Extracting Ordered Action Sequences from Natural Language Texts

The Australian National University

Jul 2020

Pandu Kerr

U6083582

Abstract

Conceptions of machine narrative generation often uses artificial intelligence planning domain models to represent and generate narratives. These domain models take human effort to create, which hinders the portability of the narrative generating applications. This project demonstrates a system to extract ordered action sequences from natural language text. Natural language processing technologies such temporal annotation and semantic role labelling are used to extract the ordered sequence. An object lexicon is also presented. This is a collection of objects and their types to be used in the planning domain. This is the first step in generating a planning model to create further narratives.

Contents

1. Introduction

Artificial intelligence planning naturally lends itself to narrative generation (Young, et al., 2013). Within artificial intelligence planning, the planning domain model is a model of a problem world which is defined by state and state-transitions. Planning domain models have shown to be an appropriate way to model the domain of a fictional or non-fictional world (Porteous, et al., 2013), (Riedl & Young, 2010). By solving a plan within the planning domain, a new narrative is effectively generated. A plethora of different narratives can be potentially be generated from the same planning domain. To avoid the high-demand task of generating handcrafted planning domains, researchers propose methods to automate this process. This work is still in its infancy, with input texts are restricted to a prespecified form, such plot synopses (Hayton, et al., 2020). To look at the more general goal of generating planning domains, a common theme is that they require a sequence of actions. In section 2.1, the planning problem is described in greater detail.

The goal for this project is to generate action sequences: a series of action observations. These actions can be used to as a basis for a planning problem. Action sequences may be served as input to algorithms such as LOCM, which infer a planning problem from an ordered sequence of actions (Cresswell, et al., 2013). The opportunity for this is sensed in a similarity between grammatical predicates and PDDL actions. If this is achieved, the planning domain model can be said to be a logical representation of the text's world. This can expand the range of domains from which to generate narratives outside of the user's imagination, given they have access to these texts.

Besides action sequences, an objection lexicon is proposed and prototyped. This aims to capture some characteristics of objects acted upon in the text's world. This may aid the planning domain acquisition process.

This project uses natural language processing tools to solve two major components: temporal annotations and relation extraction. The effectiveness of these tools in aiding the task is examined in the results section.

2. Background, Literature and Context

This project is placed within the context of narrative generation. An attempt at this task can be found in William James' 2017 honours thesis (James, 2017). His thesis served as an invaluable reference during initial work on this project. A system where a planning model is learnt from natural language have shown to be successful in the field of learning planning domains for toy artificial intelligence (AI) problems, such as Towers of Hanoi (Lindsay, et al., 2017). Generating planning models use LOCM, or an extension such as LOCM2 to transform sequences of actions into PDDL domains. These solutions are restricted to limited domains and vocabularies. The aim here is take advantage of human languages' grammatical and ontological structures to extract sequences domain-independently.

(Hayton, et al., 2020) Presents a system whereby a planning domain model basis is generated from narrative synopses. Techniques applied include pronoun resolution and name disambiguation. A limitation of this work is that it is restricted to plot synopses.

There are three main areas this research is based on. That is: automated planning, narrative generation and extracting plans from text. Background on these areas is given in the proceeding sections 2.1, 2.2 and 2.3 respectively.

Background on techniques used in the implementing the action-extraction system are given in section 4-7.

2.1. Background: Artificial Intelligence Planning

Artificial intelligence (AI) planning, sometimes called automated planning, is a field of research where the aim is to obtain a sequence of actions so that one or more agents (perhaps a robot or computer program) can achieve a goal.

AI planning has evolved since its inception to feature domain independent languages, such as PDDL (Planning Domain Definition Language). As the name "domain independent"

suggests, these are languages which can be used to model planning problems outside of a specific domain. The advantage is that there are planners written to solve problems specified by the language. Within many forms of planning, there is the concept of states and transitions. The state refers to properties of the world at a single point in execution, and transitions are the allowed changes between states.

To describe classical planning, definitions are drawn from Ghallab, et al., 2016 (Ghallab, et al., 2016) to maintain consistency. In classical planning, to accommodate large state spaces, states are conceptualised as state variables which take objects as arguments. These are also referred to as predicates. An argument of a state variable has a range which defines what object it may be. State variables are assigned to a symbol. To clarify with an example:

```
facing (Julian) = West
```

`facing` is the name of the state variable and `Julian` is an object. `Julian` is the first and only argument of `facing`. The range of the first argument of `facing` may be defined as objects that can change direction. The range of objects an argument can take may be called the argument's type in some formulations. State variables are assigned a value. Altering a state variable's value indicates a change in the world state. Above, `facing (Julian)` is assigned the value `West`. The range for this state variable's value may be of restricted to the range of compass directions. In the classical planning formulation, a state can be described as a series of state variables, where all the state variables' arguments and values have been assigned.

Transitions, in some planning languages, are defined as an "action" or "operations" which changes the world's state. In this project they are referred to as "actions". A state-transition function maps an (action, state) and to a new state. This is a partial function: some actions may not be applicable for some states. An action consists of a head, preconditions and effects. Effects are sometimes called postconditions.

An action's head consists of its name and a series of parameters. The parameters correspond to those use in preconditions and effects. For example:

lift (w, b)

Here, the first and second parameters are objects in the range of worker and box respectively. The preconditions check that whether state variables, with action parameters as arguments, have some assignment. For example:

pre: holding(w)=Nil, on-ground (b), task(w)= b

If these evaluate true, then state variables, with parameters as arguments, are assigned values according to the action's effects:

effects: holding(w) <- b, on-ground (b) <- False

Actions can have associated costs in some formulations. A series of actions is called a *plan*. The formulation of an action as above is called an *action template*. The occurrence of an action in a plan is called an *action instance*, following work in the action-template field (Mourao, et al., 2012). Notation-wise, an action instance appears like an action head, where parameters have been replaced by objects.

A parameter of action templates can be written in the form `?parameter` to differentiate it from an argument of an action instance. This notation appears in PDDL (Fox & Long, 2003).

A planning domain is defined as set of possible world states, a set of possible actions and aforementioned state-transition function. A planning problem consists of a planning domain and a start and goal states. A solver transforms the world from the initial state to the goal state through a series of actions. For an action to occur, the state of the world must satisfy its *preconditions*. The resulting change in state of the world is determined by an action's *effects*. To cope with different types of planning meta-problems, such as actions with durations and non-deterministic effects, the classical planning model may be extended and changed to accommodate it.

2.2. Background: narratives as plans

Narrative generation can be defined a "the automated creation of meaningful event sequences" (Riedl & Young, 2010). One of the attributes identified by Riedl & Young (2010) as determining the understandability of a narrative *is logical causal plot progression*. Logical causal plot progression means that a narrative's events obey the rules of the narrative's world. This attribute is well suited for domain independent planners to capture. Using a planner as a tool for narrative generation means that a narrative is encoded as plan, where events are rendered as action instances. Different narrative-plans can be generated by changing the initial and goals states. To developing a narrative planning model is to develop a domain of discourse, which may be termed a "microworld" (Young, et al., 2013). The attraction is that within in this microworld, different narratives may be quickly spun, based on different goal and initial states.

As the narrative-plan is generated by the planner, the resulting plan is subject to the idiosyncrasies of the planner. Planners can be tailored increase the richness of a narrative, such as to model narrative conflict (Ware & Young, 2011).

The relationships between characters can be encoded into a planning model as state variables. Actions involving characters are preconditioned on these relationships (Porteous, et al., 2013). For example: `romantic-interest (?character1 , ?character2)` and `upset(?character1)` can be a precondition for `bad-date (?character1 , ?character2)`. The role of a character can be encoded as an object type, such as `waiter`.

Although (Ware & Young, 2011) makes a distinction between an action induced by character and a narrative event that occurs due to the environment. This distinction can act as a signal for the planning mechanism, especially when a focus is made on characters' intentions. This won't be made here, as this project aims to generate planning actions for a classical planner. For the purpose of generating narratives used in a classical planner, this distinction is already implicated by the action name and the type of its parameters. An action simply represents a transition in the state of modelled world. It does not necessarily mean that there was an intelligent agent which motivated the action. For example `rain ( ?Location, ?Started)` could be an action, despite the lack of obvious agent.

2.3. Background: extracting plans from text

Intuitively, one can identify that grammatical constructs can act as a basis for building logical structures. Take the often-taught subject-verb-object (SVO) breakdown of sentences. Most sentences fit this structure outright, or an extended form (University of Montana Writing Center, n.d.). Take the sentence:

> "Adam drank some juice."

A simple SVO parse would be: {subject: Adam, verb: drank, object: some juice}. In dependency grammars, often used in natural language part-of-speech taggers (Toutanova, et al., 2003), a verb forms the root of a sentence and clause. It is important to note that SVO is an extraction on the syntactic level and may not convey the ideal level of semantic meaning.

It is possible to structure a clause as a binary relation with the verb as the name and with object and subject as arguments. There is an implication this relation represents change, described by the verb, applied to its arguments. This appears similar to a planning action:

```
drank (Adam, some juice)
```

In this relation, the name is the verb *drank*, and its arguments "Adam*"* and "some juice" (which are noun phrases) are enclosed in brackets. It is important to note that some relations will be in the form of a planning state variable rather than an action, such as:

```
at (Jenna, shops)
```

In these relations, arguments are often noun phrases. Relations can appear close to the form of actions in planning models. The relation name, (a verbs) correspond to actions, and arguments (which are nouns phrases) correspond to objects.

The fact that the arguments of these grammatical relations are ordered is an important fact in generating the domain model. In an action template, the parameter position implies its type (if typing is supported in the planning language used). In human language, argument position implies semantic meaning (Smith, 2004). For example, the second argument of "at" is probably a location, and the first argument is probably a thing that can move.

A recent paper presents a system where the basis for a narrative planning model is extracted from plot synopses (Hayton, et al., 2020). The basis for this system is to extract planning actions and objects. A dependency parser identifies dependencies between words. A part-of-speech tagger identifies the grammatical role of the word (such as verb or noun). Objects which are extracted are words tagged as nouns or adjectives and have dependencies which imply the word is a subject or object.

Most attempts at inferring planning domains do so by examining sequences of action instances, such as plan that has already been executed, or texts which appears similar. Planning domain extractions vary in how much of the plan they extract. For example, some aim to extract action templates while others aim to also extract state variables (Branavan, et al., 2012).

(Cresswell, et al., 2013) presents LOCM, a system that infers a planning domain model based on a series of actions (for example a fully, or partially executed plan). LOCM uses a set of assumptions to generate finite state machines (FSM). Finite state machines are used to model state transitions of types of objects in the world. These finite state machines are the basis for generating action templates. An action template is built by examining the FSM corresponding to each of its parameter's type. If an object type is the parameter for an action, then that object type's FSM will have an edge corresponding to an action. Precondition and effects correspond to out-nodes and in-nodes associated with the action's edge.

LOCM relies on a set of assumptions which are generic enough to describe most planning domains. This method's strength is that, besides the series of actions, no other background information is needed. In limited capacities, it is possible to extract action sequences appropriate for use with LOCM (Lindsay, et al., 2017). In such a case, sentences used for extraction are intended simple and are intended as instructions, such as "move parcel1 from location B into the red truck". The major limitation is that LOCM uses flawless plans as input. It cannot cope with an object's state changing due to a hidden action, for example. This drawback means that it may not do well on actions extracted from newswire or fiction.

(Branavan, et al., 2012) Uses single argument predicates extracted from the text as states, such as `have(wood)=True`. These states are used as sub-goals in a hierarchical learning planner. Preconditions for sub-goals are estimated using natural language processing mechanisms. These are tested and updated in a reinforcement learning process.

A subproblem in planning domain extraction is action extraction. Extracting an action occurs when an action template is realised. Work in this field often involves observing sequences of action instances. These sequences may be a full or partial plan within the desired planning domain.

Research often solves sub problems with action extraction. An exact solution for learning actions' effects from action sequences is given by Amir, (2005). It has been demonstrated that action templates may be extracted from noisy state observations, given complete action observations (Mourao, et al., 2012). This has been extended by (Zhuo & Kambhampati, 2013) whose AMAN algorithm can cope with noisy action observations.

In the extraction of preconditions and effects, (Sil, et al., 2010) uses mining grammatical patterns from a large corpus of web texts. Preconditions are also found using a natural language processing (NLP) pipeline consisting of a semantic role labeller (SRL) and coreference resolution. The precondition and postcondition extractions are concerned with the general state of the world, rather than the states of objects within the world. For example, after the action "cut" occurs, the effect "blood" occurs.

A common theme of inferring action templates, and the larger goal of extracting planning domains, is the need for an action sequence. This must first be extraction from natural language text. Background of this subproblem is discussed in the next section.

2.4. Problem area: Extracting action sequences

Extracting action sequences is the problem of extracting a sequence of actions instances from natural language text. Methods for extracting these representations include examining a sentence's grammatical dependency tree (James, 2017), (Hayton, et al., 2020).

Another method is to predict representations through deep reinforcement learning (Feng, et al., 2018). Deep learning methods are effective but are expensive to train on a domain, in terms of time and labelled training data. Extracting action sequences is related to the language grounding field in robotics, which is concerned with "learning the meaning of language as it applies to the physical world" (Matuszek, 2018). Language grounding techniques often make use of deep neural networks and often provide extractions in a form particular to the problem (Mei, et al., 2016).

The purpose of this work is to investigate capturing a planning domain model so that it reflects the actions and logic of input text, without training data. The only guarantee is that the input texts follow the grammatical rules of English. In general, verbs correspond to actions and nouns phrases to objects. When it comes to extracting general purpose, unsupervised relations, subject-verb-object (SVO) -like extractions are sometimes used (Angeli, et al., 2015),(Atapattu, et al., 2014), (James, 2017). This choice is often made without discussing why such an extraction is appropriate. A limitation is that such an extraction is a syntactic one. In particular, while the role of the subject and object often imply the roles of doer and is-done-to, this isn't always the case. For example:

The apple was eaten by her.

"The apple" is the subject, yet "her" is carrying out the activity. An alternative to an SVO extractor is a semantic role labeller. Depending on the type of semantic role labelling paradigm, "her" may be indicated as the agent and "The apple" as a patient. No purely syntactic or semantic relation extractor will capture all actions in a text. There are some actions only understood on the pragmatic level. These expect a common understanding from the reader to interpret.

He was young. Now he is old.

Most human readers understand that there is the implicit action of aging present. By inferring that both occurrences of “he” are referring to the same person, and that “young” and “old” are concerned with something’s age, a higher level of syntactic understanding may be gained. If there is also the understanding that, in this case, the first sentence chronologically precedes the second sentence, a similar understanding to the action “aging” is gained.

A system is developed which attempts to 1) extract meaningful semantic relationships, which can be understood as action traces, 2) infer a temporal ordering between events 3) and extract a lexicon of objects. This system is described in the following section

3. Approach

This pipeline generates two data structures: a sequence of ordered actions and an object lexicon. The raw document is labelled by three off-the-shelf natural language processing packages. The first is a temporal labeller, the TARSQI toolkit (TTK). This identifies events and the temporal links between events (Verhagen & Pustejovsky, 2008). The second labeller is the AllenNLP semantic role labeller (SRL) (Shi & Lin, 2019). The labels generated are used to identify actions and objects. The third labeller is the Stanford CoreNLP annotator. These produce more general NLP annotations, such as dependency graphs and parts-of-speech annotations.

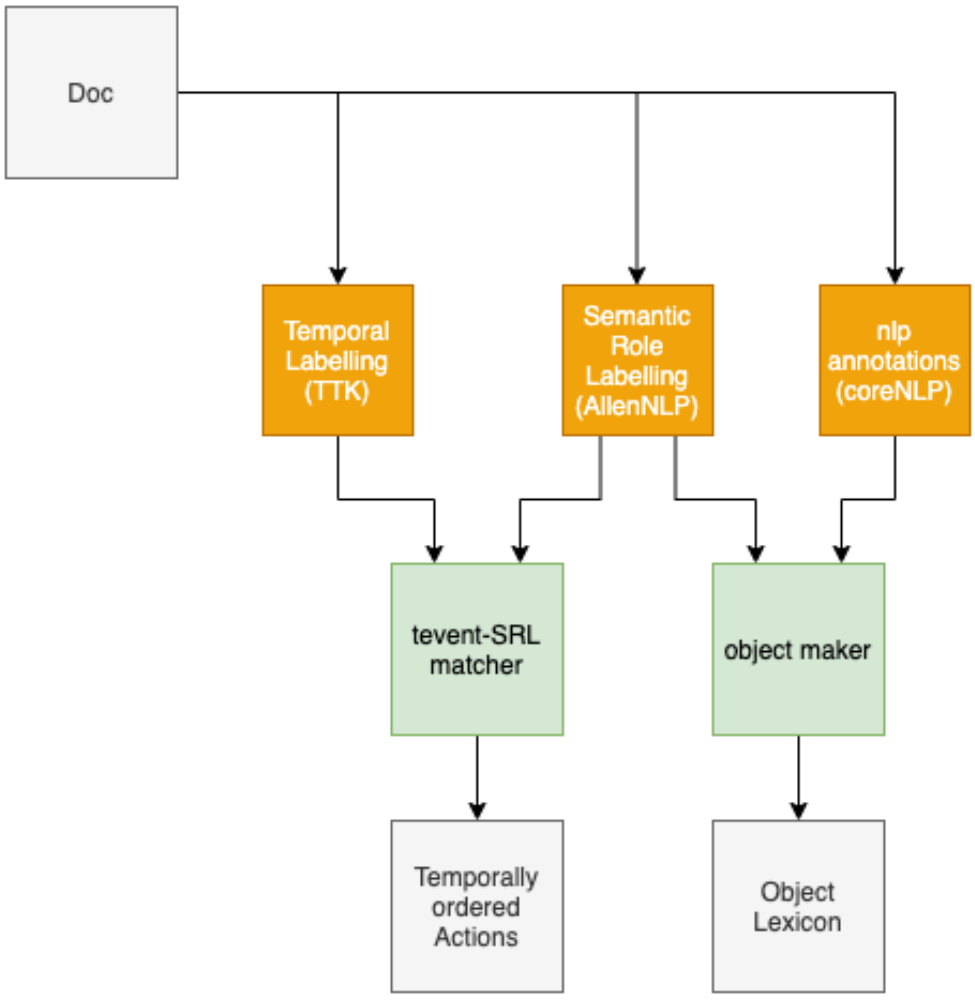

Figure 1: System architecture diagram. Green boxes are the components developed for this system.

The labelled documents are passed to two custom built components for this task. The tevent-SRL matcher matches an SRL extraction to a TTK event. The object maker aims to group together synonyms and forms an object might have and infers a basic object type. It uses Stanford CoreNLP annotations, such as its part-of-speech tagger and coreference resolution tagger to achieve this.

The next section discusses the data used. Proceeding sections will describe a component of the system, starting with background and then implementation.

3.1. Data

This system creates action sequences generated from its input texts. This means that if a planning were to be generated by these sequences, it would be flavoured by this system's input texts. Actions and objects would be named, and would have the logical relationships, of the input texts.

This system is tuned for news articles. A central component of this system is the TARSQI toolkit (TTK), which is tuned for news articles. This creates a dependency within this system on the type of input data best suited for it, removing some domain independence. Despite this apparent bias, TTK seems to perform competently in other text types, such as fairy tales.

Besides TTK's bias towards news, new articles are appropriate input for this system. There is an abundance of news articles. News articles contain real-world events, many of which have dates. As a result, narratives generated would look like a fictionalised version of the wold of news and current affairs.

4. Temporal annotation

Temporal annotation is the process of identifying events in natural language texts and identifying an order between them. Common temporal ordering processes use a combination of machine learning mechanisms and rule-based systems (Bethard, et al., 2017). The TempEval competition encourages development in the field of temporal annotation.

The problem of extracting chronologically ordered events is tackled by Ling & Weld (2010), who order events using a "points" approach (relating to temporal starting "points" and end "points"). This allows temporal annotations to be subject to algebraic operations. This is in contrast to the orderings given by the TARSQI Toolkit – where the tags "BEFORE", "AFTER" and "SIMULTANEOUS" are given. Using a points approach reduces vagueness. However, vagueness is present human language for implying a degree of uncertainty. When recalling past events, implying uncertainty may better reflect the history of the events according to the speaker's understanding.

The TARSQI Toolkit (TTK) is a temporal ordering pipeline that annotates a natural-language document, identifying events and ordering between events (Verhagen & Pustejovsky, 2008). Events identified by TTK are minimal in their descriptions. The extent to which an event is described to a human reader is a single token, which is a verb. This verb will form the action-name of the eventually extracted action instance. Of course, this leaves room for improvement – it begs the question: "is the verb tied the TTK event a suitable description of the action which has taken place?".

TTK's output is formatted, like many other temporal annotators, in TimeML specifications (Pustejovsky, et al., 2005). Links between events are specified by TLINK tags. Components of TARSQI have been trained on news data (TimeBank is the corpus TARSQI was trained on). An example extraction is given below:

"…then prime minister Julia Gillard argued that India was planning to generate 40 per cent of its electricity…"

```
<TLINK "argued" relType="AFTER" "planning" />
```

This indicates that the event "Julia Gillard argued…" occurred after the event "…India was planning…".

4.1. Implementation

In order to generate cohesive narratives, the ordering of events has to be logically sound. Logical contradictions can disrupt planning domain extractors (Cresswell, et al., 2013).

After tagging, the temporal ordering system assumes transitivity between evens. For example, given the following temporal annotations:

```
<soccer match, BEFORE, Evelyn scores>,
<Evelyn scores, BEFORE, Evelyn leaves>
```

It is implied that:

```
<soccer match, BEFORE, Evelyn leaves>
```

The implemented system constructs a series of short timelines by linking events together by the transitive property.

5. Relation extraction

Relation extraction is concerned with identifying entities and the relationship between them. This can be a binary relation on two entities, with the relation's name describing the relationship (Angeli, et al., 2015). In some implementations, the name would correspond to the main verb of a clause, and its subject and object as entities. Hence it appears as a grammatical SVO extraction. Using a main verb as the relation name is useful as, most actions are verbs.

The relation's name provides a possible link to events extracted by the temporal annotation phase (as events tagged by TTK are anchored to a verb). There are limitations to this type of extraction, however. To recall an example mentioned earlier:

> The apple was eaten by her.

We see a limitation present from an SVO IE extraction. Such an extractor, such as Stanford CoreNLP OpenIE would extract the following representation: {subject: "The apple", verb: "eaten by", object: "her"}. Subject and object are syntactic roles which don't provide information about what object is carrying out such a role.

An alternative approach is a semantic role labeller (SRL). A semantic role labeller such as the AllenNLP implementation of a *bidirectional encoder representations from transformers* (BERT) model (Shi & Lin, 2019) overcomes this syntactic limitation. The extraction given by the labeller is:

```
{verb: eaten, arg0: by her, arg1: The apple}
```

The agency of "her" is implied by appearing as an earlier argument. Sentences with indirect objects are also better described. For example:

"The apple was given to her by the witch."

Gives the extraction:

```
{verb: given, arg0: by the witch
            , arg1: The apple
            , arg2: to her}
```

Here, three arguments are extracted, besides the usual two from an SVO extractor. In this case "her" is an indirect object. Identifying and ordering semantic roles by their agency in relation to the action preserves the argument order of the lemma of the verb. This helps preserve consistency across different verb forms. The sentence "The witch gave the apple to her" generates a similar extraction that above.

Another benefit is that the AllenNLP semantic role labelling adds temporal and locations annotations to its extractions. These may prove valuable for future extraction operations. For the example sentence used in the previous section, the extractions are as follows:

Original Text:
At the 2011 ALP national conference, then prime minister Julia Gillard argued that India was planning to generate 40 per cent of its electricity with nuclear energy by 2050.
AllenNLP SRL Extraction:
Argued(a0: prime minister Julia Gillard, a1: that India was planning to generate 40 per cent of its electricity with nuclear energy by 2050) [Location: At the 2011 ALP national conference] [Temporal: then]

planning (a0: India, a1: to generate 40 per of its electricity with nuclear energy by 2050) generate (a0: India, a1: 40 per cent of its electricity with nuclear energy) [Temporal:2050]
Stanford CoreNLP OpenIE Extration:
argued at (prime minister Julia Gillard, 2011 ALP conference) argued at (minister Julia Gillard, 2011 ALP national conference) argued at (prime minister Julia Gillard, 2011 ALP national conference) argued at (minister Julia Gillard, 2011 ALP conference) generate (India, 40 per cent) generate (India, 40 per cent of its electricity)

Table 1 Comparing relation extractions

These benefits lead to the decision to use the AllenNLP SRL as the relation extractor for this system. The SRL labeller lack structural annotations, such as sentence and token index, so work has to been done to infer these through using a natural language processing toolkit.

6. Natural language processing

Natural language processing (NLP) is the process in which a machine processes natural language text. Natural language texts are texts written in a human language such as English. Stanford CoreNLP (Manning, et al., 2014) is a natural language toolkit which annotates natural language texts. Most annotation are used in a wide variety of NLP problems.

Annotations used in this project include CoreNLP's part-of-speech tagger (Toutanova, et al., 2003), coreference resolution (Recasens, et al., 2013), named entity recognition and dependency parser.

7. Action instance and object lexicon extraction

This section discussing the extraction of the action instance-like structures from the text. In the extraction system, this is the tevent -SRL matcher. tevent stands for "**t**emporal event". Extracting actions for use in timeline just concerns matching the word anchoring an event with the verb identified in SRL matcher. This trivial procedure has fails match 8% of events. This could be due to the fact that the SRL labeller seems to make an extraction for each main verb in a sentence, and temporal events are also anchored on main verbs.

Dependencies between actions and inferring object type can be used to help generate a planning model (Cresswell, et al., 2013). Understanding the type of an object could help in inferring the parameter of actions it is acted on by. Hayton, et al,. (2020) stores a dictionary of synonyms for objects, so that actions instances with the same argument as an object understanding it as the same object (Hayton, et al., 2020).

This system presents an object lexicon. Its goal is similar to that of coreference resolution; indeed, it uses coreference resolution in its construction, except that the lexicon makes sense outside a certain text. A use of this is to identify object between different texts, with similar subject matter, such as news articles.

Named entity recognition (NER) is an NLP technology that can assist in such a goal. A NER extracts named entities, such as persons' names and locations. They also label what type of object the named entity is. Extracting the name of an entity can connect the entity between two texts. The type a NER extracts can be used as the objects type. The ability for NER systems to identify persons can be of particular importance in this problem domain. This is because contemporary writing advice urges writers to mould stories, in large part, from characters and relationships (Porteous, et al., 2013). This system uses the Stanford CoreNLP NER recogniser for this goal.

The object lexicon is also meant to help out in another problem. This is the issue of nested clauses encountered in extracting arguments for a verb. For example, the sentence:

> Max planned to move his bottle and laptop before cooking his food.

Generate the following two extractions:

```
{verb:  planned, arg0: Max, arg1: to move his bottle and laptop}

{verb: move, arg0: Max, arg1: his bottle and laptop}
```

Multiple extractions are necessary if the aim is to avoid nesting. The aim is to avoid having long arguments, so that generalisations can be made about the action. Argument 1 of "planned" does not represent an atomic object about the world. It has little meaning by itself. But if we consider the fact that the position of a verb's argument holds semantic meaning we can understand that the nested clause makes sense within the context of its parent verb. In this case, the second the argument for "plan" is usually a plan. Hence, we can extract an action head which appears as this:

```
planned (person, plan)
```

We shall call the generalisation of this construct an utterance, connected to the verb. It will appear in this form: `verb_utternace`. So, `plan_utterance` is the extraction for plan. This is identified every time an argument is a nested clause. Other verbs that have utterances as arguments include "thought" and "perceived".

8. Results & discussion

A couple of paragraphs from Cinderella is used as a sample. One the sequences is presented below:

```
1: rattled (her father, her)

2: dared (" ", The poor girl, not tell her father who would have rattled her off)

3: bore (The poor girl)

4: called(" ", her, Cinderwench)

5: made (cinders and ashes, her commonly be called Cinderwench)
```

For readability, non-argument tags have been removed. For example, action 4 had the temporal modifier “commonly”. In extracted action sequences, the 0^{th} argument of an action is the doer of an action. When this is not specified, empty quotes are used to indicate its absence. To interpret these in plain English, place the verb between the 0^{th} and 1^{st} argument. The effectiveness of the sequence generator is based on TTK’s performance. Most of sequences generated are relatively short, usually 2 to 6 actions. Matching the extractions to the TTK events is trivial and highly effective. Events are compared with SRL extractions that appear within the same sentence and are matched on whether they are focused on the same verb. With no more modifications, only 8% of events cannot find a match.

Below is the object lexicon that relates to objects in the above timeline. The first item for each entry is the head of the object. Next is the object’s type. Finally, are the object’s supporting tokens. These are tokens which have appeared alongside the head and may be used to differentiate from a similar object. The first entry appears twice in the sequence, in action instance 2 and 3. It did not identify “her” in action 4, as also being the poor girl. These sorts of misses are too common for the object lexicon, and work is needed to improve accuracy.

"girl" , "ANIMATE", ["girl", "poor"]

"tell", "dared_utterance", ["off", "have", "tell", "father", "rattle"]

"father", " ANIMATE ", ["father"]

"call", "made_utterance'", ["be", "call", "Cinderwench"]

"cinder", "'Undetermined'",["ash", "cinder"]

"Cinderwench","'Undetermined'", ["Cinderwench"]

9. Related and future work

Other narrative generation techniques outside of classical planning, such as non-deterministic models for creating a narrative world may be investigated. Cause and effects between actions, may be inferred by SLINKs, which are a subordinate link. These related to the form, if A happens, then B will happen.

In constructing the object lexicon, ontologies such as WordNet may be leveraged to infer an object's type, and an action's parameter. This has been done in narrative generation. The semantic roles of arguments were learnt by extracting frame semantic components and demonstrated in (Chambers & Jurafsky, 2009). That paper used shared verb arguments to infer greater semantic information.

Action sequence may be made large by exploiting facts on the pragmatic level, such as ordering two disconnected sequences based on a date each is associated with.

10. Conclusion

This research presented a method for extracting action sequences and an object lexicon for use by a planning domain acquiring system. With further research, it may be possible to generate narrative planning models from such sequences. Temporal annotation and semantic role labelling are natural language processing technologies used to make an extraction. A review of existing work in these fields was also given.

References

1. Allen, J. et al., 2015. *SemEval-2015 Task 5: QA TempEval.* [Online] Available at: http://alt.qcri.org/semeval2015/task5/ [Accessed June 2020].
2. Amir, E., 2005. *Learning Partially Observable Deterministic Action Models..* s.l., IJCAI International Joint Conference on Artificial Intelligence.
3. Angeli, G., Premkumar, M. J. & Manning, C. D., 2015. *Leveraging Linguistic Structure For Open Domain Information Extraction.* s.l., Association of Computational Linguistics (ACL).
4. Atapattu, T., Falkner, K. & Falkner, N., 2014. *Proceedings of the 7th International Conference on Educational Data Mining.* s.l., Educational Data Mining .
5. Bethard, S., Savova, G., Palmer, M. & Pustejovsky, J., 2017. *SemEval-2017 Task 12: Clinical TempEval.* s.l., Association for Computational Linguistics.
6. Branavan, S., Kushman, N., Lei, T. & Barzilay, R., 2012. *Learning High-Level Planning from Text.* Jeju Island, Korea , Association for Computational Linguistics .
7. Chambers, N. & Jurafsky, D., 2009. *Unsupervised Learning of Narrative Schemas and their Participants.* Suntec, Singapore, ACL and AFNLP.
8. Cresswell, S., Mccluskey, T. & West, M., 2013. *Acquiring planning domain models using LOCM.* s.l., The Knowledge Engineering Review 28.
9. Feng, W., Zhuo, H. H. & Kambhampati, S., 2018. *Extracting Action Sequences from Texts Based on Deep Reinforcement Learning.* s.l., International Joint Conference on Artificial Intelligence (IJCAI-18).
10. Fox, M. & Long, D., 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research ,* Issue 20, pp. 61-124.
11. Ghallab, M., Nau, D. & Traverso, P., 2016. Chapter 2: Deliberation with Deterministic Models. In: *Automated Planning and Acting.* s.l.:Cambridge University Press, pp. 24 - 37.
12. Hayton, T., Porteous, J., Ferreira, J.̃. F. & Lindsay, A., 2020. *Narrative Planning Model Acquisition from Text Summaries and Descriptions.* s.l., Association for the Advancement of Artificial Intelligence.

13. James, W., 2017. *Event Extraction and Temporal Ordering towards Narrative Model Generation.* Canberra, Honours Thesis, the Australian National University.

14. Lindsay, l. et al., 2017. *Framer: Planning Models from Natural Language Action Descriptions.* s.l., International Conference on Automated Planning and Scheduling (ICAPS).

15. Ling, X. & Weld, D. S., 2010. *Temporal Information Extraction.* s.l., Association for the Advancement of Artificial Intelligence.

16. Manning, C. D. et al., 2014. *The Stanford CoreNLP Natural Language Processing Toolkit.* s.l., Association for Computational Linguistics.

17. Matuszek, C., 2018. *Grounded Language Learning: Where Robotics and NLP Meet.* California, International Joint Conference on Artificial Intelligence .

18. Matuszek, C., 2018. *Grounded Language Learning: Where Robotics and NLP Meet.* s.l., IJCAI.

19. Mei, H., Bansal, M. & Walter, M. R., 2016. *Listen, Attend, and Walk: Neural Mappingof Navigational Instructions to Action Sequences.* s.l., AAAI.

20. Minard, A.-L.et al., 2015. *SemEval-2015 Task 4: TimeLine: Cross-Document Event Ordering.* Denver, Colorado, Association for Computational Linguistics.

21. Mourao, i., Zettlemoyer, L. S., Petrick, R. P. A. & Steedman, M., 2012. *Learning STRIPS Operators from Noisy and Incomplete Observations.* s.l., Uncertainty in Artificial Intelligence.

22. Porteous, J., Charles, F. & Cavazza, M., 2013. *NetworkING: using Character Relationships forInteractive Narrative Generation.* s.l., AAMAS.

23. Pustejovsky, J. et al., 2005. *The Specification Language TimeML.* s.l., s.n.

24. Recasens, M., Marneffe, M.-C. d. & Potts, C., 2013. *The Life and Death of Discourse Entities: Identifying Singleton Mentions.* s.l., NAACL.

25. Riedl, M. O. & Young, R. M., 2010. Narrative Planning: Balancing Plot and Character. *Journal of Artificial Intelligence Research ,* Issue 39, pp. 217-267.

26. Shi, P. & Lin, J., 2019. *Simple BERT Models for Relation Extraction and Semantic Role Labeling.* s.l., arXiv e-prints.

27. Sil, A., Huang, F. & Yates, A., 2010. *Extracting Action and Event Semantics from Web Text.* s.l., Association for the Advancement of Artificial Intelligence.

28. Smith, N., 2004. *Chomsky: Ideas and Ideals.* s.l.:Cambridge University Press.

29. Toutanova, K., Klein, D., Manning, C. & Singer, Y., 2003. *Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network.* s.l., North American Chapter of the Association for Computational Linguistics (NAACL) , pp. 252-259.

30. University of Montana Writing Center, n.d. *Five Basic Structures of Simple Sentences.* [Online]
Available at:
https://www.umt.edu/writingcenter/docs/resourcesforwriters/fivestructures.pdf

31. UzZaman, N., Llorens, H., Derczynski, L. & Verhagen, M., 2013. *SemEval-2013 Task 1: TEMPEVAL-3: Evaluating Time Expressions,Events, and Temporal Relation.* Atlanta, Georgia, Association for Computational Linguistics.

32. Verhagen, M. & Pustejovsky, J., 2008. *Temporal processing with the TARSQI Toolkit.* s.l., COLING.

33. Ware, S. G. & Young, R. M., 2011. *CPOCL: A Narrative Planner Supporting Conflict.* s.l., Association for the Advancement of ArtificialIntelligence .

34. Yordanova, K., 2016. *From Textual Instructions to Sensor-based Recognition of User Behaviour.* Sonoma, CA, ACM IUI.

35. Young, M., Ware, S., Cassell, B. & Robertson, J., 2013. Plans and Planning in Narrative Generation: A Review of Plan-Based Approaches to the Generation of Story, Discourse and Interactivity in Narratives. *Sprache und Datenverarbeitung: International Journal of Language Processing,* Issue 37, pp. 41-64.

36. Zhuo, H. H. & Kambhampati, S., 2013. *Learning Action-Model Acquisition from Noisy Plan Traces.* Beijing, China, AAAI Press.