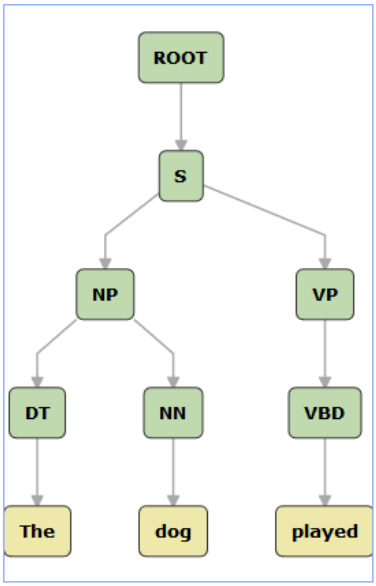

Figure 3: Constituency parsing on *"the dog played"* example.

Fifth, **DependencyParseAnnotator** is used for dependency parsing as follows: It identifies relations between tokens based on the Universal Dependencies Specification[1]. This indicates how tokens rely on each other and casually relate in terms of semantic structure and meaning (Chen & Manning, 2014). For example, Figure 2 shows the dependency relations between tokens through the linking arrows. The child of the connection is considered dependent based on the relation to the parent of the relation, the governor. In the example, *"brown"* is the dependent of the *"dog"* governor through an *amod* relation. Each relation indicates the semantic effect the dependent token has on the governor. In this case *amod* indicates an adjectival modifier. These relations are the core of the event extraction as they indicated semantic meaning. The *EnhancedPlusPlusDependenciesAnnotation* version of this annotator is used to provide the most amount of dependency information available along with the default Stanford English model.

Sixth, **Coref Annotator** is used for coreference resolution as follows: It identifies which entities pronouns are likely to refer to in text (Recasens et al., 2013; Lee et al., 2013; Lee et al., 2011; Raghunathan et al., 2010). This means identifying which noun *"he"* or *"she"* likely refer to. This is used to replace pronouns with the nouns they likely refer to. This allows linking entities by name more consistent as the absence of pronouns reduces the

ambiguity in identifying actor entities. The output of this coreference resolution process is the input text with resolved references, which is then passed through the other annotators to reprocess the modified text.

4.2 Extended Semantic Mapping

An additional modification we make to the text is a rule-based method that looks at conjunctive words before passing it through Stanford CoreNLP (to be then passed down the pipeline). We pass the text through Stanford CoreNLP first to identify semantic relations where the rule-based method then looks at conjunction dependency relations between tokens and carries across descriptive dependencies to the connected word. For example, the sentence *"I am joyful and happy"* is mapped to become "*I am joyful and I am happy"*. The words that are carried across are those with dependency relations to the original token excluding *clauses*, *coordination* and *other* (see Universal Dependency Relations specification[1]) as they are likely shared with the conjunctive word providing. The tokens are carried across only if the connecting word does not already have a token of that relation.

This modification has been seen to improve the extraction in later stages as Stanford dependency parsing usually restricts tokens to having one relation governor and thus tokens that are conjunct with another can miss descriptive relations from the governor word, making identifying causal relations more difficult. This process enables us to link these words together through duplicating them over the conjunction and still retain semantic structure and meaning. The text resulting from this method is then passed through Stanford CoreNLP again to re-annotate the text with the additional words before moving onto the next step in the pipeline.

4.3 Time Entity Parsing

The lexical processing also uses Stanford's SUTime library to annotate time expressions in text (Chang & Manning, 2012), which are then converted into a comparable value. SUTime identifies *dates* (e.g., *"17th October 2017"*, *"Tuesday next week"*) *times* (e.g.,*"9:30am"*, *"in 30 minutes"*), *durations* (e.g., *"from 1:00 pm to 3:00 pm"*, *"48 hours"*) and *sets* (e.g., *"every Tuesday"*) and provides a value string representing that object. We then take these values strings and create an object with a comparable start and end time where the start time is the initial value and the end time is the last possible value of that period. For example, *"2017-3-2"* would have a start value of *"2017-3-2"* and an end value of *"2017-3-3"* as we would have no other lower level of detail than a *day* value and as such an event that took place at that time could have occurred anytime between the start and end time. This value conversion is limited to *dates* and *times* and some *duration* values as some *duration* values and *sets* lack in sufficient information to convert. For example, when extracting a duration between two times (e.g., *"Tuesday to Friday"*) SUTime provides annotations similar to "⟨ *TIMEX3 beginPoint='t1' endPoint='t2' tid='t3' type='DURATION' value='PT0S'* ⟩", where the begin and end points indicate that the boundaries of the duration. The problem with this is that SUTime does not annotate the value that *"t1"* or *"t2"* refer to and as such it is difficult to identify in the text what these values are. Subsequently, *set* values can be difficult to express in a comparable data type and thus are excluded from this stage.

For the case of partial times where higher level time values are not specified (e.g., *"7th of August"* or *"Tuesday in October"* lack sufficient information), the specified document time is used to fill in the gaps. For example, if the specified document date is *"2017-08-27"* and *"7th of August"* is present in the text with no specified year, then the resulting date would be *"2017-08-07"*. While this might be different to what was intended in the original text, it is easiest and most reliable way to create a comparable date. In later parts of the pipeline these times are compared to events to determine if there is ordering between the time and event. This ordering is then used along with other time-event orderings to

determine additional orderings between events based on their associated time. Thus, the actual value of the time is less significant than its order and we expect this method to resolve partial times to have little impact.

5 Event Extraction

The event extraction step of the pipeline is concerned with identifying relationship triples within the text. Relationship triples indicate a relationship between two entities in the form of **{Entity}** ***Relation*** **{Entity}** (Mausam et al., 2012) and will become the basis for the actions and states in the narrative planning model. In these triples, the left entity (the actor) holds the relation to the second entity (the subject) and are usually interchangeable with slight grammatical modifications. For example, {Will} *hit* {the ball} is interchangeable with {the ball} *hit by* {Will}. Thus the ordering of entities used in the extracted triples is reliant on the original sentence structure and token ordering. Three different extraction types are identified, *events*, *possession* and *properties* with *events* populating most extractions. Subsequently, most *possessions* and *properties* are extracted similarly to *events*, but have an indicative relation verb (*'has'* or *'is'* respectively).

This section explores the event extraction step of the pipeline. First we discuss previous work relating to event extraction in Section 5.1 and then we discuss our approach to event extraction in Section 5.2. We then illustrate how we extract some triples based on nouns in Section 5.3 and how we retain causal semantic information through the use of modifiers in Section 5.4. In Section 5.5 we then discuss additional attributes we identifying pertaining to the triples such as event type and tense and in Section 5.6 we discuss how we evaluate this step of the system and present some results from it.

5.1 Related Work: Event Extraction

The concept of extracting information has been around for a while and with the recent rise of statistical methods over large unstructured information it has been at the forefront of NLP research. Information extraction is the process of automatically extracting structured information from text documents (structured or not) (Manning et al., 2008). This extraction can be split into several subtasks, *named entity recognition* (NER) to identify important entity names (e.g., identifying *"Will"* as the name of a person in a sentence), *coreference resolution* to link textual entities (e.g., identifying which noun *"he"* refers to in the text), *relationship extraction* between entities in text (e.g., the triples mentioned earlier in this section) and *extraction of domain specific structured information* (e.g., extracting comments or reviews from web-pages pertaining to a topic of interest) (Manning & Raghavan, 2008). These tasks have been applied in many fields including automated question answering (Srihari & Li, 2000), information learners (Mitchell et.al, 2015) and advertising (Pham & Pham, 2012). In this paper, we focus on identifying relationships between entities in text in the form of relationship triples (Entity, Relationship, Entity) as these can be converted analogous to actions, which are the basis for the planning approach to narrative model generation.

Approaches to information extraction are usually either rule-based approaches or machine learning approaches (often combined with rules to improve classification). Rule-based approaches focus on using semantic and/or syntactic rules that express expert knowledge to extract relevant information and machine learning approaches focus on statistical techniques and learning extraction rules from data. While statistical and big data analytics approaches are timely and popular (Chiticariu et al., 2013), work has still been done on these rule-based methods which has yielded promising results. These rule-based methods are seen to work effectively when the text is in a structured or semi-structured format as rules can exploit this format to identify relevant information.

Recent work on information extraction (Chenn, An & Zeng, 2015) uses a rule-based approach to extract information from semi-structured *human-readable scientific* (HRS) documents. The information extracted from these documents allows the important information to be automatically extracted and organized. Their method splits the HSR into different regions based on a distinctive structure or semantic boundary (e.g., the header region) and split them into regions of interest (or not) and entities of interest (or not) based on the type of region it is and the qualitative data present. These regions are then processed to identify important information based on surrounding tags or regular expressions, which are then mapped to the database templates. While the focus of this research is different to the goal of this paper, it demonstrates how rule-based methods can be used to effectively extract information from a structured format.

Similar work looks at applying rule-based information extraction to structured clinical documents (Mykowiecka et al., 2009). The system developed extracts information based on several grammatical and semantic heuristics in relation to the clinical document format, identifying information such as patient and treatment information. This research addresses several linguistic issues such as ambiguity, negation, coordination and anaphoric expressions. For majority of tested samples, precision and recall scores above 80 per cent are obtained, indicating accurate and effective retrieval. This research further demonstrates the effectiveness of rule based methods in terms of information extraction.

Machine learning-based information extraction refers to using machining learning models to extract information (e.g., naïve-bayes, support vector machines) which have learned from previous data sets. These approaches are a way of using these well-defined and researched machine learning techniques to avoid the labour and uncertainty of manually defined rules. Due to the large research interest in machine learning techniques and information extraction being largely seen as a classification task, these types of approaches for information extraction have been popular (Chiticariu et al., 2013). Machine learning approaches can be supervised, learning off data which indicate the correct classes, or unsupervised, learning from data with the correct classification absent, often clustering them together or learning patterns. These approaches also often incorporate aspects of rule-based methods to preprocess text and define features to help improve classification. *TextRunner* (Yates et al., 2007) was one of the first Open Information Extraction systems to take advantage of this where system makes a single pass over the given corpus to extract relations without the need of any human input. *TextRunner* uses a classifier trained on labeled data to determine whether a relationship between two noun phrases should be extracted. An unsupervised probabilistic model called *Resolver* is then used to link extractions that indicate the same relation. Evaluation of *TextRunner* returned 80.4 per cent accuracy as deemed by human reviewers.

WOE^{parse} (Wu & Weld, 2010) introduced an information extraction method built on unsupervised learning from Wikipedia articles. *WOE* used the structured format of Wikipedia pages to bootstrap training information for a classifier. Training data was created through comparing the Wikipedia info boxes (relational information in a structured format) to the sentences that express that information (unstructed text). This presented a novel method of generating testing data that could be used to effectively train an information extraction classifier, achieving improved results over *TextRunner*.

OLLIE (Mausam et al., 2012) improves upon previous relation extractors by basing relations not only on verbs, but also on nouns and adjectives. This system also further defines some clausal relations between events as defined in text. *OLLIE* learns open pattern templates from datasets bootstrapped from the CLueWeb corpus, a large set of triple style information relations. Extraction patterns are then learned from this dataset based on semantic structure between tokens present along with lexical rules which generalise the patterns based on word similarities. These patterns are then matched to the semantic structure of text to identify new relation triples. This system further improved on previous systems evaluation scores.

Comparing these rule-based approaches and machine learning-focused approaches, each method has distinguished strengths and weaknesses. Machine learning methods can effectively learn information extraction formats without the need of any expert knowledge or manual labour in defining the rules. These methods, however, require large amounts of data and can be difficult to set up. These methods are also black box in the sense that the inner workings of classifiers are difficult to intuitively explain. Rule-based methods on the other hand are easy to understand, define and verify. They do, however, require manual labour to define and often require expert or domain knowledge to be effective (Chiticariu et al., 2013). With these considerations, this thesis focuses on a rule-based approach to identifying triples. This approach was chosen due to the text being in a semi structured format (after the lexical processing stage) where rules could easily be applied and as we do not have currently any pre-labelled data, a machine learning centralised approached seemed less appropriate.

5.2 Identifying Triples based on Verbs

Event extraction is performed through using manually defined semantic rules that represent expert knowledge. These rules look at the part of speech and dependency relations (both identified in the lexical processing stage) to identify likely triples. These dependency structures present a structured tree format of token relations over a finite set of dependencies where rules can effectively be used to identify relevant information triples. The dependencies identified by Stanford CoreNLP (Manning et al., 2014) each depict a specific relation between tokens and these can be adapted to identify relation triples. The definition of each of these relations is specified by Universal Dependencies[1]. The triple extraction is based around non-auxiliary verbs within the text and while some nouns indicate events (*"birthday"*, *"baseball game"*, etc.), extracting actor and subject of verbs is much simpler and consistent than nouns in terms of the sentence's typed dependency structure. In this case, auxiliary verbs are any verbs which is the dependant of an *aux* relation to another token in the sentence, shown in an example sentence in Figure 4 below where *"will"* is an auxiliary dependent to *"walk"*. An *aux* relation indicates that the dependant (always a verb) is an auxiliary (used to define tense, mood, and voice) to the governor (always a verb) and thus an extraction on these words is less informative than on the governor verb. Other examples of auxiliary verbs include *"could"*, *"will"* and *"must"*. Additionally, all non-auxiliary verbs require an actor and/or subject, allowing a valid event triple to almost always be identified. While events can be expressed through nouns, information identified by noun events will often be captured through a verb. For example in *"James had a birthday"*, the *"birthday"* event is captured by the *"had"* verb in {James} *had* {a birthday}. However, at this stage the event would be centralised around the verb rather than the noun.

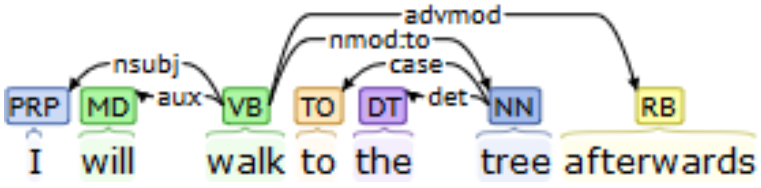

Figure 4: Auxiliarly verb relation example.

With each of the main event verbs identified, the actor and subjects are identified through the dependency relations between that verb and other tokens. There are many dependencies identified by the EnhancedPlusPlusDepenency annotator in Stanford CoreNLP (Manning et al., 2014; Chen & Manning, 2014) and the ones linking the chosen verb and others that indicate triple information come from four groups: *dependants* of the verb that indicate a subject entity, tokens that are *dependants* of the verb that indicate an actor entity, *governors* of the verb that indicate a subject entity and *governors* of the verb that indicate an actor entity. The dependencies that fit into each category where chosen through observing the

types of sentences that demonstrate them and understanding the type of triple information that they indicate. Rules representing these variables are given below where *gov(t)* gets the list of dependency governors of token t and *rel(v, t)* gets a dependency relation between tokens v and t where *rel(v, t)* is equivalent to *rel(t, v)* as there can be no more than one relation between two tokens. From the example in Figure 4, $gov(I) = \{will\}$ with $rel(I, walk) = nsubj$ and $gov(tree) = \{walk\}$ with $rel(tree, walk) = nmod$ (additional information on relations like *:to* are ignored as it is not important at this stage). The definition of the relations are defined by Enhanced Universal Dependencies specification[1], for example, *dobj* refers to a direct object noun, *xcomp* refers to an open clausal complement which is a clausal complement without its own subject (e.g., a verb that is the subject of a triple), *rcmod* is a relative clause modifier of a noun phrase, and *nsubj* refers to a nominal subject which is the syntactic subject or the parent token.

$$L_{dep}(V) = \{t | V \in gov(t) \text{ and } rel(V,t) \in \{csubj, expl, nsubj, xsubj\}\},$$
$$L_{gov}(V) = \{t | t \in gov(V) \text{ and } rel(V,t) \in \{acl, vmod\}\},$$
$$R_{dep}(V) = \{t | V \in gov(t) \text{ and } rel(V,t) \in \{acomp, dobj, iobj, nmod, xcomp\}\}, \text{ and}$$
$$R_{gov}(V) = \{t | t \in gov(V) \text{ and } rel(V,t) \in \{rcmod\}\},$$
$$\text{where} \quad V \notin \{AuxiliaryVerbs\} \text{ and } V \in \{Verbs\}.$$

For each non-auxiliary verb, lists of left and right candidates for triple extractions are made based on the rules above. For the example of Figure 4, *"walk"* is seen as a non auxiliary verb and is a governor of *"I"* with an *nsubj* relation and is also a governor of *"tree"* with an *nmod* relation. Thus using the extraction rules, $L_{dep}(walk) = \{I\}$ and $R_{dep}(Walk) = \{tree\}$, indicating that *"I"* is a left triple entity (an actor) and *"tree"* is a right triple entity (a subject) of *"walk"*. Thus the resulting extraction at this stage is {I}*walk*{tree}. Additional tokens such as *"to"* and *"will"* are implemented as modifier relations (discussed in Section 5.4).

An additional rule is put in place for *"be"* verbs (verbs with a lemmatization root equal to *"be"*) as the only dependency relations these have is to the a single entity that would take place in the triple and thus cannot identify left and right entities through the method above. For example, from the sentence *"James is happy"* the extraction of {James} *is* {happy} would be desired but *"is"* is not linked to *"James"*, only to *"happy"* through a *copula* relation, as seen in Figure 5 below.

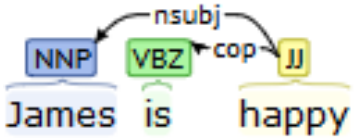

Figure 5: *"James is happy"* dependency parse tree.

Therefore *"James"* could not be identified as the actor to *"is"*. In these cases, the extraction is centralised on the right entity of the *"be"* verb, using the left entity from this extraction in the triple, allowing a {James}*is*{happy} extraction to be made. Note that these rules are run on every verb even if is not a *"be"* verb but will only find entities if the verb is the child of a *cop* relation (which indicates the same information regardless of the verb type). These rules are as follows:

$$R_{cop}(V) = \{t | t \in gov(V) \text{ and } rel(V,t) = cop\} \text{ and}$$
$$L_{cop}(V) = \{L(t) | t \in gov(V) \text{ and } rel(V,t) = cop\}.$$

Given these propositions and the ones before, we can form the following rules:

$$R(V) = R_{dep}(V) \cup R_{gov}(V) \cup R_{cop}(V) \text{ and}$$

$$L(V) = L_{dep}(V) \cup L_{gov}(V) \cup L_{cop}(V).$$

These two rules give us the left and right entities for the triples and are called on each non-auxiliary verb. Thus the right entities of a non-auxiliary verb is the union of the right entities found through the *dependent* relation rules (the verb is the *governor*), the right entities found through the *governor* relation rules (the verb is the *dependent*), and the right entities found through the *copula* rule (usually only if the verb is a *"be"* word). Similarly, the left entities of the verb is the union of the left entities found through the *dependent* relations, *governor* relations and *copula* rule.

From this point, we now have a list of triple extractions centralised around verbs, each with lists of left and right entities which can be converted into actors and subjects. Some verb actors may, however, be absent due to not being directly related in the dependency graph but semantically be part of the extracted triple. Cases of this may be due to *conjunct* relations, for example, in *"James hit and ran"*, *"James"* is not directly linked to *"ran"*, which instead connected to *"hit"* through a *conjunct* relation, and yet would be considered the actor. Aforementioned, we talk about using these relations to expand the semantic mapping (Section 4.2), that is, the above sentence becomes *"James hit and James ran"*, but found that this sometimes failed due to complex dependency tree structures and therefore this section is a way to capture those misses. Similarly, cases can occur for *advcl* and *parataxis* relations. *Advcl* relations identify adverbial clauses to the token they modify, that is, in *"John hummed while thinking" "thinking"* is an *advcl* to *"hummed"* (Figure 6), and *parataxis* relations indicate when two phrases are placed side by side without any coordinating-like words, essentially indicating coordination without explicitly stating it, that is, in *"Let's face it we're annoyed" annoyed* has a parataxis relation to *Let's* (Figure 7).

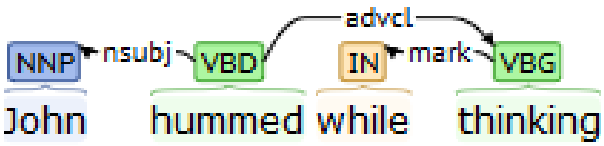

Figure 6: *"John hummed while thinking"* dependency parse tree.

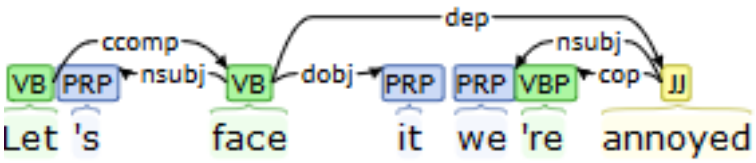

Figure 7: *"Let's face it we're annoyed"* dependency parse tree.

These relationships we identified as they were seen in cases where they linked two verbs and indicated the same actor. In these cases, we recursively check the entities of tokens from those relations until an actor is found, which is then transferred across. This finds the closest actors suitable for the event based on its relations with others provided it is not already in that event subject list. This actor rule is given by:

$$Actor(V) = \begin{cases} L(V), & \text{if } L(V) \neq \emptyset, \\ Actor(T), & \text{if } T \in gov(V) \text{ and } rel(V,T) \in \{advcl, conj, parataxis\}, \text{ and} \\ \{\}, & \text{otherwise.} \end{cases}$$

A similar case is made for selecting the subject of the triple. If the extracted event has no right entities, then the subjects of the same relations are carried across only if the subject has an index greater than the event (occurs later in the text). This captures situations where the subject is later in the sentence but not directly linked to the verb due to a conjunction (e.g., *"*Actor* *Event* and *Event* *Subject*"*). This is expressed in the following rule where $index(t)$ returns the sentence index of token t,

$$Subject(V) = \begin{cases} R(V), & \text{if } R(V) \neq \emptyset \text{ and} \\ \{s | \exists t : (t \in gov(V) \text{ or } V \in gov(t)) & \text{otherwise.} \\ \quad \text{and } rel(V,t) \in \{conj, parataxis, advcl\} & \\ \quad \text{and } s \in Subject(t) \text{ and } index(s) > index(V)\}. & \end{cases}$$

Extractions that have no found actors and subjects are then excluded from the extraction list. In cases where extractions have multiple actors and/or subjects, an extraction is created for all combination of singular actors and subjects which are then linked together in later stages. Given this and the previous rules, extractions made based on a non-auxiliary verb can now be defined as

$$Extractions = \{\{A\}V\{S\} | V \notin AuxiliaryVerbs, V \in Verbs, \\ A \in Actors(V), S \in Subject(V)\}.$$

5.3 Triples Based on Nouns

On top of the event triples that have been extracted, additional relations are extracted through looking at relations between nouns within the text. These extractions are less complex than the previous section and simply use single relations to identify relations triples that is implied through the text, but not explicitly stated. The relations that are used are *possessive*, *appositional* or *adjective* modifiers and are copied across through conjunctive relations.

Possessive modifiers are used to identify *possession* relations to produce triples such as {Actor} *has* {Object}. For instance, *"Will's dog is..."* has a *possessive* modifier between *"Will's"* and *"dog"* and as such is used to identity a triple of {Will} *has* {dog}. While this collapsed triple (*"William has dog"*) is not grammatically correct, it could easily be fixed through adding a determiner to the subject based on its plurality.

Appositional modifiers are used to express a relation where the following noun immediately modifies the initial. For example, *"Jack, John's brother, ..."* has an *appositional* relation between *"Jack"* and *"brother"* to express that *"Jack"* is modified by the tokens between the commas. Triples extracted from these relations are of the form {Actor} *is* {object}, e.g., {Jack} *is* {brother}. In these cases, the entity inside of the commas is said to be a property of the entity outside the commas and is usually interchangeable depending on the subject of the text.

Lastly, *adjective* modifiers are used to express when an adjective modifies a noun and are used to produce triples of the form{Actor} *is* {Adjective}. For example, in *"the brown dog ran..."* the triple of the dog *is* brown is extracted due to the adjective modifier between *"brown"* and *"dog"*. The purpose of these is to explicitly express adjective properties as their own triple as they describe entity properties and may affect how they act or are interacted with.

5.4 Modifier Relations

At this stage, a list of events with actors and subjects have been identified, but in the process some information may have been lost such as such as adjectival modifiers, compound nouns,

or clausal relations. All the events and entities at this point are expressed through single tokens based on dependency relations and as such additional rules are introduced to capture the lost information. The modifier rule looks at each part of the triple and annotates it with all relations and tokens that it governs in the dependency graph (provided it is not already present in the triple). The rule is then repeated for those annotations until there are no more relations that have not been expressed. As a result, each part of the triple is recursively annotated with relations they govern over in the sentence structure, capturing all information present in the original sentence.

These modifiers serve two main purposes. The first purpose is to be able to print the triple as it grammatically appears in the sentence and the second purpose is to retain all clausal modifiers, which may be valuable when using the triple. For displaying the triple we would want it to retain the same grammatical structure, displaying tokens such as *"and"* or *"the"* which may not provide any particular insight into the extraction, but instead provides a readable output for it. This also allows us to easily evaluate and understand what the extraction is expressing. Due to these modifiers being identified recursively from dependency relations, there can be a large amount and as such the ones that are displayed in the readable output are limited to specific relations based on the extraction part (*actor, relation, or event*) and level (e.g., is the word a direct modifier to an extraction part or is it a modifier to a direct modifier)

Similarly other modifiers are annotated such as clausal modifiers or conjunctive modifiers which may affect how the extraction can be used. While in this research none of these modifiers are used, they are expected to affect how the extractions are converted into the narrative planning model actions as these modifiers provide insight into the conditions of using the triple (area for future work).

In addition, similarly to finding actors and subjects, some modifiers are transferred from conjunct tokens directly to the entity. The modifiers that are transferred are based on the original sequence and the number of modifiers the entity already has. If the entity already has a modifier from the group that exists in the transferred sequence, then only the modifiers before that one is transferred across. For example, in *"of the red apple and the grape"* the *"apple"* entity has a modifier sequence of *case-determiner-amod* (Figure 8) and only the case modifier is carried across to *"grape"* due to *"grape"* having a determiner modifier, making only the modifiers left of this to be transferred across. Similarly, if entity has an existing modifier in the same group, it will not be transferred. For instance, *"a green apple and 5 oranges"* *"apple"* has a sequence of *determiner-amod* and none are carried across to *"oranges"* due to it having a *numod* (*"5"*) modifier which is absent from the sequence (Figure 9). The index of the transferred words is also used; the transferred word needs to have the same ordering with the entity as it has with its parent in the original text.

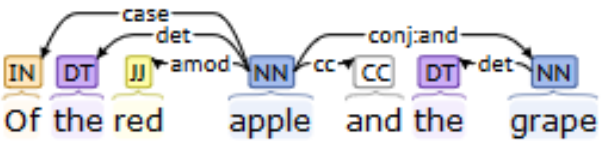

Figure 8: *"Of the red apple and the grape"* dependency parse tree.

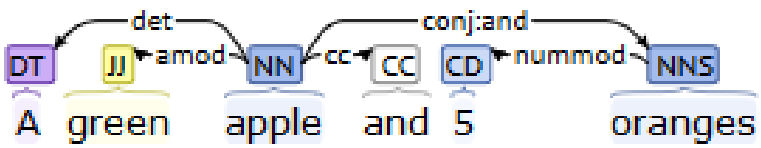

Figure 9: *"A green apple and 5 oranges"* dependency parse tree.

The group of relations transferred are defined below where $modifiers(E)$ gets the list of modifiers identified for token E,

$$TransferGroups_{entity} = \{\{case\}, \{det, numod\}, \{acl\}, \{amod, neg, nmod\}\},$$

$$TransferSeq_{entity}(E) = x_1, ..., x_k | x_i \in modifiers(E),$$
$$rel(x_i, E) \in TransferGroups_{entity} \text{ and } index(x_i) < index(x_{i+1}), \text{ and}$$
$$TransferRel_{entity}(E) = x_1, ..., x_k | x_i = rel(E, TransferSeq_{entity}(E)_i).$$

If two entities, E and T, are then connected through a conjunction and the groups of $TransferRel_{entity}(E)$ is a subset of $TransferRel_{entity}(T)$ and $index(T) < index(E)$, add *el* from $TransferSeq_{entity}(T)$ to $modifiers(E)$ if the position of its relation in $TransferRel_{entity}(T)$ is to the left of the first relation type in $TransferRel_{entity}(E)$).

This captures cases where tokens are semantically shared through a conjunction but missing the typed dependency and thus could not be directly linked and ensures only the shared information is passed. In the example sentence of Figure 8, $TransferSeq_{entity}(apple) = Of, the, red$ with $TransferRel_{entity}(apple) = case, det, amod$ and $TransferSeq_{entity}(grape) = the$ with $TransferRel_{entity}(grape) = the$. Only *Of* is added to $modifiers(grape)$ as grape has a *det* modifier and thus we only transfer the modifiers from $TransferSeq_{entity}(apple)$ that has a relation to the left of a *det* relation, ruling out *the* and *red*. Similarly, if $TransferSeq_{entity}(grape)$ was empty, *Of, the, and red* is all added to $modifiers(grape)$. In the other example (Figure 9), $TransferSeq_{entity}(apple) = A, green$ with $TransferRel_{entity}(apple) = det, amod$ and $TransferSeq_{entity}(oranges) = 5$ with $TransferRel_{entity}(oranges) = nummod$. Thus none are carried across to $modifiers(oranges)$ as $TransferSeq_{entity}(oranges)$ is not a subset of $TransferRel_{entity}(apple)$. Note that these rules are only used on tokens which have been identified as an entity from the earlier extraction rules.

The same process is used in transferring modifiers between conjoined events, the only difference being the groups that are used. These group are:

$$TransferGroups_{event} = \{\{case\}, \{det, aux\}, \{advcl\}, \{advmod, neg\}\}.$$

One additional minor rule for events is if the object entity is connected to the event verb through a *nmod* relation, then the *case* token that defines the relation is carried across to the verb. For example, {I}*played*{with the dog} would be changed to {I}*played with*{the dog} as *"dog"* was connected to *"played"* through a *nmod* relation with *"with"*. This simply allows the relation of the triple to more closely represent the action being made.

From these modifier rules, we now have each part of the triple labeled with additional information that affects how each part is understood, and the understanding of the triple holistically. However, currently in the system none of these modifying tokens are taken any further beyond identification in terms of understanding how the they affect the reasoning and information on the triple; this is an area for potential future research.

5.5 Additional Extraction Information

Following, each extraction is annotated with additional information that aids their processing in the later parts of the pipeline. The additional information identified is event type, grammatical aspect, tense, direct negation and modality. Each of these are found

through rule based methods based on semantic information identified in the lexical processing. Other information extraction systems (e.g.,OLLIE (Mausam et al., 2012)) define these relations into specific classes, but as our modifiers are defined over the finite set of dependency relations (from Universal Dependencies[1]), we are not concerned with further categorizing them into classes. For instance, in the sentence *"I will buy the book if they have it"* (Figure 11), the *buy* event would have the *have* event as adverbial clause modifier by *if* (*advcl:if*)and as such can be reasoned about without having to further define the modifier class. In this example, other extractors may have the *buy* event be *AttributedTo* the *have* event.

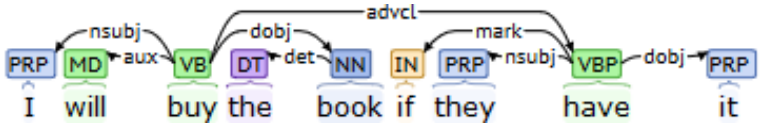

Figure 10: *"I will buy the book if they have it"* dependency parse tree.

5.5.1 Event Type

The event type refers to how the extraction acts with respect to its primary verb. Event class is determined by looking at the verb used for the event and using the parse tree of the sentence based at the extraction object phrase. The classes and rules used to find them can be seen in the Table 1.

Event class	**Rule**
PROPERTY	If the verb is a *"be"* token or if the object phrase is an adjective phrase (*ADJP*).
POSSESION	If the verb is a *"have"* token.
TWO_PLACE_TRANSITIVE	If the object phrase starts with two Noun phrases (*NPNP*) or starts with a noun and then pronoun phrase (*NPPP*).
TRANSITIVE_ACTION	If the object phrase starts with a noun phrase (*NP*).
INTRANSITIVE	Otherwise.

Table 1: Event classes and rules.

5.5.2 Grammatical Aspect

The grammatical aspect denotes how the main verb extends over time and is extracted through looking at the relation type between the event verb and its modifiers. Grammatical aspect can be identified through the type of auxiliary verbs it has and the *part-of-speech* (POS) that the verb has. This can be seen in the following Table 2.

Grammatical Aspect	**Rule**
PROPERTY	If the verb is a *"be"* token or if the object phrase is an adjective phrase (*ADJP*).
PERFECT	If the verb has a POS of *VBD* (past tense verb) or *VBN* (past participle verb) and has an auxiliary modifiers with a lemmatization of *"will"* and *"have"* or just *"have"*.
PERFECT_PROGRESSIVE	If the verb has a POS of *VBG* (present participle verb), *VBP* (non 3rd person singular present verb) or *VBZ* (3rd person singular present verb) and has auxiliary modifiers with lemmatization of *"will"*, *"have"* and *"be"* or *"have"* and *"be"*.
PROGRESSIVE	If the verb has a POS of *VBG* (present participle verb), *VBP* (non 3rd person singular present verb) or *VBZ* (3rd person singular present verb) and has auxiliary modifiers with lemmatization tokens *"will"* and *"be"* or just *"be"*.
SIMPLE	Otherwise.

Table 2: Grammatical Aspect and rules.

5.5.3 Tense

The tense of the extraction is the tense of the main verb to describes when it took place in time. The tense can be extracted, similarly to Grammatical Aspect, through looking at the POS of auxiliary modifiers and mark modifiers the main verb has. This is seen in the Table 3.

Tense	**Rule**
FUTURE	If the verb has a *TO* part of speech (indicating "to") or has a *IN* part of speech (indicating a preposition) and the original text of *"until"*.
PRESENT	If the outermost aux modifier (verb included) has a part of speech of *VB* (base verb), *VBZ* (3rd person singular present verb) , *VBP* (non 3rd person singular present verb) or *VBG* (present participle verb).
PAST	If the outermost aux (verb included) modifier has a *VB* (base verb) POS.
NONE	Otherwise.

Table 3: Tense and rules.

If the outermost verb is a modal word (*MD* POS) then the tense is instead based on the word from the three groups in Table 4.

Tense	Rule
FUTURE	{*"will", "shall", "must"*}
PRESENT	{*"need"*}
NONE	{*"could", "would", "may", "might", "should", "can"*}

Table 4: Tense and rules for modal connections.

5.5.4 Modality

Modality indicates the type of modal words that the extraction has. These are found through looking at the auxiliary modifier and open clausal component lemmas that the main verb has. The rules used are defined in table 5 below.

Modality	Rule
CAN	Has a auxiliary of *"can"* or has a *"have"* auxiliary which is part of a *"have to be able to"* string.
MAY	Has a *"may"* auxiliary modifier.
MUST	Has a *"must"* auxiliary or has a *"have"* auxiliary which is part of a *"have to"* string.
SHALL	Has a *"shall"* auxiliary.
WILL	Has a *"will"* auxiliary.
NONE	Otherwise.

Table 5: Modality and rules.

5.6 Evaluation

The main aspect we wish to evaluate for this extraction systems is its ability to extract triples from general input text. There are two scores generally associated with evaluating these types of systems, recall and precision (Manning et al. 2008). For information extraction, recall refers to the ability of the system to identify triples that are in available from the text, often being a score of the number extractions made out of the possible. Recall can be measured through the following formula:

$$recall = \frac{|\text{correct extractions}|}{|\text{expected extractions}|}.$$

In the case of recall, the ground truth (or gold standard results) would be what the researcher expects should be extracted from the original text. This ground truth is used as we do not have a corpus with this already indicated and it is difficult to automate the analysis of the measure due to the complexity of the problem.

In information extraction, precision refers to its ability to produce correct triples, often a score of the number of correct extractions made over the total made. The formula for precision is similar to recall:

$$precision = \frac{|\text{correct extractions}|}{|\text{correct extractions}| + |\text{incorrect extractions}|}.$$

In the case of precision, the ground truth is then what extractions the researchers judge as being correct with regards to the extraction and original text. This ground truth was chosen for similar reasons as recall. Both recall and precision provided a value between 0 and 1 (1 being preferred) and we present it between 0 and 100 to be analogous to a percentage.

These two scores of recall and precision can then be combined into what is known as an F1-measure to provide a measure for the overall extraction (Manning et al. 2008). The balanced F1-measure is a value between 0 and 1 (1 being preferred) and and is defined as:

$$F1 = 2 * \frac{precision * recall}{precision + recall}.$$

This F1-measure is useful as precision and recall are dependent on each other, an increase in precision is usually at the expense of recall and vice versa. In terms of information extraction, increasing precision would mean that the system makes less correct extractions (or more correct extractions) and is likely to be more restricted in what it extracts from the original text, causing less extractions overall. Thus with less extractions, recall is likely to get worse as recall due to it being a measure of the number of correct extractions out of the possible ones. This relation is also reciprocatory, an increase in recall means more extractions are made and thus precision is likely to decrease due to the greater chance of incorrect extractions. Therefore looking at either score in isolation can be misleading on the overall performance of the system and whereas the F1-measure provides a single score representing both.

The data corpus we use for these measures is the Hansard House of Representatives Start of Business Parliamentary Speeches (Australian House of Representatives, 2017).These documents are transcribed speeches made during the start of business during parliament, a time where politicians or representatives can raise an issue or give news to the other politicians. This set was used as each document is self contained, has a maximum size of about 350 words due to a time limit, and were told in a narrative and factual way. However, as they are transcripts of speeches, some of the grammar and sentence structure were very complex and caused errors in sentence parsing and extraction, but this was included in testing to demonstrate some negatives of the approach. This corpus was chosen for these strengths and weaknesses as it allows us to test our system across varied text that highlights the strengths and limitations. The recall and precision of our rule based method are determined through parsing 30 randomly chosen documents from this corpus and having the researcher manually assess the extractions with respect to the two scores.

5.6.1 Precision

Precision is the measure of the number of extractions made that were correct in the context of the sample text (Manning et al. 2008). Measurements were split into three categories: *correct*, the extraction was correct and had all necessary modifiers, *partial*, the extraction was mostly correct lacked some information in terms of valid modifiers or had the wrong entity but had the correct entity as a modifier, and *incorrect*, the extraction was incorrect in the context of the tested text. Over the tested documents, the precision results are shown in Table 6.

Type	Count	Precision
Correct	1398	73.50
Partial	230	12.09
Incorrect	274	14.41

Table 6: Precision results

From the testing 73 per cent of extractions were considered correct, 12 per cent were considered partial and 14 per cent were considered incorrect (Table 6). The majority of incorrect and partial extractions were due to parsing errors of the test text. In some cases, the sentences has unusual or complex grammatical structures and used idioms or phrases which have difficulty being recognized and parsed into valid trees. Some of the parse trees coupled the rules associated actors and subjects that did not match the text. A common occurance was the parser incorrectly categorized tokens after commas to have a "appos" relation with the token before (indicating that the token after is a direct modifier) when they should have had a different relation. Additionally, in some cases the sentence structure was ambiguous in a way that valid extractions could not be made from the extraction tree alone as they required domain knowledge to understand.

A one-sample t-test (Haslam & McGarty, 2003) is used between these precision scores to determine if there is a significant difference between the type . The accuracy between all scores had a pairwise p-value less than the critical value of 0.05, indicating that they are significantly different. This indicates that correct extractions is expected to occur at a significantly different rate than the other two and the partial extractions is expected to occur at a significantly different rate from the incorrect extractions.

For comparison, a naïve baseline is used where each extraction is based off of each verb and the closest left and right nouns and the entities. The results can be seen in the Table 7 with a comparison to our system's precision through a two tailed two sample t-test.

Type	Count-baseline	Baseline precision	Our system precision	p-value
Correct	233	32.22	73.50	< 0.00001
Partial	278	38.45	12.09	< 0.00001
Incorrect	212	29.32	14.41	< 0.00001

Table 7: Precision results with baseline comparison

The results from Table 7 demonstrate that our system's precision is significantly different from the naïve baseline; All p-values are smaller than the critical value of 0.05.

5.6.2 Recall

Recall is the measure of the number of correct extractions identified from the possible extractions in the text (Manning et al. 2008). Measurements were split into three categories: *found*, extractions correctly identified, *partial*, extractions that were identified but had slight differences to what was expressed in text such as missing modifiers or ordering of entities, and *missed*, extractions that were not identified. The recall results can be seen in Table 8.

Type	Count	Recall
Found	1398	73.89
Partial	230	12.35
Missed	264	14.18

Table 8: Recall results

From the testing, 74 per cent potential extractions were found, 12 per cent of potential extractions were found but had slight differences and 14 per cent of potential extractions were missed (Table 8) . Missed and partial extractions were mostly due to parsing errors, similarly to the precision score. In these cases where are considering simple extractions made through adjective-noun relations to be relevant in the extractions made and is one of the reasons why so many extractions are found compared to those that are missed. The missed extractions were often those that in sentences with complex structure and occurred between tokens relatively distance and thus was difficult to detect through the dependency structure alone.

A one-sample t-test (Haslam & McGarty, 2003) is used between these recall scores to determine if there is a significant difference between the type. The found accuracy is significantly different from the other two accuracy scores at the 0.05 critical alpha level, whereas the difference between the partial and missed scores is not significant at the 0.05 critical alpha level. This indicates that partial and missed extractions are to occur roughly at the same rate whereas correct extractions occur at a significantly different rate as the other two.

For comparison, a the same baseline from the precision is used, one which extracts by associating each verb with left and right nouns as the entities. The results for the baseline compared to the recall accuracy for our system can be seen below where they are compared using a two-tailed two sample t-test as seen in Table 9.

Type	Count-baseline	Baseline recall	Our system recall	p-value
Found	233	12.32	73.89	< 0.00001
Partial	278	14.62	12.35	0.0412
Missed	1381	72.99	14.18	<0.00001

Table 9: Recall results with baseline comparison

The results from Table 9 demonstrate that our system's recall results are significantly different from the baseline; All scores have a p-value smaller than the critical value of 0.05.

5.6.3 F1-measure

We use the F1-measure only on the correct precision value and found recall value as we omit partial values. Thus, the balanced F1-measure based on the precision and recall values is:

$$F1 = 2 * \frac{0.7350 * 0.7389}{0.7350 + 0.7389} = 0.7369.$$

This F1-measure of 74 per cent is comparable if not slightly worse to previous extraction system scores. The evaluation environment and method between other system's scores and ours is likely different with different interpretations of what is "correct" and corpera used and thus any comparison can't be taken at face value. Our system, however, has two key advantages over previous systems, it does not require training data and indicates causal and modifier relations (absent from many previous extraction systems (Yates, et al., 2007; Wu & Weld, 2010; Chambers & Jurafsky, 2009)).

There are, however, several distinct limitations to this system. First, it assumes that the sentence structure is grammatically correct with no irregularities (otherwise parsing errors occur) and unable to infer meaning beyond what is presented in the text. For example, from *"Will enjoys the taste of apples"* a triple of {Will} *enjoys* {the taste of apples} will be the only one extracted, whereas you might be able to deduce a triple of {Will} *eats* {apples} based on the implied lexical meaning. This limitation is, however, a difficult task to solve as

it requires general semantic and lexical knowledge. An additional limitation is that it relies heavily on the Stanford CoreNLP parsing in the lexical processing and any errors in this is likely to cause further extraction errors.

Further improvements to this event extraction system would be to incorporate some machine learning techniques demonstrated in other extraction work (Wu & Weld, 2010; Mausam et al., 2012), producing a hybrid extractor that is likely to perform well. This, however, would require large amounts of pre-labeled training data and would take significant time to get the training and configurations correct, more than we currently have in this project.

5.6.4 Evaluating Event Features

Minor aspects of the system we can then evaluate are the event features which we have extracted. We only evaluate the detection of tense due to time constraints and the lack of data for the other aspects. The corpus used is the TimeBank 1.2 corpus (James et al., 2006), a set of 183 manually annotated news articles that contains event information. In the corpus, events are identified in the text along with several aspects, one of which is tense. This corpus is discussed more in the temporal event ordering part of this paper (Section 6) as the corpus also contains temporal relations between events. Thus we use our tense detection on the events in the text and compare it to the annotations in the corpus. We only consider the 4 tense that we identify with our (FUTURE, PRESENT, PAST, NONE) whereas the corpus identifies several additional ones (which are less common). The results for this tense evaluation can be seen in the table below.

Tense	Correct	Incorrect	Precision
FUTURE	182	17	91.46
PRESENT	855	46	94.89
PAST	1499	40	97.40
NONE	175	97	64.34
Total	2711	200	93.13

Table 10: Tense evaluation

From Table 10 we have a total tense precision of 93.13 per cent with past tense being the most accurate with 97.40 per cent and none being the least accurate with 64.34 per cent. This least accurate case appears to be significantly lower than the others which have accuracy in the 90s, indicating that none tense is more difficult to identify than the others.

Inspection of the incorrect results found that in the majority of these cases the tense was ambiguous or required information outside of the sentence or from other tokens not directly linked to the event to determine. Some cases (particularly the *NONE* cases) also lexical knowledge of word meanings to determine the tense. As our system determines tense on isolated sentences and only through the links that the verb has and not their meaning, it is understandable that it fails in these cases.

6 Temporal Event Ordering

At this stage of the pipeline event and property triples have been extracted with additional features identified and we now want a way to link them together. This linkage will form the basis of identifying the preconditions and postconditions of the actions for the planning model. One way of identifying this linkage is through identifying temporal ordering between event pairs. For instance, if an event is seen to consistently occur before another, then the first event is likely a precondition for the second or the second to be a post condition for the

first. Temporal ordering is determined through identifying ordering between even pairs and time-event pairs and joining the mapping through transitive properties. Both use multilayer perception networks to determine different ordering aspects.

This section explores the event ordering step of the pipeline. First we discuss previous work relating to temporal event ordering in 6.1 and then we discuss the corpera we use to approach the problem in 6.2. In 6.3 we discuss how we identify ordering in event-event pairs and in 6.4 we discuss how we identify ordering in time-event pairs. Then in 6.5 we discuss how we use transitive mapping properties to expand the ordering mapping between all events and in 6.6 we discuss the results and limitations of the approaches we use to identify temporal ordering.

6.1 Related Work: Temporal Event Ordering

The problem of determining the order of which events occurs is difficult and has been extensively researched. Events and their relations can be expressed in many ways and require a sense of general knowledge to understand. In many situations, the order of events is inferred through meaning behind the text and not explicitly stated. There are different forms of event ordering too, there is the time of occurrence (when the event occurred), time of reference (the time from which the "speaker" views the event on the timeline) and speech time (the time at which the text was produced), which can be arranged in different ways and be difficult to differentiate (Reichenbach, 1947). Additionally, the type of ordering is not only limited to before and after. Events can have varying length and a hierarchical structure and as such can interact in ways such as having an event initiate another or having an event be included in another (Saurí, 2006). All these reasons can make determining the event ordering significantly difficult for human and even more so for computers.

There have been many different approaches to this problem, one of which came from research by Chambers and Jurafsky (2008). This work consists of identifying partially ordered event chains relating to a chosen protagonist, called narrative event chains. To identify this chain ordering a temporal classifier is trained on the timebank corpus (James et al., 2006), a set of manually annotated text indicating events and temporal order between them. Focus is placed on identifying *before* relations, allowing them to significantly reduce the relation space to two classes (BEFORE and OTHER). The number of relations are also expanded through applying transitive rules as the Timebank corpus omits any transitive relations. A *support vector machine* (SVM) is used as the classifier and is fed a range of features including individual features such as tense, grammatical aspect and aspectual class, and event-event pair featues such as syntactic dominance relations, order in text and bigram individual features (e.g., *"past present"* if one event had past tense and the other had present tense. In terms of determining whether an event was before the other, an accuracy of 72.1 per cent was achieved over the Timebank corpus using this approach. Earlier work by Chambers et al. (2007) considered six possible relations (BEFORE, IMMEDIATELY-BEFORE, INCLUDED-BY, SIMULTANEOUS, BEGINS and ENDS) and was trained using a Naïve Bayes model. This earlier approach over a larger relation set achieved a 59.4 per cent classification accuracy over the Timebank Corpus.

Work by Mirroshandel and Ghassem-Sami (2012) further investigates identifying temporal ordering of event through machine learning approaches. The Timebank specification is also used where the possible relations were condensed into 6 (BEFORE, IMMEDIATELY-BEFORE, INCLUDES, SIMULTANEOUS, BEGINS and ENDS) and an SVM model is used. Bootstrapped Cross-Document Classification (BCDC) is used as a training method for the SVM where the model is first trained on a standard corpus and then is retrained for each test document based on some related information. The motivation behind BCDC is that the type of language and words used is often reliant on the domain and topic of the initial text. Thus, training a general model and then retraining on related documents, of which the languages and words used is expected to be more similar, is expected to have improved performance on related test documents over a general model. The features used

are similar to those of Chambers and Jurafsky (2008), individual event features such as tense and grammatical aspect and event-event features such as distance between the event words and syntactical dominance. This approach was tested over the Timebank corpus (James et al., 2006) and Opinion corpus (Verhagen & Moszkowicz, 2008), a similar library of documents annotated to Timebank specification. The use of BCDC saw a 66.18 per cent classification precision on the Timebank corpus (compared to the 59.83 per cent without BCDC) and 68.07 per cent classification precision on the Opinion corpus (compared to 66.55 per cent without BCDC).

Other work in event ordering within clinical documents (Jung et.al, 2011) takes a different approach. This research's focus is to be able to build timelines of patient's medical history based on narrative clinical records. After the information is extracted, a rule-based method is used to determine their ordering. The rules that are used are based on deep semantic understanding of the text and involves identifying events and time phrases in text and determining the association between them. This association is determined through understanding the semantic relations that are mapped to the tokens of the text which they previously identify through a pattern-based extraction system. As these clinical documents are semi-structural with areas labelled, this rule based temporal ordering and pattern-extractor provides promising results, differing from the more common approach of applying machine learning techniques.

6.2 Corpus

All stages of determining ordering between events use multilayer perception networks which have been configured differently. These networks are trained on the Timebank 1.2 corpus (James et al., 2006) and Opinion corpus (Verhagen & Moszkowicz, 2008), both containing manually annotated news articles using TimeML specification (Saurí, 2006). These annotations indicate time and event phrases in the text and their relation between them along with gold standard information such as tense or class. Timebank consists of 183 articles and Opinion contains 73. The possible relations between event-event pairs and time-event pairs are BEFORE, AFTER, INCLUDES, IS_INCLUDED, DURING, SIMULTANEOUS, IAFTER, IBEFORE, IDENTITY, BEGINS, ENDS, BEGUN_BY, ENDED_BY, DURING_INV. Due to the previous event extraction identifying events based on only verbs, the ordering is only considered between events indicated by verbs, whereas the corpera also indicate some noun-based events.

The occurrences of no ordering occurring between object pairs is more prevalent than its existence in both corpera as seen in Table 11 and Table 12 below. The data is split into ordering between event-event pairs and time-event pairs as these are two different tasks we are approaching.

Corpus	Type	Count	%
Timebank	no ordering	673558	99.13
	ordering exists	5917	0.87
Opinion	no ordering	606156	98.53
	ordering exists	9029	1.47

Table 11: Existence of ordering in corpus between event-event pairs

corpus	type	count	%
Timebank	no ordering	63276	97.62
	ordering exists	1542	2.38
Opinion	no ordering	50511	96.72
	ordering exists	1711	32.76

Table 12: Existence of ordering in corpus between time-event pairs

It is apparent that the existence of ordering is heavily skewed towards no ordering in both corpera for both pair types. There are several possible explanations that contribute to this, one being that the corpera do not label transitive ordering (i.e., ordering implied through transitive properties of other orderings), causing there to be less ordering data than actually exists. Another explanation is that the ordering between object pairs simply aren't labeled even though they exist, perhaps due to the manual temporal tagging task being large and tedious, or some orderings being seen as less significant than others. Regardless, due to the data being heavily skewed towards no ordering, identifying temporal ordering between object pairs is established in two stages: detection of the existence of ordering and then the detection of the type of ordering (if indicated by the first stage). This process is used for both the event-event pair ordering and time-event pair ordering. This condenses the first stage into a binary classification, reducing its difficult and thus improving its precision.

Two additional techniques are used to increase performance on detecting the existence of ordering, one of which is the *synthetic minority over-sampling technique* (SMOTE) (Chawla et al., 2002). SMOTE is an algorithm that synthetically produces new samples for the minority class through constructing similar feature sets with small random differences. The algorithm selects real minority feature sets and comparing it to the k-closest other minority features sets, generating differences in the feature sets based on the similarities. SMOTE was used with 10-neighbors to increase the data set by 1000 per cent. This is used for both the event-event pair ordering and time-event pair ordering, but the features that are synthesized are specific to the task, as discussed below in their respective section. SMOTE causes the data more the data to become less skewed, improving the detection of ordering.

Cost sensitive classification is another technique to deal with the skewed data (Sammut & Webb, 2010).This increases the error cost of false negatives (no ordering selected when it exists) over false positive (selection of ordering when it does not exist). This is used during training and the ordering exists cases are given a cost of *three* times the ordering does not exist cases. Without the this technique, the training is more likely to move towards detection of no ordering as there is a lot more data and thus would produce better error values. As a result, the classifier would choose "no-ordering" regardless of the features as this produced the optimal error rate from the training data, which would be pointless to use. This technique is used for both the event-event pair detection and time-event detection alongside SMOTE.

After the existence of ordering is determined, the type of ordering is selected. The ordering classes come from a range from TimeML specification where we join DURING, DURING_INV (occurs for the duration of), SIMULTANEOUS and IDENTITY into a single one, reducing the number of ordering classes from 14 to 11. These classes where joined as it was less important to differentiate between them for our task in terms of what they represent. The class space for the ordering type is thus {BEFORE, AFTER, INCLUDES, IS_INCLUDED, SIMULTANEOUS, IAFTER (immediately after), IBEFORE (immediately before), BEGINS, ENDS, BEGUN, ENDED}. Other research (Mini et al., 2006; Mirroshandel & Ghassem-Sani, 2012) reduces the relation state space by combining relations with their counterpart. For example, combining AFTER and BEFORE into one relation as they can be used to represent the same information. We, however, avoid this to ensure the classification is correct regardless of the ordering of feature pairs. For example, if a correct class is AFTER for event pair E1-E2, we want to be able to identify AFTER for event pair E1-E2 and BEFORE for event pair E2-E1, making the ordering correct regardless of the event pair

configuration. The reduced relation space would, however, not be able to detect one of these relations as one of them doesn't exist and thus the correct classification is dependent on the event pair configuration. The relation types both event-event pairs and time-event pairs in either corpus for can be seen in Table 13 and Table 14 below.

Corpus	Type	Count	%
Timebank	BEFORE	742	25.08
	AFTER	492	16.63
	INCLUDES	195	6.59
	IS_INCLUDED	90	3.04
	SIMULTANEOUS	1,318	44.54
	IAFTER	20	0.68
	IBEFORE	26	0.88
	BEGINS	25	0.84
	ENDS	4	0.14
	BEGUN_BY	9	0.30
	ENDED_BY	38	1.28
Opinion	BEFORE	1,111	24.61
	AFTER	2,380	52.72
	INCLUDES	159	3.52
	IS_INCLUDED	417	9.24
	SIMULTANEOUS	324	7.18
	IAFTER	19	0.42
	IBEFORE	45	1.00
	BEGINS	25	0.55
	ENDS	4	0.09
	BEGUN_BY	13	0.29
	ENDED_BY	17	0.38

Table 13: Ordering type occurrences in corpera for event-event pairs

Corpus	Type	Count	%
Timebank	BEFORE	69	4.47
	AFTER	46	2.98
	INCLUDES	40	2.59
	IS_INCLUDED	1,000	64.85
	SIMULTANEOUS	230	14.91
	IAFTER	0	0
	IBEFORE	0	0
	BEGINS	12	0.78
	ENDS	78	5.06
	BEGUN_BY	37	19.46
	ENDED_BY	38	2.40
Opinion	BEFORE	244	14.26
	AFTER	82	4.79
	INCLUDES	107	6.25
	IS_INCLUDED	1,236	72.24
	SIMULTANEOUS	2	0.12
	IAFTER	2	0.12
	IBEFORE	0	0
	BEGINS	6	0.35
	ENDS	1	0.06
	BEGUN_BY	25	1.4
	ENDED_BY	6	0.35

Table 14: Ordering type occurrences in corpera for time-event pairs

It is clear that some of the classes occur significantly more than others in either corpus. Despite this, the techniques used to improve minority class classification (SMOTE and cost sensitive classification) in the determining ordering part are avoided as no type of ordering is significantly more important than another for our task. Thus we want our classification to best represent real data where those common orderings are likely present. Being able to determine that ordering exists is, however, more important that determining that it doesn't, making it ideal to use the techniques for this task.

It is also seen that ordering in some direction is more prevalent than others. In event-event pairs, for example, in Timebank for event-evnt pairs BEFORE ordering relations is a lot more evident than AFTER relations even though both relations can be used to express the same thing. We thus choose to double the data by copying these ordering to its respective counterpart, but with reversed ordering in the features. For example, event-event pair *E1E2* with expressed relation of *AFTER* , indicating E1 is AFTER E2, would also produce data for event pair *E2E1* with relation of *BEFORE*, indicating E2 is BEFORE E1. In these situations both classifications represent the same information, allowing us to increase the data available to essentially twice its size. This ensures that there is equal data for each relation direction and classification will be correct regardless of which pair order (E1-E2 vs. E2-E1) is being classified. This is only done with event-event pairs as time-event pairs can only be in the direction of time to event.

Additional features are provided in the corpera such as tense or class as ground truths. However, these values are not used in either classification task and only features determined by internal processes are used. This ensures that the feature sets of the training data is similar to feature sets beyond the training corpera as we are unlikely to have the ground truths in these situations and would have to rely on processes in our system to provide them.

The classification models for either stage are trained supervised on 70 per cent of the corpera and tested on the remaining 30 per cent.

6.3 Event-Event Pair Ordering

Both classifiers for determining Event-Event pair ordering are multilayer perception networks that use the same features, but differ to the classes they predict. The outcome of the first network is a binary class to indicate whether ordering exists or is absent and the outcome of the neural network is one of twelve classes to indicated ordering type.

6.3.1 Features

A number of features are used, some being individual features and some being joint. For each pair *E1-E2* the features list is expressed as individual features for *E1*, individual features for *E2*, and then the joint features. The features which are classed are split into a sequence of binary features, e.g. if feature is the third class of four classes then it is classified into a sequence of *0 0 1 0*. The individual features are *tense*, *grammatical aspect*, *event type*, *modality*, *perspective of main verb POS* (first-person, third person, etc.) and *direct negation* (has a direct *negation* relation in the dependency tree). All of these features (other than the verb perspective and direct negation) were previously identified in the event extraction part of the pipeline. A binary feature is also used to indicate whether the event verb has an preposition word (*after, before,...* as a modifier, indicating that it has an some type ordering in the text. This uses Wordnet, a lexical data base containing English words split into synonym sets (synsets) (Miller, 1995), and JWI 2.4.0, the Java interface (Finlayson, 2014), to get the synsets of the most common preposition words/phrases which are then matched to the modifiers. The last individual features are ConText (Harkema et al., 2009) evaluations for the actor, event and subject words in the text. ConText is based on the NegEx negation algorithm and evaluates words which represent events and returns classes

representing whether it occurred (*affirmed* or *negated*), who experienced it (*patient* or *other*) and when it occurred (*recent*, *historical* or *not particular*). This is performed on all parts of the extracted triple, even if it is not an event word, to gain any additional information that would be helpful in classification.

The joint features include the event distance between the extraction event words (counting all other extractions between the two in the pair and then normalize over the maximum), a binary value for whether event E1 is a modifier or an ancestor in the typed dependency graph to E2, a binary value whether E2 is a modifier or an ancestor to E1 and another binary value to indicated whether E1 and E2 has common ancestors in the dependency graph. The next four features is a binary feature to tell whether there is similarities between the actors and objects of the extractions. This again uses the Wordnet synset similarities to determine whether two entities might refer to the same thing, possibly linking the two events. These four features are each possible combination of comparisons between E1 entities and E2 entities (actor or object). Another feature is the Wu & Palmer measure for event Wordnet synsets (Wu & Palmer, 1994). The Wu & Palmer measure is a normalized value between zero and one which determines similarity by looking at the synset depth of the two words/phrases.

There are eleven features that (similar to individual features) look at words in the sentence and matches them to common preposition words/phrases using Wordnet. A rule based method is used to determine whether the preposition might refer to a relation between the event pairs in the text. This looks at the position of the preposition with respect to the event positions and then matches it to the relation type it is to likely indicate. For example, for event pair *E1E2* a sentence of "*Before E1, E2*", then it is likely that E2 occurred before E1 as there is a *"before"* preposition before both events with no other events between the entities. A binary feature is then used for each relation type to indicate whether the rule based method finds a relation for the two events of that type. The relation types are the same as the ordering classes (except the *DURING_INV* class) and while it identifies basic ordering like the example before, sentences can be very complex and this method can have difficulty with this on its own, thus it is used to supplement classification features rather than as its own method. One additional feature is used to indicate whether the events occur in the same sentence and another to indicate whether E1 is before E2 in the text. Due to the parse trees being sentence based, if the events were not on the same sentence some joint features would not be applicable and are instead labeled as false (zero).

6.3.2 Ordering Exists

Detection on whether ordering exists between event pairs is performed on a multilayer perception network. The network is configured to have 136 neurons (equal to feature count) in a single hidden layer, using an entropy error function and sigmoid activation.

As the data is heavily skewed, SMOTE and cost sensitive training is used, as previously mentioned. Using these techniques, the networks are trained on 70 per cent (Included SMOTE data) of the corpus data and tested on the remaining 30 per cent (excluding SMOTE data). As two corpera are used, the classification is tested in three situations, training on 70 per cent of the Timebank corpus and tested on the remaining 30 per cent and 100 per cent of the Opinion corpus (Table 15), training on 70 per cent of the Opinion corpus and tested on the remaining 30 per cent and 100 per cent of the Timebank corpus (Table 16), and training on 70 per cent of both and testing on the remaining 30 per cent of both (Table 17). These results are shown in the tables below,

Corpus	Type	Correct	Incorrect	Precision
TimeBank	no ordering	185,559	16,470	91.85
	ordering exists	1,574	239	86.82
Opinion	no ordering	571,689	34,467	94.31
	ordering exists	7,447	1,581	82.49
Total	no ordering	757,248	50,938	93.70
	ordering exists	9,021	1,820	83.21

Table 15: Ordering exists evaluation from training on Timebank corpus

Corpus	Type	Correct	Incorrect	Precision
TimeBank	no ordering	629,116	44,442	93.40
	ordering exists	5,148	768	87.02
Opinion	no ordering	177,599	4286	97.64
	ordering exists	1,991	676	74.65
total	no ordering	806,715	48,728	94.30
	ordering exists	7,139	1,444	83.18

Table 16: Ordering exists evaluation from training on Opinion corpus

Corpus	Type	Correct	Incorrect	Precision
Timebank	no ordering	180,996	21,003	89.59
	ordering exists	1,527	316	82.85
Opinion	no ordering	168,507	13,537	92.56
	ordering exists	2,158	353	85.94
total	no ordering	349503	34530	91.01
	ordering exists	3685	669	84.63

Table 17: Ordering exists evaluation from training on joined corpus

A two-tailed t-test (Haslam & McGarty, 2003) was used between the total accuracy score of each class type for each training method and with a critical value of 0.05 with results suggesting that all the scores where significantly different from one another. The only exception is the comparison between ordering exists on the Timebank training and ordering exists on Opinion training, suggesting there was no significant different between these two scores. As we place more significance on the "ordering exists" class and due to the small difference in accuracy between training, the chosen model is the one that is trained on the joined corpus. This is expected to be more general as it performs well on either corpus, whereas the other models are more specialized, performing better on one corpus than the other.

For comparison, a baseline system without the use of SMOTE or cost sensitive classification would simply select the majority class 100 per cent of the time. Thus no ordering would always be chosen over ordering exists. While this would produce the lowest error due to occurrence of classes in the data, the system would be useless and being able to detect the existence of ordering is more important for our task.

6.3.3 Type of Ordering

If ordering is determined through the previous step, then the type of ordering is classified. The same features are used with instead the outcome classification being the relation class between both events. Ordering is determined through a multilayer perception network configured similarly to the previous network. This network has a single hidden layer consisting

of the number of neurons equal to the number of input features. An entropy error function is used with sigmoid activation function on the hidden layer. This network was then trained similarly to the previous section: on Timebank (Table 18), Opinion (Table 19) and then a combination (Table 20).The output of the network is a value for each of the possible relation classes, and the largest value is selected as the relation type.

Corpus	Correct	Incorrect	Precision
TimeBank	1,305	470	73.52
Opinion	3,185	5,843	35.28
Total	4,490	6,313	41.56

Table 18: Ordering type evaluation from training on Timebank corpus

Corpus	Correct	Incorrect	Precision
TimeBank	2,020	3,897	34.14
Opinion	2,295	413	84.75
Total	4,315	4,310	50.03

Table 19: Ordering type evaluation from training on Opinion corpus

Corpus	Correct	Incorrect	Precision
TimeBank	1,290	485	72.68
Opinion	2,110	598	77.92
Total	3,400	1,083	75.84

Table 20: Ordering type evaluation from training on joint corpus

Similar to the ordering type, a two-tailed t-test (Haslam & McGarty, 2003) is used to detect if there is significant difference between the total accuracies for each training arrangement. Using a critical value of 0.05, the results of each table was found to be significantly difference, indicating that the training corpus significantly impacts the testing results. It is evident that the system performs a lot better on the corpus it is trained on over the other. Thus, if supervised data was available for the domain of documents we wish to evaluate, classification is expected to be improved when trained on this data as opposed to a general model. However, at this stage we are not considering any specific domain of application and choose the network that is joined on both corpera as the classification is more general rather than specified to a specific corpus.

A technique called network chaining (Zaamout & Zhang 2012), where the networks are split up to detect groups of similarly occurring groups to allow improved detection of less occurring ordering (similar to how ordering existence was split from ordering type), was also tested but resulted in similar scores to those above and was thus not used.

A baseline version of this classifier would be one which always selects the highest occurring relation from the joint corpera (AFTER). This classifier would produce the results seen in the Table 21 below,

Corpus	Correct	Incorrect	Precision
TimeBank	492	2,467	16.63
Opinion	2,380	2,134	44.54
Total	2,872	4,601	38.43

Table 21: Ordering type evaluation using baseline classifier

A direct comparison between these results and other systems is not possible due to the feature spaces and data being manipulated differently for to different research motivations. However, as a point of reference an imperfect comparison is presented in Table 22. It is also assumed that these measures do not include the "No ordering" relation between events, as the research does not indicate otherwise. Also note that Mirza (2014) trained on both corpera, but does not provide an individual test score for either, thus it is assumed classification score is the same for both.

System	Timbank Precision	Opinion Precision
Baseline (Most common selected)	16.63	44.54
Mirza (2014) (SVM based)	58.80	58.80
Chambers et al. (2007) (SVM based)	59.43	65.48
Mirroshandel & Ghassem-Sani (2012) BCDC (SVM based)	66.18	68.07
Our Approach	72.68	77.92

Table 22: Ordering type evaluation comparisons

Again, a two-tailed t-test was used to determine whether our system is significantly different from the other systems. Using a critical value of 0.05, our system was found to be significantly different from the other approaches for either corpus.

From these results it is evident that our system performs better than the compared systems. While this is true, each system likely occurs in different environments with different motivations, and thus the results may not be the same for the same task. For example, Chambers et al. (2007) classified the relations over a reduce class space, whereas our system retained the majority of classes. It is also possible that these systems were trained on less data and tested on more than our system. Thus while our system improves on the classification of those we compare to, it should be taken with a grain of salt.

6.4 Time-Event Pair Ordering

In addition to event-event pairs, relations are also extraction between time-event pairs. Time entities are previously identified in the lexical processing part of the pipeline through Stanford's SUTime library (Chang & Manning, 2012) and are used to only provide additional temporality to extracted events. Evaluation of time-event pairs uses outputs and features similarly to event-event pairs and is split into two stages: identifying whether ordering exists and then identifying the type of ordering. The ordering exists stage establishes a binary class representing whether ordering exists between the time-event pair and if it does, then the type of ordering establishes the exact relation out of the 11 types discussed in event-event pairs. This was split into two parts similarly due to sparsity of data for ordering exists compared to no ordering. The purpose of establishing ordering between time-event pairs is to use them to expand the mapping between event-event pairs as we are only concerned with this type of ordering to establish postconditions and preconditions. This is trained on the same corpera as the event-event pairs as both Timebank and Opinion provide labeling of time phrases and ordering between them and events.

6.4.1 Features

The features used between time-event pairs is similar to event-event pairs except the time entity has less individual features due to not having associated semantic aspects. Additionally, due to the event and times being different objects, features can only be expressed in one way, whereas event-event pair ordering features can be expressed in two different ways. The features used involve individual features for the event, individual features for the time phrase

and then joint features between the two. The individual features for the events are the same as in the event-event ordering, that is *tense*, *grammatical aspect*, *event type*, *modality*, *perspective of main verb POS* (first-person, third person, etc.), *direct negation* (has *negation* relation in dependency structure) ,*associated preposition word*, and ConText results. The individual time features used include *type* (DATE, TIME, DURATION, or OTHER) and ones which are identified similarly to events including *direct negation*, *associated preposition word*, and ConText output for the time phrase.

The joint features are similar to the event-event joint features, but are changed to match the time object. These joint features include normalized event difference between phrases in the text, three binary values indicating whether the event is a dependency parent of the time phrase, whether the time phrase is a dependency parent of the event phrase and whether they have a common ancestor in the dependency tree. Other joint features include whether the event occurs before the time phrase, whether the time phrase occurs before the event phrase, whether the time phrase is a modifier to the event's actor, a modifier to the main event phrase and a modifier to the event's subject.

All of these features are extracted through processes in the system based on the initial text. The corpera may provide some of the ground truths for these values, but these are not used.

6.4.2 Ordering Exists

The networks for detecting the existence of ordering between time-event pairs is configured the same as event-event pair networks. A multilayer perceptron network with a single layer consisting of 105 neurons (equal to number of input features) using a sigmoid activation function and an entropy error function. Due to the sparsity of ordering existing data we again use SMOTE and cost sensitive training, as previously mentioned. The networks are also trained and tested in the same data configurations as the previous section (except with the differing features) as shown in Table 23, Table 24 and Table 25.

Corpus	Type	Correct	Incorrect	Accuracy
TimeBank	no ordering	26,063	1,495	94.58
	ordering exists	648	57	91.91
Opinion	no ordering	48,861	1,650	96.73
	ordering exists	760	951	44.41
Total	no ordering	74,924	3,145	95.97
	ordering exists	1,408	1,008	58.28

Table 23: Ordering exists evaluation from training on Timebank corpus

Corpus	Type	Correct	Incorrect	Accuracy
TimeBank	no ordering	55,676	7,600	87.99
	ordering exists	1,131	411	73.35
Opinion	no ordering	13,868	1,276	91.57
	ordering exists	425	98	81.26
Total	no ordering	69,544	8,876	88.68
	ordering exists	1,556	509	75.35

Table 24: Ordering exists evaluation from training on Opinion corpus

Corpus	Type	Correct	Incorrect	Accuracy
Timebank	no ordering	25,351	2,223	91.94
	ordering exists	596	88	87.35
Opinion	no ordering	14,028	1,132	92.53
	ordering exists	377	130	74.36
Total	no ordering	39,379	3,355	92.15
	ordering exists	973	218	81.70

Table 25: Ordering exists evaluation from raining on joined corpus

Like the other networks, a two tailed t-test (Haslam & McGarty, 2003) is used to determine if there is any significant difference from the data it is trained on. With a critical value of 0.05, each training configuration result was found to be significantly different one another, indicating that the corpus used for training impacts the network's performance.

From these results the network training follows similar trends to the event-event pair networks in that they perform better on both corpera when trained on a joint corpus. Thus, the network selected is the last one that was trained on the joint corpus for the same reasons as the event-event pair network selection.

6.4.3 Type of Ordering

To detect the ordering type between time-event pairs a network with the same configuration was used. That is a network consisting of a single hidden layer with the number of neurons equal to the number of input features (105) and the use of an entropy error function with a sigmoid activation function. Similarly to the other networks, this was tested and trained over three different configurations of the training and testing data. Using the time-event pair features, these results can be seen in Table 26, Table 27, and Table 28. The networks are trained and tested in the same fashion as the previous.

Corpus	Correct	Incorrect	Precision
TimeBank	335	128	72.35
Opinion	1,027	684	60.02
Total	4,490	6,313	41.56

Table 26: Ordering type evaluation from training on Timebank corpus

Corpus	Correct	Incorrect	Precision
TimeBank	889	653	57.65
Opinion	372	141	72.51
Total	4,315	4,310	50.03

Table 27: Ordering type evaluation from training on Opinion corpus

Corpus	Correct	Incorrect	Precision
TimeBank	334	129	72.14
Opinion	337	136	73.49
Total	3,400	1,083	75.84

Table 28: Ordering type evaluation from training on joint corpus

A two-tailed t-test (Haslam & McGarty, 2003) was used to determine if the training configuration significantly affects the performance of the system for the total score. Using

a critical value of 0.05, the results of each training corpus configuration was found to be significantly different, indicating that the corpus used affects the performance of the system.

These results follow similar trends as the past networks where the network trained over the joint corpus performs better over both. We choose the network trained on the joint corpus.

6.5 Transitive Property Mapping

Once extracted events are evaluated to obtain ordering between event-event pairs and time-event pairs, these mappings can be expanded through transitive properties. For example, if event E1 is before E2 and E2 is before E3 we can conclude that E1 is before E3. This allows a larger mapping of orderings through logical rules. The mappings are done directionally to reduce the space of relations to eight (UNKNOWN, BEFORE, IBEFORE, SIMULTANEOUSLY, STARTED, ENDED, INCLUDES, NOTSIMULTANEOUSLY). This is done through an algorithm that creates directed acyclic graph (DAG) structures in two steps (Thulasiraman & Swamy, 1992) for the event ordering where nodes are events and the connection is the relation between them.

The first step is to create a list of all relation mappings. This is done by first taking the known event-event relations (identified through the temporal ordering networks) and the time-event relations and expand the event-event relations through one loop over the time-event mappings. For example, if event E1 is before time phrase T1 and E2 is after T2 and T1 is before T2, then a the ordering E1 before E2 is established. All event-event relations are then looped through, adding any transitive relations between two events based on relations they have with a common event. The transitive orderings are, however, not definite as two orderings can imply a relation out of a set of possible ones. For example, if E1 is before E2 and E2 ended E3, we can conclude that either E1 existed at the same time as E3 (SIMULTAENOUS) or was before E3. Due to not knowing the duration of E1 or E3 or the time between E1 and E2, we cannot pinpoint the exact ordering between E1 and E3. For these cases multiple possible relations are added to the ordering mapping between the two events. These mappings are then reduced or increased based on the relations of other event-event pairs, e.g. if another transitive states that a relation has to be one type then the possible relations between those events are reduced to that single one. The rules that define what transitive relation/s are created are manually defined. Each relation is looped through until a loop yields no additions, indicating a complete set of all possible transitive relations based on the original ones identified by the classification networks. If a relation is not found between two events then it is simply listed as UNKOWN.

The second stage of this algorithm then uses the complete set of mappings to create DAGs. Each relation mapping is iteratively added to a DAG. If there is multiple possible relations between two events, then multiple DAGs are created for each relation. If a relation is incompatible for the existing DAG (creates a cyclic ordering) then it is not implemented into the tree, meaning that the number of connections in each DAG is indicative of the number of relations accepted. The relations that have an absolute absolute (only one possible relation, e.g. the ones that are found through the networks) are enforced onto DAGs and ones that don't accept them are set aside from the main candidate DAGs (but still gain new relations) and are added back to the main list only if the list becomes empty. This ensures that a DAG always exists even if some implied transitive relations are invalid, reducing the consequence incorrect ordering classifications. Thus these DAGs indicate what transitive relations are accepted and also can be used to verify the correctness of the temporal ordering step by comparing the number of transitive relation to the largest DAG. This is, however, avoided in this thesis due to time constraints. The DAGs are then continuously expanded and branched until all transitive relations have been chosen.

The DAG with the largest number of connections is then chosen as it is the one that has accepted the most orderings. If there are multiple DAGs with the same number of

connections, then the DAG chosen is the one which has the lowest number of stronger relations. For example, STARTED would be considered stronger than SIMULTANEOUS as it indicates more information about the relation of the two events. For a DAG to exist for each relation, this indicates that we are unsure whether the ordering is STARTED or SIMULTANOUS. Thus we choose the DAG with the SIMULTANEOUS ordering as it is weaker and has less implications if incorrect due to providing less information about the relationship between the two events.

6.6 Event Ordering Approaches Discussion

At this stage we now have all events ordering based on event-event pairs and time-event pairs and then expanded this through transitive properties. The results of determining the type of ordering is seen to behave well compared to others based on the tested corpus. There are however some inherent limitations to such an approach.

The results of determining the type of ordering is reliant on the corpus that is tested. It can be seen in each of the classification tasks that it has improved performance on the corpus it is trained on over other. While this was combated by training on the join corpus and using techniques such as doubling the ordering data for each direction, classification is expected to be worse on corpus that are not similar to those that it is trained on. Timebank and Opinion is based on sets of manually annotated news articles, and while these articles follow similar structures, training on one and testing on another produces unsatisfactory results. A possible reason for this is that while the annotators followed the same specification (TimeML), they may have placed more importance on some relations over others or chose the ground truth relations differently. While the corpera both consist of only news articles, they also include different splits of ordering types, Timebank being biased towards SIMULTANEOUS and Opinion being biased towards AFTER, likely affecting the results. Thus the the classification accuracy on new documents is expected to be reliant on its similarity in structure and domain to that of the tested ones, with less similarities producing worse results. The improvement to this would be to train on documents from a larger range of domains or to only train on documents that are similar to the ones the system will be used on, making the classifier specialized to the task. We, however, were only able to acquire the two corpera used and aimed to have the classifiers to be more general, and thus did not employ either of these improvements.

Another limitation is the effect of incorrect classifications. If a chosen ordering is incorrect, it will likely cause inconsistencies within the ordering mapping that is created. While the consequences are alleviated through the use of multiple tree creations and the techniques used to improve classification, this concern will always be prevalent. This concern is not, however, only limited to that of our system as detecting ordering between events has been seen as a difficult task (Chambers et al., 2007; Mirroshandel & Ghassem-Sami, 2012) and thus some incorrect classifications can be expected. Simply improving the classification through investigating and employing additional techniques, such different network types, would help improve the error handling, but due to time constraints these classifiers were not taken any further.

7 Discussion

7.1 Lexical Ordering Summary and Limitations

Through this pipeline we have been able to identify and extract events and identify the orderings between these extractions. The first step of the pipeline, the lexical processing, identified the necessary information for the later parts of the pipeline, including part-o-speech tags and dependency annotations. This stage also formated the text through co-

reference resolution and extended mapping of verbs to make the text more consistent and structured. This step also identified time phrases and converted them to a comparable value, necessary for identifying event ordering with time pairs and being able to use them to identify ordering between events. This, however, heavily relies on Stanford CoreNLP (Manning et al., 2014) and assumes all annotations made are truthful. This presents a limitation to the systems as while Stanford CoreNLP is consider state of the art in terms of NLP tools, it can be incorrect on complex and unusual sentence structures and thus the parts of the pipeline that rely on these annotations would likely be incorrect to an extent. This, however, is be expected as a margin of error is always prevalent NLP annotation tasks. These annotations provide semantic information in a structured format which can be used in the later parts of the pipeline.

7.2 Event Extraction Summary and Limitations

After we format the text and annotate additional semantic information we use a rule-based method to extract events. The method uses several rules to identify extraction triples based on verbs and their relations with other entities. Additional extractions are made based on nouns to identify possessive relations and adjectival relations. These extractions are then annotated with modifiers based on the type dependency relations to allow the extractions to retain information lost in the extraction such as conditional clauses. This extraction was then evaluated using 30 randomly selected documents Hansard House of Representatives Start of Business Parliamentary Speeches (Australian House of Representatives, 2017) where the gold standard was the researcher's interpretation of the results. Our rule based method achieved a precision score of 73.40 per cent and a recall score 73.89 per cent, producing an F1-measure of 0.7369 per cent when only counting "correct" extractions as opposed to "correct" and "partial". This F1-measure was found to be comparable to existing methods if not slightly worst, presenting an effective general method of event extraction that avoids the need for training data, a key strength of rule-based approaches. Our evaluation is, however, limited as the gold standard data is based off of the researcher's interpretations of the results and thus the results are unlikely to be fully replicable by other researchers.

Our rule-based approach to event extraction deviates from the machine learning focus that most current work takes (Chiticariu et al., 2013) and while ours method provides a fast and effective way at identifying triple extractions without the need of training data, there are several inherent limitations to the approach. Being a rule-based method, many of these rules are defined manually by the researchers and thus it would be difficult and tedious to change or implement new rules if some aspects have been overlooked in the current system. A machine learning approach would, however, more easily implement these changes as you would just train it on new data that represents the change, rather than having to implement new rules or change existing ones and ensuring they all work together. Another limitation is that the system is unable to infer meaning beyond what is represented in the parse tree and thus important extractions may be missed. For a machine learning approach, this information could just be represented in the data and the system would be able to adapt. Additionally, important semantic information such as condition clauses (such as *"if"*) are only identified and annotated to the extraction in our system, but not actually used to alter the extractions beyond this, this is expected to be future work. Combining our system with a machine learning approach is therefore expected to improve the results, but the implementation of such was avoided due time constraints and the lack of annotated training data. A common practice is to use rule-based approaches to approximate features which are then fed into a machine learning model (Chiticariu et al., 2013), a possible avenue to improve our system.

We also identify additional extraction features, such as tense and and grammatical aspect, through rule based methods that look at the semantic structure of the sentence. Only the tense was evaluated due to time constraints, but this achieved 93.13 per cent accuracy on the Timebank 1.2 corpus (James et al., 2006) where the majority of incorrect classifica-

tions was where tense was ambiguous or required understanding of words to identify. This demonstrates how rule based methods can be effective in identifying minor event features indicated by semantic structures. These methods, however, have inherent limitations such as assuming the annotations made in the lexical processing are correct and using only these annotations (which are sentence based) whereas the aspect may require understanding word definitions or relations beyond the immediate sentence to determine.

7.3 Temporal Event Ordering Summary and Limitations

After the event extraction step of the pipeline we now have a list of events and properties in the form of triple extractions and need to establish ordering between them through the use of neural networks. This is necessary to identify preconditions and post condition relations between the extractions.We determine ordering through event-event pairs and time-event pairs where event objects are those that come from the event extraction step and time objects come from the lexical processing step. These are established in two stages, determining whether ordering exists and if it exists, then determining the type of order from a set of relations defined by TimeML specification (Saurí et al., 2006). Both steps are done as machine learning tasks using multilayer perceptron networks with sigmoid activation and entropy error functions on a single layer. The corpera used was TimeBank 1.2 (James et al., 2006) and Opinion corpus (Verhagen & Moszkowicz, 2008) which both consist of news articles annotated to TimeML specification. The use SMOTE algorithm (Chawla et al., 2002) to synthetically increase minority cases and cost-sensitive training (Sammut & Webb 2010) to improve minority classification greatly improved the detecting of the existences of ordering as otherwise the networks would over-fit to the more common class (no ordering) 100 per cent of the time. This produced 91.01 per cent accuracy on detecting the existence of no ordering and 84.61 per cent accuracy on detecting the existence of ordering for event-event pairs and 92.15 per cent accuracy for the existence of no ordering and 81.70 per cent accuracy for the existence of ordering for time-event pairs. Using the same network configurations (excluding SMOTE and cost sensitive training) for the type of ordering produced 75.84 per cent accuracy on event-event pairs and 75.4 per cent accuracy on time-event pairs.

This presented a hybrid method in the sense that many of the features used in the network training are found through rule-based methods whereas the models that are trained are based on machine learning methods. Comparing this to other work in determining event ordering type for event-event pairs showed that our method improved on those approaches which was primarily SVM based (Mirza, 2014; Chambers et al., 2007; Mirroshandel & Ghassem-sani, 2012). This highlights the effectiveness of a hybrid method with the features used and a simple neural network configuration.A transitive mapping algorithm is then used to expand the event orderings through transitive properties. While the results of these methods appear to be effective, there are several inherent limitations. One limitation is that the networks appear to perform better on documents similar to the ones they are trained on. This will restrict the performance on any new document analyzed which is outside of the training domain. An improvement would be to include a larger range of documents in training or to focus on a specific domain for the extraction to take place. Additionally, our evaluation measures for the networks are only based on the two corpses used, both consisting only of news articles, and thus it is unlikely for the performance to carry across to documents of another domain.

An additional limitation is that incorrect classifications by the network can have detrimental effects through the transitive mapping algorithm. While the impact are reduced through retaining the correct DAG mappings and using the most valid ones, if the incorrect ordering is valid with the others then it will be used. The main way around this would be to improve the performance of the methods or put more checks in place as it can be difficult to detect when the chosen ordering is incorrect despite working with all of the others.

7.4 Future Work

At the end of this pipeline we now have lists of extracted events and properties and orderings between them and while this doesn't match the overall goal of a scalable scalable narrative model built on machine learning processes, it provides the foundations for such work. This overall goal was found to be a very large tasks with many internal operations and contributing concepts. Thus being able to develop the entire model was found to be out of the scope of this project due to time constraints and we decided to focus on the initial steps in working towards this goal. From this point, work to be done includes using the orderings to determine preconditions and postconditions of extraction events, this could include determining which events refer to the same action and then counting the number of occurrences of ordering between them and other events over multiple cases to calculate the likelihood of being preconditions or postconditions.

Another method may be to train a machine learning model on sets of event orderings and their proposed pre/post-condition. Additional relations would then need to be identified based on the modifiers within event extraction words and determining whether they are part of another event. For example, in *"I will pay you money if you make me food"* two events would be extracted {I}*pay*{money} and {you}*make*{food}. The one of the modifiers on the *pay* extraction would be an *if clause* relation of *make* indicating that the *pay* extraction is dependent on *make* through an *if* statement. Thus this modifier would be then linked to the *make* extraction and a relation would then be identified through this. Doing this would require looking at each of the possible modifier relations, understanding what they imply and then expanding these to produce links between events.In addition to this, there is also all of the improvements to be made that have been discussed to help reduce the limitations of the system.

7.5 Significance

The narrative generation system we are working towards would have significant implications in a range of fields. Medical care is a hot spot for text summarization and generation (Workman et al., 2012; Mishra et al., 2014) and our system would be able an approach for extracting information from these documents and generating new text on it which could be used to aid diagnosis and training. The text generation could also be used in patient recovery (Hall & Powell, 2011) by taking text that represents their situation as input and providing textual output in the form of queries or stories to aid them. For example, outputs may be questions and visual stories tailored to the situation of the patient to aid their recovery.

Another application of the system is its uses of generating general text based on input text and past learned information. Thus could build up a large and wide knowledge base and be able to generate text and narratives from large ranges of topics. An application of this could be generating speeches or articles on specific topics which would then be edited and improved for publishing online or spoken in institutions. This would provide a fast and easy way to generate ideas and a starting point for these types of documents. Textual documents are something we use on a day to day basis and thus being able to generate comprehensive ones based on learned information would have a great impact.

References

Angeli, G., Premkumar, M. J., & Manning, C. D. (2015). Leveraging linguistic structure for open domain information extraction. *Proceedings of the Association of Computational Linguistics (ACL), 1*, 344-354.

Australian House of Representatives. (2017). *Start of business transcripts.* Canberra, ACT: Hansard.

Chambers, N., & Jurafsky, D. (2008). Unsupervised learning of narrative event chains. *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, 602-610.

Chambers, N., & Jurafsky, D. (2009). Unsupervised learning of narrative shemas and their participants. *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, 602–610.

Chambers, N., & Jurafsky, D. (2009). Unsupervised learning of narrative schemas and their participants. *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, 602-610.

Chambers, N., Wang, S., & Jurafsky, D. (2007). Classifying temporal relations between events. *Proceedings of ACL-07*, 173-176.

Chang, A. X., & Manning, C. D. (2012). SUTIME: A library for recognizing and normalizing time expressions. *8th International Conference on Language Resources and Evaluation (LREC 2012).*

Chawla, N.V., Bowyer, K.W., Hall, L.O., & Kegelmeyer, W.P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321-357.

Chen, D., & Manning, C. (2014). A fast and accurate dependency parser using neural networks. *Proceedings of EMNLP 2014.*

Chen, G., An, B., & Zeng, S. (2015). A rule-based information extraction system for human-readable semi-structured scientific documents. *Computer Science and Network Technology (ICCSNT), 2015 4th International Conference*, 75-84.

Cheong, Y. G., & Young, R. M. (2006). A computational model of narrative generation for suspense.*Proceedings of the 21st national conference on Artificial intelligence, 2*, 1906-1913.

Chiticariu, L., Li, Y., & Reiss, F. R.. (2013). Rule-based information extraction is dead! long live rule-based information extraction systems!. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 827–832,

Derczynski, L., & Gaizauskas, R. (2010). Usfd2: annotating temporal expresions and tlinks for tempeval-2. *Proceedings of the 5th International Workshop on Semantic Evaluation*, 337–340.

Doust, R., & Piwek, P. (2017). A model of supsense for narrative generation. *Proceedings of The 10th International Natural Language Generation conference*, 178–187.

Finkel, J.R., Grenager, T., & Manning, C. (2005). Incorporating non-local information into information extraction systems by Gibbs sampling. *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005), 1*, 363-370, doi:10.3115/1219840.1219885.

Gervás, P. (2009). computational approaches to storytelling and creativity, *AI Magazine, 30(3)*, 49-62.

Hall, J. M., & Powell, J. (2011). Understanding the person through narrative. *Nursing Research and Practice*. 10 pages, doi: 10.1155/2011/293837.

Harkema, H., Dowling, J.N., Thornblade, T., & Chapman, W.W. (2009) .ConText: an algorithm for determining negation, experiencer, and temporal status from clinical reports. *Journal of Biomedical Informatics, 42(5)*, 831-851.

Haslam, S.A., & McGarty, C. 2003. *Research Methods and Statistics in Psychology*. Sage Publications Ltd, London.

Hecht, B., Starosielski, N., & Dara-Abrams, D. (2007). Generating educational tourism narratives from Wikipedia. *Intelligent Narrative Technologies, Papers from the 2007 AAAI Fall Symposium*, 37-44.

Julie Porteous, Fred Charles & Marc Cavazza. (2013). NetworkING: using character relationships for interactive narrative generation. *The 2013 International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2013)*, 595-602.

Jung, H., Allen, J., Blaylock, N., & Swift, M. D. (2011). Building timelines from narrative clinical records: initial results based-on deep natural language understanding. *Proceedings of BioNLP 2011 Workshop*, 146-154.

Kiciman, E., & Richardson, M. (2015). Towards decision support and goal achievement: identifying action-outcome relationships from social media. *KDD '15 Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 547-556.

Klein, D., & Manning, C. D. (2002). Fast exact inference with a factored model for natural language parsing. *Proceedings of the 15th International Conference on Neural Information Processing Systems*, 3-10.

Klein, S, et al. 1973. Automatic Novel Writing: A Status Report. In *Technical Report 186, Computer Science Department, The University of Wisconsin, Madison.*

Lebowitz, M. (1983). Creating a story-telling universe. *Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 1*, 63-65.

Lee, H., Chang, A., Peirsman, Y., Chambers, N., Surdeanu, M., & Jurafsky, D. (2013). Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics, 39(4)*, 886-916.

Lee, H., Peirsman, Y., Chang, A., Chambers, A., Surdeanu, M., and Jurafsky, D. (2011). Stanford's multi-pass sieve coreference resolution system at the coNLL-2011 shared task. *Proceedings of the CoNLL-2011 Shared Task*, 28-34.

Mani, I., Verhagen, M., Wellner, B., Lee, C.M., & Pustejovsky, J. (2006). Machine learning of temporal relations. *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, 753-760.

Manning, C. D., Raghavan, P., & Sch utze, H. (2008). *Introduction to information retrieval*. Retrieved from https://nlp.stanford.edu/IR-book/.

Mausam, Schmitz, M., Bart, R., Sorderland, S., & Etzioni, O. (2012). open language learning for information extraction. *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 523-534.

Meehan, J. R. (1977). Tale-spin, an interactive program that writes stories. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, 91-98.

Miller, G.A. (1995). WordNet: A lexical database for english. *Communications of the ACM, 38(11)*, 39-41. Mirroshandel, S. A., & Ghassem-Sani, G. (2012). Towards unsupervised learning of temporal relations between events. *Journal of Artificial Intelligence Research, 45*, 125-163.

Mirza, P. (2014). Extracting temporal and causal relations between events. *Proceedings of the ACL 2014 Student Research Workshop*, 10-17.

Mishra, R., Bian, J., Fiszman, M., Weir, C. R., Jonnalagadda, S., Mostafa, J., & Fiol, G. D. (2014). Text summarization in the biomedical domain: A systematic review of recent research. *Journal of Biomedical Informatics*, 457–467, doi:10.1016/j.jbi.2014.06.009.

Mitchell, T., et al. (2015). Never-ending learner. *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Mykowiecka, A., Marciniak, M., & Kupść, A. (2009). Rule-based information extraction from patients' clinical data. *Journal of Biomedical Informatics, 42(5)*, 923-936.

Nathanael Chambers, Shan Wang, and Dan Jurafsky. 2007. Classifying temporal relations between events. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions, 1*, pages 173-176.

Pérez y Pérez's, R. (1999). *MEXICA: A Computer Model of Creativity in Writing* (Doctoral Dissertation).

Pham, L.V., & Pham, S.B. (2012), Information extraction for vietnames real estate advertisements. *Knowledge and Systems Engineering (KSE), 2012 Fourth International Conference*, 181-186, doi:10.1109/KSE.2012.27.

Raghunathan, K., Lee, H., Rangarajan, S., Chambers, N., Surdeanu, M., Jurafsky, D., & Manning, C. (2010). A multi-pass sieve for coreference resolution. *EMNLP '10 Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 492-501.

Recasens, M., Marneffe, M., & Potts, C. (2013). The life and death of discourse entities: identifying singleton mentions. *Proceedings of NAACL 2013*.

Reichenbach, H. (1947). *Elements of symbolic logic*. University of California Press, Berkeley.

Riedl, M. O. (2004). *Narrative planning: balancing plot and character*. (Doctoral dissertation). Retrieved from NCSU Libraries.

Riedl, M. O., & Young, R. M. (2003). Character-focused narrative generation for execution in virtual worlds. *Proceedings of the International Conference on Virtual Storytelling*, 47-56.

Sammut, C., & Webb, G. I. (2010). *Encyclopedia of machine learning*, doi:10.1007/978-0-387-30164-8.

Santos, C.N., & Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis of short texts. *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, 69–78.

Saurí, R., Littman, J., Knippen, B., Gaizauskas, R., Setzer, A., & Pustejovsky, J. (2006). TimeML annotation guildelines version 1.2.1.

Srihari, R., & Li, W. (2000). A question answering system supported by information extraction. *Proceedings of the sixth conference on Applied natural language processing*, 166-172.

Stubbs, A., & Pustejovsky, J. (2012). *Natural language Annotation for Machine Learning*(1). Sebastopol, CA: O'Reilly Media, Inc.

Thulasiraman, K., & Swamy, M.N.S. (1992). *Graphs: Theory and algorithms.* Concordia University, Montreal, CA: John Wiley & Sons, Inc.

Toutanova, K., Klein, D., Manning, C., & Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. *Proceedings of HLT-NAACL 2003*, 252-259.

Toutanova, K., Klein, D., Manning, C., & Yoram Singer. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network.*NAACL '03 Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, 1*, 252-259. doi:10.3115/1073445.1073478.

Turner, S. R. (1993). *Minstrel: A Computer model of Creativity and Storytelling.* (Doctoral dissertation). Retrieved from ACM Digital Library. (UMI GAX93-19933).

Verhagen, M., & Moszkowicz, J. (2008). *AQUAINT TimeML 1.0 corpus documentation.* Brandeis University.

Workman, T. E., Fiszman, M., & Hurdle, J. F. (2012). Text summarization as a decision support aid. *BMC Medical Informatics and Decision Making, 12(41)*, 12 pages.

Wu, F., & Weld, D.S. (2010), Open Information Extraction Using Wikipedia. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 118-127.

Wu, Z., & Palmer, M. (1994). Verbs semantics and lexical selection. *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, 133-138.

Yates, A., Cafarella, M., Banko ,M., Etzioni O., Broadhead, M., & Soderland, S. (2007). TextRunner: open information extraction on the web.*Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations.* 25-26.

Young, M. R., Ware, S., Cassell, B., & Robertson, J. (2013). Plans and planning in narrative generation: A review of plan-based approaches to the generation of story, aiscourse and interactivity in narratives. *SDV: Sprache und Datenverarbeitung: Intenational Journal of Language Processing, 37*, 41-64.

Zaamout, K., & Zhang, J.Z. (2012). Improving neural networks classification through chaining. *Proceedings of the 22nd international conference on Artificial Neural Networks and Machine Learning, 2*, 288-295.