David Ziegler

Determining the Temporal Order of Events in Natural Language Using Learned Commonsense Knowledge

Supervisors:
Asst. Prof. Ivan Titov
Dr. Breanndán Ó Nualláin
Ashutosh Modi

Word Count: 8337
Student Number: 10225412
Major: Science

Capstone Thesis submitted in partial fulfillment of the requirements for the degree of Bachelor of Science

May 28, 2014

Abstract

Natural language understanding systems, for example question answering or document summarization, can be expected to benefit from knowledge about the temporal order of events in a text. Traditionally, only linguistic features (for example verb tense, syntactic structure etc.) are used to determine the temporal order of events. However, when humans read a piece of text they use commonsense knowledge additional to the information in the text to reason about temporal order (for example in a restaurant setting the event `eat` usually precedes `pay` even if this information cannot be deduced from the text).

This thesis tests whether using commonsense knowledge about the prototypical order of events can improve the performance of a temporal ordering system. To this end, the output of Modi & Titov's (2014a) classifier for learning prototypical temporal order of events is used as an additional feature for Bethard's (2013) temporal order classifier 'ClearTK TimeML' and is evaluated on the TempEval-3 task (Uzzaman et al., 2013).

The performance using both the 'commonsense' features as well as ClearTK's linguistic features is similar to using linguistic features only. However, the classifier learning 'commonsense knowledge' is only trained on the same data that is used as training data for ClearTK while it should be trained on a much larger separate data set in order to be able to call it 'commonsense knowledge'. The 'commonsense' feature taken by itself reaches a similar performance as the linguistic features, although using a very different approach, which shows that this feature is predictive. The results suggest that using 'commonsense' knowledge is a promising approach to further pursue, however, this knowledge should be learned from much larger data sets.

Contents

List of Tables

List of Figures

Appendices

1 Introduction

Natural language understanding systems are created in order to facilitate interaction between humans and computers. One prerequisite for understanding natural language is determining the order in which events happen that occur in a text. Understanding temporal order can be beneficial for a number of Natural Language Processing (NLP) applications, such as, for example, question answering and document summarization.

In order to understand a piece of text, humans do not only use explicitly stated information but combine this with commonsense knowledge about the world. Consider the following excerpt from a news item:

(1) *The action* ***came*** *in response to a petition* ***filed*** *by Timex Inc. for changes in the U.S. Generalized System of Preferences for imports from developing nations.* [1]

When reading this piece of text humans know that, contrary to the order in which the events are mentioned in the text, the petition was first filed and then the government took an action. Similarly, natural language understanding systems (computer systems that induce a representation of the meaning of natural language) can be expected to benefit from commonsense knowledge about the prototypical ordering of events (Modi & Titov, 2014a).

Modi and Titov (2014b) developed a classifier to learn the typical ordering of events. It is representing events as a combination of verbs and their arguments (subject, object etc.). A machine learning *classifier* is a system that decides to

[1] Excerpt from the file wsj_0026.tml in the TimeBank corpus. See section 4.1 for a description of the data sets used.

which class an observation belongs to, based on *training data*. In *supervised learning*, the training data is annotated with the correct outcomes such that the system can learn from it and *predict* outcomes for the test data. If these annotations are created by humans, the training data is called *gold-standard* and the annotations are called *gold annotations*. As an example, assume that the ordering of event A and event B are to be predicted. A classifier then uses a set of *features* of these events in order to predict whether A precedes B or B precedes A. The features are in this case the verbs and arguments (subject, object etc.) of each event. If, for example, the training data contain a number of events that have similar verbs and arguments as A which precede events that are similar to B then the classifier will make the prediction 'A precedes B'. How the events are represented in Modi & Titov's model in order to determine which events are 'similar' will be discussed later.

Traditionally, systems to determine the temporal order of events in natural language only use linguistic features (syntactic and semantics features extracted from the events and their context). The aim of this thesis is to test whether such a system benefits from knowledge about the typical ordering of events, which Modi & Titov's classifier provides. TempEval-2013[2] is a competition for determining temporal order. The task is the following: given raw text, the systems are supposed to find time expressions, events, and temporal relations between two events or events and times (Uzzaman et al., 2013). The best performing system in TempEval-2013 was ClearTK TimeML by Bethard (2013)[3].

For this thesis the ClearTK system, which currently only uses linguistic features (for example verb tense and aspect) is modified to take the output of Modi & Titov's classifier as an additional feature for the prediction of temporal order of events. If this modified classifier yields better results than Bethard's original results in TempEval-3 it can be concluded that making use of commonsense knowledge about temporal ordering can indeed improve the performance of natural language understanding systems. This thesis will first give some background about temporal

[2]TempEval-3 is part of SemEval-2013, the Semantic Evaluation Exercise by the International Workshop on Semantic Evaluation. Website describing TempEval-3: `http://www.cs.york.ac.uk/semeval-2013/task1/`

[3]ClearTK is a Natural Language Processing framework of which ClearTK TimeML, the temporal ordering classifier, is just one part. The ClearTK website: `https://code.google.com/p/cleartk/`

ordering systems and explain Modi & Titov's classifier in particular, as well as the TempEval-3 tasks. Then, it is explained how the ClearTK system was modified in order to conduct the experiments. The experiments section contains explanations of the data used in different experiments as well as how the results were analysed. The presentation of the results is followed by a discussion to interpret them.

2 Literature Review and Rationale

2.1 Semantic Scripts

Abelson and Schank (1977) introduced the notion of scripts into artificial intelligence. A script is a type of schema (organized simplification of the world) that captures a prototypical sequence of events, e.g. the events involved when going to a restaurant. It has been shown that knowledge about scripts can improve the performance of natural language understanding systems. For example, Miikkulainen (1995) uses script knowledge for automatic paraphrasing of text and question answering. He mentions the following example:

(2) *John went to MaMaison. John asked the waiter for lobster. John left a big tip.*
(Miikkulainen, 1995)

Human readers of this passage, having commonsense script knowledge, infer a number of events that must have happened additional to the events that are explicitly mentioned: that John ate lobster at the restaurant, that he liked it etc. Miikkulainen's system can answer questions like "How did John like the lobster at MaMaison?" using script knowledge learned from data.

Regneri, Koller, and Pinkal (2010) developed a machine learning system to induce script knowledge which they represent in directed acyclic graphs (nodes represent events and an arc from node A to B means 'A precedes B'). Modi and Titov (2014b) argue that graphs are not practical for applications because it is not clear which ordering scenario should be chosen for which context, requiring human decisions for working with this kind of representation. Their system works with a one-dimensional 'timeline' instead, on which all events are ordered.

2.2 Compositional Semantics and Distributed Representations

In order to learn script knowledge from training data one could simply calculate the frequency of occurences where verb A precedes verb B, verb A and B representing two events that are related. However, consider the following example of a number of events: `walk into the restaurant, order at the counter, walk to a table, walk out of the restaurant`. Representing events only by verbs does not seem to yield good results in this case as the event representations would just be: `walk, order, walk, walk`. Therefore, Modi and Titov (2014b) use a simple compositional approach and represent events by verbs and their arguments (e.g. verb = `walk`, argument = `restaurant`).

'Compositional semantics' refers to the idea to take the combination of words into account in order to represent a phrase, as opposed to 'lexical semantics' which is concerned with representing meaning of individual words. Compositional semantics is based on the 'Principle of Compositionality' or 'Frege's Principle' which states that the meaning of a phrase is determined by the meaning of its constituents (Frege, 1892). Compositional semantics has recently been receiving a lot of attention in natural language processing as more accurate meaning representations promise to improve performance for many NLP applications (Mitchell & Lapata, 2010; Socher et al., 2013). The key difficulty in compositional semantics is the 'curse of dimensionality': With a dictionary of tens (or hundreds) of thousands words the number of possible sentences is so large that a phrase for which a prediction should be made is likely to not occur a single time in the training data (Bengio, Ducharme, Vincent, & Jauvin, 2003). This problem can be alleviated if not only the representation of a whole phrase but also the representations of its elements capture their syntactic and semantic properties. For example, if the training data contains the sentence "The cat is walking in the bedroom", the sentence "A dog was running in a room" should be identified as similar because the semantic and syntactic properties of `the` and `a`, `cat` and `dog`, `is` and `was` etc. are similar (Bengio et al., 2003).

Bengio et al. propose to use *distributed representations* for this purpose: in distributed representations a word is represented by a vector of real values, also called *word embedding*, such that each dimension of the vector represents a syntactic or semantic feature of the word. In this space, similar words are close to each other, at least in some of the dimensions. Word embeddings are often learned using neural networks that map a sequence of word embeddings to a prediction, for example temporal order in our case. The word embeddings are learned such that the performance of the predictions is best. The features of word embeddings are usually not explicitly assigned. Since the embeddings are learned and evaluated with the predictions, it is ensured that features are learned that are informative for the predictions of interest. The idea of distributed representations originates from connectionist approaches (representing concepts by neuron-like activation units) (Hinton, 1986). In recent years, using learned distributed representations has been shown to be useful in a number of Natural Language Processing applications (Turian, Ratinov, & Bengio, 2010; Bengio et al., 2003).

2.3 The Commonsense Classifier

Modi & Titov use a neural network model with distributed representations to represent words. A function C is applied to the stem of each word to obtain the word embedding (see Figure 2.1 which shows the composition of `bus`, `disembarked`, and `passengers`). The embeddings are then linearly transformed (using the matrices T and R) and added to obtain an embedding of the whole event (after one more linear transformation using matrix A). C, T, R and A are learned together with the temporal classifier, which ensures that the representations are informative for temporal ordering. The error in the event representation layer is backpropagated to the word representation layer which means that the word embeddings are changed until they are informative determining temporal order.

The machine learning scenario explained in the introduction is based on *supervised learning*, which is a *data-driven* approach. In *unsupervised learning* (when no training data with correct annotations is available to learn from) one can rely on *rule-based* methods which use a fixed set of rules. Such a rule could be for example "If the word `after` occurs between event A and event B, then B precedes A."

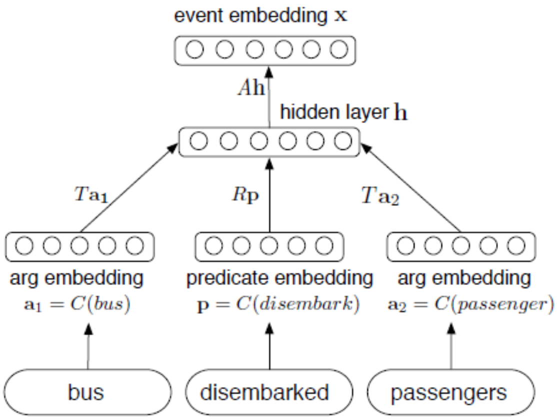

Figure 2.1: Event representations as composition of verbs and their arguments.

Modi & Titov use a rule-based temporal classifier to induce temporal ordering from large unannotated data sets relying on signal words like `then`, `before`, `after` etc. In this way, annotated data is generated which serves as training data for the neural embedding classifier (which is data-driven and does not make use of the signal words). The classifier needs to be trained on very large data sets in order to learn 'commonsense' knowledge about the world (script knowledge). For the scope of this thesis, however, the classifier is only trained on the TempEval-3 data. Thus, it does not really add external 'commonsense'-knowledge but makes use of the same training data, that ClearTK is trained on.

2.4 The TempEval-3 Task

In TempEval-3, the participants are given raw text and need to find and temporally order events and times in natural language text according to the TimeML specifications. TimeML is the Time Markup Language and is described in (Pustejovsky, Ingria, & Sauri, 2005). The TempEval task is divided into three subtasks (Uzzaman et al., 2013).

2.4.1 Task A

Task A includes determining the extent of time expressions (finding the time expressions) according to TimeML `TIMEX3` tags. Times also have two attributes that need to be annotated: TYPE (time, date, duration, and set) and VALUE (time-value).

2.4.2 Task B

Task B asks to find the extent of events in the text using TimeML `EVENT` tags and to determine the CLASS attribute (`occurence`, `perception`, `reporting`, `aspectual`, `state`, `I_state` or `I_action` (Pustejovsky et al., 2005)) although their annotation is not part of Task B, events contain more attributes like tense, aspect, modality, polarity, and part-of-speech, which the systems may use as features. [1]

2.4.3 Task C

Task C entails identifying events or times that have a temporal link, and determining the temporal relation between them. Pairs of entities that can have a temporal link are:
1) event-event (same sentence)
2) mains event-main event (consecutive sentences)
3) event-time (same sentence)
4) event-document creation time

2.4.4 Task C Relation Only

This is the temporal relation part of Task C: given time, event and temporal link annotations, the relation type of these links needs to be predicted.

While ClearTK is capable of doing Task A, B, and C, this project is only concerned with Task C Relation Only as this is the only task for which the script knowledge might be informative.

[1] Some basic linguistic terminology: the *aspect* of a verb is its relation to the flow of time, e.g. "I used to do this." is `imperfective` while "I did this" is `perfective`, *modality* is a category concerning whether beliefs or attitudes are expressed (e.g. with words like "can" and "would"), *polarity* here refers to the degree of positivity expressed, *part-of-speech* is the word class (e.g. `verb` or `noun`), *token* refers simply to the text of a word, and the *lemma* is its canonical form (for example the lemma of 'goes', 'going', 'went' and 'gone' is `go`).

2.5 Participants in TempEval-3 Task C

The participants in the temporal ordering task of TempEval-3 use both rule-based and data-driven approaches.

NavyTime by Chambers (2013) is a hybrid approach using both rule-based and data-driven methods. Some of the features used for temporal ordering are each event's token, lemma, part-of-speech tag, the syntactic path between events etc.

UTTime is a hybrid system as well which also uses paths in the syntactic parse tree among other features Laokulrat, Miwa, Tsuruoka, and Chikayama (2013).

JU-CSE by Kolya, Kundu, and Gupta (2013) is a rule-based system which uses rules features like the following among others: modal context (whether the event's context contains words like `will`, `shall`, `can`, `may`), context words like `before`, `after`, `less than` etc.

KULRunABC (Kolomiyets & Moens, 2013) uses a data-driven approach and morphosyntactic features: each event's token, lemma, part-of-speech tag, the sentence's root verb, temporal signals in the phrase, the annotated event attributes (class, tense, aspect, polarity, modality) and other features.

ClearTK is the best performing system in Task C as well as Task ABC (Uzzaman et al., 2013). However, it did not participate in Task C Relation Only. For Task C these features are used: event attributes (aspect, class, tense), the text of the first child of the grandparent of each event in the constituency tree, the path between the two events through the syntactic parse tree, the tokens between the two events.

All TempEval-3 participants use linguistic features such as syntactic and semantic information, event attributes, tokens in the context etc. but no script knowledge. The undertaking to use script knowledge as a feature, which will be described in the next sections, is therefore a new approach to the TempEval task.

3 Methods

3.1 Pipeline

This section describes how the output of the Commonsense Classifier is added as a feature for the temporal link classifier in the ClearTK project. Figure 3.1 shows the pipeline of the whole process. The gray elements of the pipeline are the ClearTK project by Bethard (2013). ClearTK is written in Java and freely available to download.[1] The original ClearTK classifier for Task C first reads in the existing annotations for time expressions and events (step 4 in figure 3.1). Then, the following features are extracted for the temporal link classifier (step 5): aspect, class, and tense of each event, the text of the first child of the grandparent of the event in the syntactic parse tree, the path from event to event through the syntactic parse tree, and the tokens between the two events (Bethard, 2013). These features are then used to annotate temporal links between events and predict the relation types. The only event-event temporal link classifier used by ClearTK is the `TemporalLinkEventToSubordinatedEvent`-classifier. It only considers events that are in the same sentence and where one event is in a subordinate clause and the other in the independent clause. It only uses the relations `BEFORE` and `AFTER`, out of the 14 possible relation types given by the TimeML specifications, since it focuses on temporal ordering for the relation types.[2]

In order to obtain the output from the Commonsense Classifier we run it in a preprocessing stage before the ClearTK pipeline. Before this, we first need to extract the information from the training data that the Commonsense Classifier

[1] `https://code.google.com/p/cleartk/downloads/list`

[2] The possible relation types according to TimeML are: `BEFORE`, `AFTER`, `INCLUDES`, `IS_INCLUDED`, `DURING`, `SIMULTANEOUS`, `IBEFORE` = immediately before, `IAFTER`, `IDENTITY`, `BEGINS` `BEGUN_BY`, `ENDS`, `ENDED_BY`, `DURING_INV` = inverse of during (Pustejovsky et al., 2005).

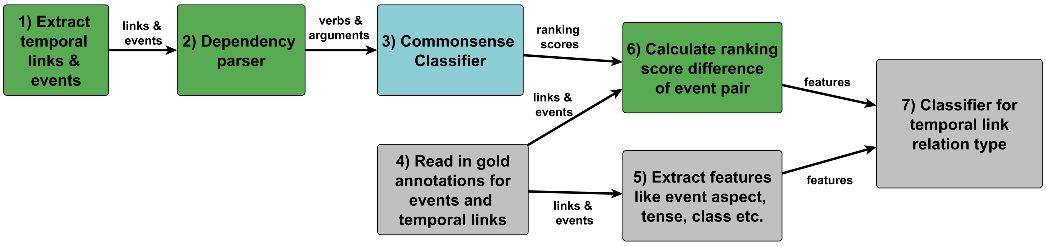

Figure 3.1: Pipeline; gray: ClearTK-TimeML project (Bethard, 2013), light blue: Commonsense Classifier (Modi & Titov, 2014b), green: this project's additions.

takes as input. In step 1 (in figure 3.1), all events that occur in a temporal link are extracted. Using a dependency parser the arguments of the verbs representing events are extracted. The Commonsense Classifier takes these verbs and their arguments as input and outputs a ranking score for each event which determines its position on the timeline of events (step 3). Then, the ClearTK TimeML pipeline is run which reads the event and time expression annotations from the data and extracts features for the temporal link annotator. The difference to the original ClearTK pipeline is, that at this point (step 6), the ranking scores for the events for which the links are to be predicted are considered. The difference of the ranking scores of each event pair is calculated and passed to the classifier together with the other features.

The green elements in figure 3.1, which are the contributions to the pipeline by this project, are described in more detail in the following.

3.2 Preprocessing

The training and test data are annotated in the *TimeML* format (Pustejovsky et al., 2005). This is an XML format that contains tags among other things for time expressions, events, and temporal links. Listing 3.1 shows an excerpt of a file from the training data[3]. It contains event annotations (`looking`, `battle`, and `disappointed`), and temporal links (e.g. a `BEFORE` relation from `looking` to `disappointed`). Moreover, the events are annotated with attributes such as aspect, tense, polarity, part-of-speech (in the `makeinstance` tags), and class.

[3]This is an excerpt from the file ABC19980120.1830.0957.tml which is part of the TimeBank corpus. See section 4.1 for a description of the TimeBank data set.

Listing 3.1 : Excerpt of a Training Document

```
...
<TEXT>
Those observers <EVENT eid="e51" class="I_ACTION">looking</EVENT> for a
<EVENT eid="e52" class="OCCURRENCE">battle</EVENT> between uncompromising
representatives and very different ideologies will, in all likelihood, be
<EVENT eid="e53" class="STATE">disappointed</EVENT>. Castro has said that
you can be a Communist and still be a Christian.

At the end of the broadcast <TIMEX3 tid="t1000" type="TIME" value="1998-01-20
    TEV">this evening</TIMEX3>, one more trip around Havana to see what it's
been like since the last time.
</TEXT>

<MAKEINSTANCE eventID="e51" eiid="ei449" tense="PRESPART" aspect="NONE"
       polarity="POS" pos="VERB" />
<MAKEINSTANCE eventID="e52" eiid="ei450" tense="NONE" aspect="NONE"
       polarity="POS" pos="NOUN" />
<MAKEINSTANCE eventID="e53" eiid="ei451" tense="FUTURE" aspect="NONE"
       polarity="POS" pos="VERB" />

<TLINK lid="l28" relType="INCLUDES" eventInstanceID="ei449"
       relatedToTime="t0" />
<TLINK lid="l30" relType="BEFORE" eventInstanceID="ei449"
       relatedToEventInstance="ei451" />
<SLINK lid="l45" relType="MODAL" eventInstanceID="ei449"
       subordinatedEventInstance="ei450" />
```

The Commonsense Classifier takes scripts encoded in an XML format as input, which contains verbs and their arguments (see Appendix A). Although only verbs and arguments are used by the Commonsense Classifier, all other available information about the events (such as the event attributes aspect, class, polarity etc.) is also included in the input data in case we decide to make use of it in the future.

Not all temporal links that are present in the data can be used as input for the Commonsense Classifier. There are three criteria that a link must meet to be included in the input data:

1. **Event-event link:** The data contains event-event and event-time links. However, the Commonsense Classifier only learns the temporal order of events, thus, only event-event links are included.

2. **Both events are verbs:** The Commonsense Classifier only uses events that are verbs. This is because it is based on using the verb and their arguments. Some events in the data are nouns or other parts-of-speech but for these, it is not as easy to define what the arguments would be.

3. `BEFORE/AFTER` **link:** The `TemporalLinkEventToSubordinatedEvent`-classifier only predicts `BEFORE` and `AFTER` relations. Therefore, links that do not have the relation type `BEFORE` or `AFTER` do not need to be included for the Commonsense Classifier.

As only links that comply to the above criteria are processed by the Commonsense Classifier and get ranking scores these links will be referred to as *ranking score links* in the following.

Four steps are carried out to create input for the common sense classifier from the training/test data:

1. For each event: retrieve the sentence it occurs in and other information from the TimeML annotations.

2. Find all `BEFORE/AFTER` links between events.

3. For events that are verbs run the Stanford Dependency Parser[4] to retrieve their arguments.

4. Add the information about events gathered in steps (1) and (3) to the respective events in the list of temporal links from step (2).

More specifically, these actions are performed for the above steps:

1. **Events:** In order to retrieve the sentence for each event the sentences are first split using the sentence classifier from Stanford CoreNLP. This is more

[4]The Stanford Parser is part of Stanford CoreNLP, a collection of natural language analysis tools (Marneffe, MacCartney, & Manning, 2006). In this project it is used for annotations of part of speech, named entity recognition and dependency parsing.

accurate than just for example splitting sentences at each period because periods can also occur in abbreviations such as “U.S.”. Then, the events and surrounding sentences are extracted by means of a regular expression.
After this step the following information about each event is known: event ID, event text, event class, sentence ID, sentence text, and document name.

2. **Temporal Links:** For each temporal link (`tlink` tag in TimeML):

   - It is tested whether it is an event-event link.
   - It is tested whether the relation type of this link is either `BEFORE` or `AFTER`.

   The outcome of this step is a list of temporal links, each containing two temporal entities (events or times) and the relation type. If the two criteria above are met, a temporal link is marked as `included`.

3. **Dependency Parser:** For each temporal link in the list from step (2) carry out the following steps for each of the two events:

   - Use the Stanford CoreNLP part-of-speech tagger to determine whether the event is a verb. If any of the two events is not a verb set `included` to `false` for this temporal link.
   - The Stanford Dependency Parser (Marneffe et al., 2006) is used to retrieve each verb's arguments. For example the verb `disappointed` in the sentence in listing 3.1 has the dependencies: `observers` (NSUBJ), `will` (AUX), `likelihood` (PREP_IN), and `be` (COP)[5]
   - It is tested whether the dependency type is one of the following in which case the dependency is not included: AUX, AUXPASS, ATTR, APPOS, CC, CONJ, COMPLM, COP, DEP, DET, PUNCT, MWE. Dependencies of these types proved not to be predictive for the Commonsense Classifier.

[5] The syntactic dependency types are given in brackets: NSUBJ = nominal subject, AUX = auxiliary, PREP_IN = prepositional modifier with `in`, COP = copula. See Marneffe and Manning (2008) for a list of all syntactic dependencies that can be outcomes by the Stanford Dependency Parser.

- Other than the dependency parser, Stanford CoreNLP was also used to lemmatize the words and retrieve named-entity-recognition (NER) and part-of-speech labels (POS).

After this step the following information about each events is known: event text, lemma, POS, NER, token ID, and sentence ID of the verb and text, lemma, POS, NER, token ID, sentence ID, and dependency type of each argument. Although the Commonsense Classifier currently only uses verbs and arguments as features all additional information is written to the files in case we would like to use this information in the future.

This section described how the input data for the Commonsense Classifier is prepared. After running the classifier we need to integrate its output into the ClearTK project which is explained in the following section.

3.3 Modifying ClearTK TimeML

3.3.1 The Original ClearTK TimeML Project

In order to describe the changes made to the ClearTK project this section first explains the original project and some terminology.

ClearTK is implemented in the Unstructured Information Management Architecture (UIMA)[6]. The basic goal of the UIMA framework is the processing of any kind of unstructered information, such as text, audio, video etc. (which does not express much meaning from the viewpoint of a machine) to create structured information, for example tokens, semantic entities, people, places, times, events, opinions etc. such that machines can deal with it. In UIMA, these bits of structured information are called *analysis results* and they are created by *annotators*. All analysis results pertaining to one document are stored at one place, the *Common Analysis Engine (CAS)*. This ensures that each annotator can use the analysis results of other annotators, as each annotator reads from and writes into the CAS. For example, in ClearTK the following annotations are produced after each other

[6]See `http://uima.apache.org/downloads/releaseDocs/2.2.2-incubating/docs/html/overview_and_setup/overview_and_setup.html\#ugr.ovv.conceptual` and Ferrucci and Lally (2004) for an overview over UIMA .

where some of the annotators depend on the annotations of others: sentences, tokens, part-of-speech of tokens, token stems, syntactic parsing, extent of event expressions, event type, event tense, temporal links between events and so on. The analysis results in the CAS have a certain *annotation type*, for example the types Time, Event and TemporalLink in the case of ClearTK.

In ClearTK the evaluation for TempEval-3 is performed by running the *TempEval2013* class. It provides the possibility to either do Task ABC or Task C. It first trains the model, then makes predictions on the test data. In order to train, all gold annotations are added into the CAS, then the pipeline with all annotators runs. In training mode, each annotator does not do much more than retrieving those gold annotations that are relevant to it from the CAS (events, times, temporal links etc.) and saving them in a file such that they can later be used to make predictions. For testing, the pipeline of annotators runs again, but this time each annotator produces annotations based on predictions made by the models, using the training data annotations. For predictions each model uses a number of features of the annotations. The `TemporalLinkEventToSubordinatedEventAnnotator` which annotates temporal links and relation types between two events uses the following features: aspect, class, and tense of each event, the text of the first child of the grandparent of each event in the constituency tree, the path through the syntactic constituency tree from one event to the other, and the tokens between the two events (Bethard, 2013). The features are obtained using *feature extractors*. An extractor takes an annotation (for example an event) as input and retrieves some information about it that can be used as a feature for a classifier (for example the tense of the event).

3.3.2 Modifying ClearTK to Perform Task C Relation Only

ClearTK can either perform the TempEval2013 Task ABC (annotate times, events, temporal links and relation types) or Task C (given times and events, annotate temporal links and relation types), but did not participate in Task C Relation Only (given times, events and temporal links, annotate the relation types). As the ranking scores provided by the Commonsense Classifier only help in predicting relation types the most accurate approach for this thesis is to test on Task C

Relation Only. We modified ClearTK to do Task C Relation Only in the following way:

- Exclude the annotators for time extent, time type, event extent, class, aspect, modality, and polarity from the pipeline from the pipeline and only include `TemporalLinkEventToSubordinatedEventAnnotator`. The annotators for temporal link event to document creation time and temporal link event to sentence time (Task C) are not included as they are concerned with event-time links.

- An annotator is added to the testing pipeline which copies the temporal link gold annotations into the CAS when testing. The relation types are then removed because these are supposed to be predicted.

- The `TemporalLinkEventToSubordinatedEventAnnotator` is the child of `TemporalLinkAnnotator_ImplBase`. We modified this class such that in testing, it first creates a list of all existing temporal links in the CAS and their reverses. Then, for each link to be annotated (each event pair that satisfies the condition that one is in the subordinate clause of a sentence) it is checked whether it occurs in the list. Only for those links the relation type is predicted.

3.3.3 Adding Output of Commonsense Classifier as a Feature

In order to predict the relation types of temporal links between two events their ranking scores obtained by the Commonsense Classifier might be informative. The ranking scores give each event a position on a timeline. Therefore, the difference of the ranking scores of two events indicates the order of the events. We hypothesize that not only the sign of the difference matters but that the quantity of the difference gives a measure for confidence of some kind. The ranking score difference feature is referred to as *RankingScoreDifference* in the following, matching the feature's name in the code. The following changes are made to ClearTK such that the `EventToSubordinatedEvent`-classifier uses the RankingScoreDifference as a feature:

- For each UIMA annotation type, there is an xml file specifying its attributes etc. We added the RankingScoreDifference to the features of the *Event* type.

- An extractor for the RankingScoreDifference feature was added: The *RankingScoreDifferenceExtractor* takes two events as input, retrieves their ranking scores and returns the difference of the ranking score of the target event minus the ranking score of the source event. If either one of the events does not have a ranking score the RankingScoreDifference cannot be computed. As the RankingScoreDifference feature is of type Double some value needs to be returned in these cases. 0.0 is returned in these cases which should not harm the performance of the classifier as events that actually have a RankingScoreDifference of 0.0 do not provide any information about their ordering as well. The classifier can be expected to learn that 0.0 is not a predictive value since it gets training instances with both outcome types `BEFORE` and `AFTER` that have a RankingScoreDifference feature 0.0.

- In the `TemporalLinkEventToSubordinatedEventAnnotator` two types of extractors are used: extractors that work on each event separately (event, the tense, aspect, class and syntactic first child of grandparent of leaf) and extractors that take both events as input (syntactic constituency tree from one event to the other, tokens between the two events). The `RankingScoreDifferenceExtractor` is added as part of this second set of extractors.

3.3.4 Feature Scaling

Other than the RankingScoreDifference, the original ClearTK features are not numerical. ClearTK uses the "LIBLINEAR" classifier for which the features are turned into numerical binary features (for example the tense feature is splitted up into features called `PAST`, `PRESENT`, `PRESPART` etc. of which one gets passed to the classifier with value 1. The RankingScoreDifference on the other hand is continuous with almost all values in (-0.1, 0.1). It is undesirable to have features in extremely different ranges since this gives some features more influence on the outcome than others. For this reason, features are often normalized such that they are in the same range. In order to properly normalize the features one needs to know the maximum and minimum value for the whole data set. Since the ClearTK pipeline always processes one document at a time we chose a simpler workaround: multiplying the features with a fixed value that is found such that the results are best. We try scaling factors of 5, 15 and 30 such that most values are in (-0.5, 0.5), (-1.5, 1.5) and (-2, 2) respectively.

3.3.5 Bucket Feature

The classifier used is a linear classifier, however the relationship of the RankingScoreDifference and temporal ordering might not be linear. One needed to search through the space of transformations to find the right function. However, another possibility is discretizing the feature to indicate whether the value lies in a certain interval.

We discretize the RankingScoreDifference feature as follows:

$$\begin{aligned}
&\text{if } RankingScoreDifference < -\delta, && \text{return feature-value 0}\\
&\text{if } RankingScoreDifference \in [-\delta, \delta], && \text{return feature-value 1}\\
&\text{if } RankingScoreDifference > \delta, && \text{return feature-value 2}\\
&&& \text{for some } \delta \in \mathbb{R}^+
\end{aligned}$$

The three different values of this feature correspond to the following three possibilities concerning events A and B: A succeeds B, A precedes B, or the difference is too small to make a prediction.

The bucket feature is implemented by a new feature extractor called `BucketRankingScoreDifferenceExtractor`. The input of this extractor are two events and a delta-value, it calculates the difference, and returns either 0, 1, or 2 as the feature-value. Several `BucketRankingScoreDifferenceExtractors` can be included in `TemporalLinkEventToSubordinatedEventAnnotator` in order to combine features with different δ-values.

4 Experiments

4.1 Data

TempEval-3 uses several corpora for training and testing the models: TimeBank and AQUAINT are two corpora that include 183 and 73 news documents respectively. These data sets are annotated according to the TimeML specifications by humans which means they are gold standard. A corpus of 20 files is used as test data, which is annotated by human experts (platinum standard)[1].

As discussed in 3.2 not all links in the data receive a ranking score by the Commonsense Classifier. There are 11098 temporal links in the TimeBank and AQUAINT data sets in total. 5826 of these are event-time links and 5272 event-event links. 2231 of the event-event links do not have the relation type `BEFORE` or `AFTER`. In 2302 of the event-event links at least one event is not a verb. After excluding links that do not match these three criteria (several reasons may apply to one event) 1895 links are used as input for the Commonsense Classifier which is about 17% of all links.

4.2 Experimental Setup

For the purpose of this project the TimeBank and AQUAINT data is split into a training set and a development set such that approximately 90% of the temporal links are in the training set and 10% in the development set. Instead of evaluating on the test set the development set should be used for evaluating the model at

[1] All TempEval-3 data sets can be downloaded here: `http://www.cs.york.ac.uk/semeval-2013/task1/index.php?id=data`

intermediary stages. A test set always needs to remain unseen as long as adjustments might be made to the model. If the model was evaluated on the test set and was then adjusted to improve its performance, overfitting could occur. Overfitting is the case if a model fits the data it is evaluated on so perfectly that it is adjusted to only this set of data and performs worse on new, unseen data. To prevent overfitting one evaluates on a development set when still doing adjustments to the model. We first conduct experiments on the development set, leaving the possibility open to improve the model, then as a final step the system is evaluated on the test set. This ensures that the final results are a good indication for how the model performs on new, unseen data.

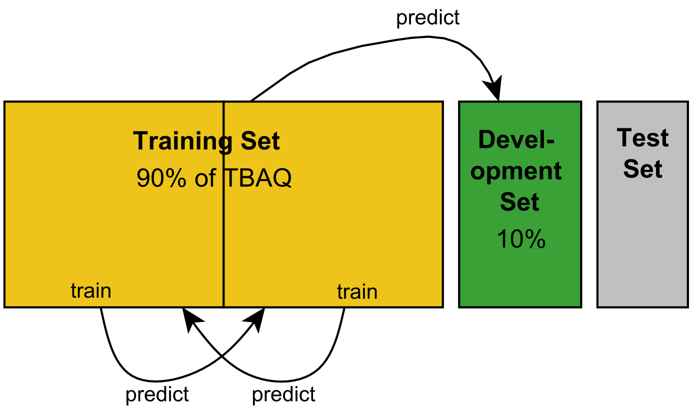

Figure 4.1: Division of data in training set, development set and test set.

There are multiple features that can be used based on the RankingScoreDifference: The RankingScoreDifference itself (with diffferent scaling factors), any combination of bucket features with different δ-values, or a combination of the RankingScoreDifference and bucket features. In order to find the most predictive of these features, 2-fold cross-validation is performed on the training set. In 2-fold cross-validation the data is split into two parts. First, the model is trained on the first part in order to predict annotations for the second part. Then, it is trained on the second part and predicts for the first part. This process is repeated several times for different combinations of features and different values for δ. With this method we determine the most predictive set of features. These features are then used to run experiments using the whole training set for training and the development set to evaluate. With these experiments the performance of the original ClearTK setup is compared to the performance of adding knowledge from the Commonsense Classifier to ClearTK. The experiments for finding the most predictive features cannot be evaluated on the development set for the same reason for which the overall experiment cannot be evaluated on the test set: overfitting. If parameters are tuned while evaluating on the development set the model will fit very well to the development set but not necessarily to new data. For this reason, first cross-validation is performed to find the best features, then the system is evaluated on the development set. In the end, when no other improvements will be done anymore, the system can be evaluated on the test set.

4.3 Temporal Links Used in the Experiments

In TempEval-3, Task C (and Task C Relation Only) entail the identification of temporal links between:

1. Event-event (same sentence)
2. Main events of consecutive sentences
3. Event-time (same sentence)
4. Event-document creation time

This thesis is concerned with temporal ordering of events. Thus, only the links between two events (type (1) and (2)) are considered, since the Commonsense Classifier is only designed for events, not time expressions. We take the subset of those links with relation type `BEFORE` or `AFTER` in order to focus on temporal ordering and call this set of links **event-event-temporal-ordering**.

Bethard (2013) only included annotators for (1), (3) and (4) when participating in ClearTK because accuracy dropped when including the annotator for (2). For (1) he chose to only include event pairs where one event is in the main-clause and one in the subordinate clause of the sentence because previous research showed that temporal order can be predicted very accurately for these kinds of event pairs (Bethard, Martin, & Klingenstein, 2007). These type (1) links with the subordinate sentence structure will be reffered to as **subordinate links** in the following.

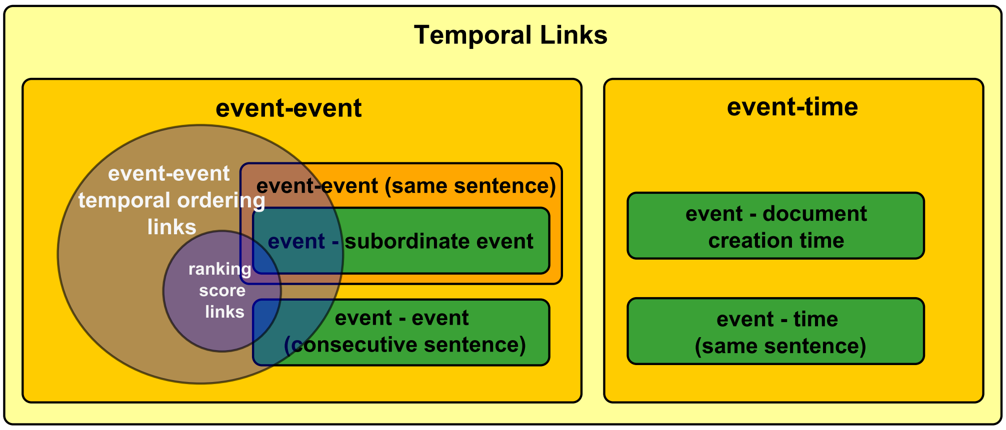

Figure 4.2: Temporal Links to be annotated in TempEval-3.

4.4 Cross-validation Experiments

The aim of the cross-validation experiments is finding the best features using the output of the Commonsense Classifier. Since the RankingScoreDifference as well as the bucket features are based on the ranking scores produced by the Commonsense Classifier, they will be referred to as *ranking score features* in the following. The cross-validation is performed using the RankingScoreDifference with different scaling factors, bucket features with different δ-values, and combinations of several bucket features with different values. In a subsequent experiment, the combination of bucket features with the best performing δ-values and the RankingScoreDifference with the best performing scaling factor are used as combined features to see whether this further increases performance.

The cross-validation experiments are conducted with the *event-event-temporal-ordering* links.

4.5 Experiments Evaluating on Development Set

When evaluating on the development set, the original features (referred to as *orig* in the results) by themselves are compared to the features based on ranking scores (using the best performing values from cross-validation) and the combination of original features and ranking score features.

A) Event-event-temporal-ordering links: This is the temporal ordering of events task on which the classifier is mainly evaluated.
→ Training, predicting and evaluating on event-event-temporal-ordering links.

B) Subordinate links: We also compare to the original ClearTK setup, which uses only subordinate links that are in the same sentence (type (1)). The ClearTK annotators for type (3) and (4) are not included in these experiments as the Commonsense Classifier is not designed to process time expressions. However, these experiments are still evaluated against all links (1-4) as this is the standard TempEval-3 evaluation. Thus, experiment (A) sheds light on the question whether prediction for those links that the Commonsense Classifier has information about improves when using its features. Experiment (B), on the other hand, shows how the system performs on the official TempEval-3 task (although only annotating 1 out of 4 link types).
→ Subordinate links used for training and predicting, evaluated against all links.

C) All event-event links: As experiment (B) is not really fair (annotating only 1 link type but being evaluated with 4 types) towards the system this experiment annotates link types (1) and (2), the two event-event link types. For this purpose we created a new annotator (for Task C Relation Only) which predicts a relation type for every temporal link between two events. However, one of the original ClearTK features, the path between the two events in the syntactic parse tree, cannot be extracted for events that are not in the same sentence. For this reason two cases are distinguished: if two events are in the same sentence all original features are used. If the events are in different sentences all features except for the syntactic path are used. This adaptation of the original features is referred to as *orig** in the following.
→ Event-event links used for training and predicting, evaluated against all links.

4.6 Experiments Evaluating on Test Set

A) Event-event-temporal-ordering links: The classifier is evaluated on the test set with the event-event temporal ordering task.

→ Training, predicting and evaluating on event-event-temporal-ordering links.

4.7 Analysis of Results

For the predictions, ClearTK produces files in the TimeML format (.tml) that contain the annotations predicted by the system. These are evaluated using the TempEval-3 Evaluation Toolkit[2] which compares the system annotations to the gold standard annotations (or *reference* annotations).

4.7.1 Precision & Recall

In machine learning, the measures *precision* and *recall* are usually used to evaluate the performance of a classifier. Precision (P) is a measure for the correctness of the predictions made by the system:

$$P = \frac{\#\text{ correct predictions}}{\#\text{ all system predictions}} \tag{4.1}$$

Recall (R) is a measure for how many of the reference annotations were found by the system:

$$R = \frac{\#\text{ correct predictions}}{\#\text{ reference annotations}} \tag{4.2}$$

In order to get a combined measure of P and R, the F1 score is used, which is the harmonic mean of P and R.

$$\text{F1} = 2 * \frac{P * R}{P + R} \tag{4.3}$$

The harmonic mean has desirable properties for the evaluation of machine learning classifiers other than for example the average: For example, a classifier with $P =$

[2] `http://www.cs.york.ac.uk/semeval-2013/task1/index.php?id=data`

0.5 and $R = 0.5$ should get a better score than a classifier with $P = 0.99$ and $R = 0.01$ because the latter would find almost none of the gold annotations. While the average is 0.5 for both cases, the harmonic mean of the former is 0.5 and only 0.0198 in the latter case.

4.7.2 Official TempEval-3 Evaluation

As temporal ordering has the property of transitivity, the organizers of TempEval adjusted the definitions of precision and recall to provide a standard for the evaluation of temporal ordering (UzZaman & Allen, 2011): Transitivity means that if $A \prec B$ and $B \prec C$, then $A \prec C$[3]. This derivation of a new relation is also called temporal closure. Suppose graph K in figure 4.3 is the reference annotation graph and S_1, S_2, and S_3 are annotations by three different systems. The bold edges are annotated temporal links and the dotted edges are derived from transitivity. S_1 and S_2 make two predictions matching with the reference annotations, S_3 makes one explicitly matching prediction and one prediction that is implicitly correct (matching the relation derived from transitivity). Therefore, precision in all three cases is: $P = \frac{2}{2} = 1$ (that a prediction is not explicitly stated does not make it a wrong prediction). Now, for recall it matters whether all explicitly annotated relations are found or not because if a relation was not explicitly annotated it is deemed less important. Therefore, the implicitly correct prediction is not counted towards recall: for S_1 and S_3, $R = \frac{2}{3}$; for S_2, $R = \frac{1}{3}$.

In general, precision and recall are defined as follows for TempEval-3 (Uzzaman et al., 2013):

$$P = \frac{|Sys^- \cap Ref^+|}{|Sys^-|} \tag{4.4}$$

$$R = \frac{|Sys^+ \cap Ref^-|}{|Ref^-|} \tag{4.5}$$

where Sys is the graph given by the system relations and Ref is the graph given by the reference relations. G^+ is the closure of graph G (added derived relations) and G^- is the reduced of graph G (redundant relations removed).

Precision is calculated as the number of relations in the reduced graph Sys^-

[3]Notation: '$A \prec B$' means A precedes B

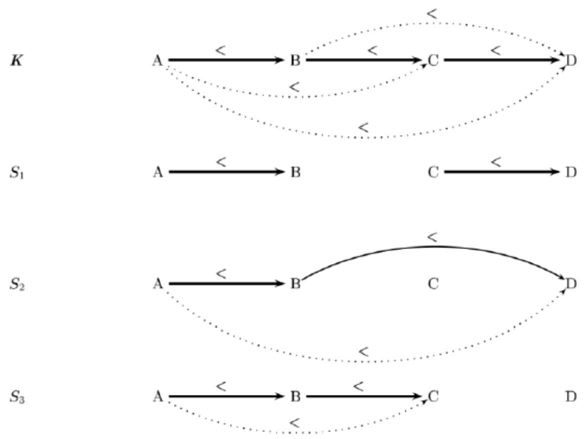

Figure 4.3: Example temporal relations: K is the reference graph, S1, S2, and S3 are predictions by three systems. Retrieved from (UzZaman & Allen, 2011).

that are correct according to the closure graph Ref^+, divided by the number of relations in Sys^-, which means implicitly correct relations through transitivity are taken into account. Recall is calculated as the number of relations in the closure system graph Sys^+ that are correct according to the reduced reference graph Ref^-, divided by the number of relations in Ref^- which means only finding explicitly annotated reference relations counts towards recall.

Using these modified definitions of precision and recall, the F_1 score is calculated by means of formula 4.3.

4.8 Results

4.8.1 Cross-validation Experiments

Table 4.1 shows that bucket features and scaled RankingScoreDifference features perform better than the pure RankingScoreDifference feature. Among the bucket features the delta values {0.001, 0.01, 0.01, 0.5, 1}, {0.001, 0.01, 0.1} and {0.01} perform best in F1. This result suggests that the combination of several bucket features does not add important information and the single feature with $\delta = 0.01$ should be preferred following the notion of Occam's razor. This principle states that one should always prefer a simple model over a complex one if they perform

Links used: event-event-temporal-ordering links

Features	Parameter	P	R	F1
bucket0	0	77.144	77.478	77.311
bucket6	δ= 0.01	78.549	78.789	78.669
bucket1	δ = 0.05	77.489	78.018	77.753
bucket2	δ = 0.1	73.662	74.277	73.968
bucket4	δ = 0.5	72.364	73.081	72.721
bucket7	δ = 0.001, 0.01, 0.1	78.549	78.789	78.669
bucket3	δ = 0.001, 0.01, 0.1, 0.5, 1	78.549	78.789	78.669
rank1	scaling factor = 1	73.013	73.660	73.335
rank5	scaling factor = 5	78.411	78.943	78.676
rank15	scaling factor = 15	78.876	79.329	79.102
rank30	scaling factor = 30	78.876	79.252	79.064
rank15+bucket0	scaling factor = 15, δ = 0	79.256	79.715	79.484
rank15+bucket6	scaling factor = 15, δ = 0.01	78.976	79.290	79.133
rank15+bucket1	scaling factor = 15, δ = 0.05	79.294	79.676	79.485
orig		79.605	79.753	79.679
orig+rank1		81.204	81.373	81.288
orig+rank5		83.476	83.610	83.543
orig+rank15		83.126	83.263	83.194
orig+rank30		83.009	83.147	83.078

Table 4.1: Cross-validation, Training and evaluating on: event-event-temporal-ordering links. (P, R, and F1 in %)

equally well. This is not just an intuition but proves useful in machine learning because complex models tend more often to overfitting: modelling random noise in the training data, leading to a good performance on the training data but worse performance on new, unseen data.

For the RankingScoreDifference features, different scaling factors are tested in combination with the original features, such that it is tested whether assimilating the ranges of the different features improves performance. In these experiments, a scaling factor of 5 yields the best results. When tested as a single feature the RankingScoreDifference with scaling factor 15 leads to a better performance. These differences are very small and might just as well be due to chance. The scaling factor of 15 is chosen for the subsequent experiments. In order to find the best set of parameters based on the ranking scores RankingScoreDifference with factor 15 is combined with the best performing bucket features. The differences between these are again very small. A delta value of 0 performs best together with $\delta = 0.05$. We again prefer the simpler model over a complex one and choose $\delta = 0$ for subsequent experiments. Note that the bucket feature with $\delta = 0$ is equivalent to taking the sign of the RankingScoreDifference or with other words taking the sign as a feature is a special case of the bucket feature.

4.8.2 Experiments Evaluating on Development Set

A) Event-event-temporal-order links: Table 4.2 shows that the features based on ranking scores perform similar or slightly worse than the original ClearTK features. However, when combined with the original features performance improves

A) Links used: event-event-temporal-ordering links

Features	Parameter	P	R	F1
orig		81.818	82.129	81.973
rank15	scaling factor = 15	80.228	80.608	80.418
rank15+bucket0	scaling factor = 15, $\delta = 0$	79.468	79.848	79.657
orig+rank15	scaling factor = 15	85.227	85.551	85.389
orig+rank15+bucket0	scaling factor = 15, $\delta = 0$	84.470	84.791	84.630

Table 4.2: Development set. Training, predicting, and evaluation on event-event-temporal-ordering links. (P, R, and F1 in %)

from $F_1 \approx 82.0$ to about 85.4.

B) Subordinate links: The recall values are very low in general in this experiment. This is due to the fact mentioned in section 4.3, that only subordinate links are annotated while the system is evaluated agains all link types (1-4) according to the official TempEval-3 evaluation. The ranking score features show clearly higher precision, but lower F_1 because of lower recall values than the original features. When combining the features, precision takes an intermediate value, higher recall than any of the two feature by itself and in consequence a higher F_1 score than both features by themselves. It seems counterintuitive that precision decreases with additional features. However, it can happen that a classifier does not manage to take each feature as much into account as it is predictive, e.g. by giving some features too much weight. Comparing the combined features to the original features P, R and F_1 are all higher with the combined features.

C) Event-event links: As expected, recall in general increases since this experiment annotates two out of four link types. Similar to experiment B, precision is very high for the ranking score features, but recall is low, resulting in low F_1. The combination of features performs better than only the original features.

	Links used	Features	Param.	P	R	F1
B)	subordinate	orig		70.000	5.120	9.542
	links	rank15+bucket0	$\delta = 0$	84.615	4.542	8.621
		orig+rank15+bucket0	$\delta = 0$	76.289	6.028	11.173
C)	event-event	orig*		71.498	12.056	20.633
	links	rank15+bucket0	$\delta = 0$	88.430	8.836	16.066
		orig*+rank15+bucket0	$\delta = 0$	74.009	13.708	23.131

Table 4.3: Development set. Training on and predicting subordinate links/event-event links. Evaluating on all links. (P, R, and F1 in %)

4.8.3 Experiments Evaluating on Test Set

Features	δ-values	P	R	F1
orig		67.519	67.602	67.561
rank15	scaling factor = 15	61.990	61.480	61.734
rank15+bucket0	scaling factor = 15, delta = 0	62.595	61.990	62.291
orig+rank15	scaling factor = 15	67.176	66.582	66.877
orig+rank15+bucket0	scaling factor = 15, delta = 0	67.176	66.582	66.877

Table 4.4: Test set. Training, predicting, and evaluating on: event-event-temporal-ordering links. (P, R, and F1 in %)

Overall performance is lower on the test set than with the development set for all features. Other than in the development-set experiments, combining the original features with the ranking score features does not improve performance. The original features show the best performance. The ranking score features perform worse. In combination the values are very similar to those of the original features.

5 Discussion

5.1 Cross-validation Experiments

In 3.3.3 we hypothesized that the distance of two events on the timeline given by their ranking scores gives a measure of confidence of the respective ordering. However, this effect, if existent, is very small: simply using the sign of the RankingScoreDifference (bucket feature with $\delta = 0$) yields $F_1 \approx 77.3$ while using the RankingScoreDifference with the best scaling factor (15) yields $F_1 \approx 79.1$.

Scaling the RankingScoreDifference and using it in combination with a bucket feature proved beneficial compared to the pure RankingScoreDifference: $F_1 \approx 73.3$ with the unchanged RankingScoreDifference, $F_1 \approx 78.7$ with a bucket feature with $\delta = 0.01$, and $F_1 \approx 79.0$ for the RankingScoreDifference multiplied by 15.

5.2 Experiments Evaluating on Development Set

Adding the features derived from the output of the Commonsense Classifier to the original ClearTK features improved both precision and recall in all three experiments (A, B and C). However, the difference is not very large (e.g. improvement from $F_1 \approx 82.0$ to $F_1 \approx 85.4$ in experiment A). On the other hand, with such high precision and recall values which already the original features show in experiment A, even a small improvement is worthwhile since the links predicted wrongly are increasingly difficult when approaching 100%. Even human annotators do not reach 100%. 'Inter-annotator agreement' measures to what extent annotations of several human annotators match. Inter-annotator agreement for Task C (Relation Only) is not known but for Task A and B (on the test set) it is: 87% (for event annotations), 92% (event class), 87% (time expressions), and 88% (time values)

(Uzzaman et al., 2013).

It is worth noting, that the ranking score features by themselves reach very similar values as the original features (e.g. in experiment A) although the two sets of features are conceptually completely different: while the original features use syntactic and semantic properties of the events among others the ranking scores are solely based on learned script knowledge. The Commonsense Classifier does not make use of any clues in the context such as "then" and "after", it does not make use of the tense of the verbs etc. The fact that this is such a different approach suggests that the information gained from the Commonsense Classifier, the ranking score features might be predictive for different links than ClearTK's linguistic features. This hypothesis is supported by the results since on experiments A-C recall for the combined features was higher than for any of the two features by itself, although this effect is not very large. Bethard excluded ClearTK's annotator for links of type 2 from his participation in TempEval-2013 because it did not perform well in some tests and in TempEval-2007 (personal communication, April 15, 2014). Since he preferred high precision over recall he only annotates subordinate event links (and the time links, type 3 and 4). The ranking score features might be able to fill this gap. ClearTK's linguistic features work best on same-sentence event pairs. For script knowledge, on the other hand, it should not make much of a difference whether two events are in the same or in different sentences. If this hypothesis was true it should be visible from comparing experiments B and C. Although precision in C is much higher than in B for the ranking score features other than for the original features, this is not the case for recall. The difference between B and C in F_1 is even higher for the original features than the ranking score features. It is surprising that not only recall improves when also annotating links of type 2 but even precision improves slightly with the original features. This would mean that there is no reason to exclude the annotator for links of type 2 from the participation in TempEval. However, the results in experiment C were obtained using our own annotator as mentioned in section 4.5 under (C). While Bethards annotator for these links only uses the event tense, aspect and class as features, this annotator uses all features by the EventToSubordinatedEventAnnotator except for the syntactic path. This difference in features could explain the diverging results.

5.3 Experiments Evaluating on Test Set

The fact that the overall performance drops when evaluating on the test set compared to the development set for all features is not suprising. The test set was generated apart from the already existing TimeBank and AQUAINT data sets which might lead to differences between the data sets. Comparing the features, the ranking score features perform worse than the original features. Combining the two leads to similar values as the original features which probably means that the classifier did not take the less predictive ranking score features into account.

However, the system in the current state uses a relatively naive approach: for the Commonsense Classifier to really learn 'commonsense' script knowledge it needs to be trained on very large data sets. Here, we simply trained it on the same data, that was used to train ClearTK. This data set contains about 10,000 links of which only less than 3,000 could be used by the Commonsense Classifier due to the constraints (event-event links, before/after, both events are verbs).

5.4 Future Research

In future research, the Commonsense Classifier should be trained on a large data set in order to learn more reliable script knowledge. The approach explained in section 2.3 can be used for this purpose, where a rule-based classifier is used to create training data from unannotated text. After the data is annotated, the Commonsense Classifier can be trained on it using supervised learning.

As mentioned in 4.1 only about 17% of the temporal links in the data receive a ranking score (event-event links with before/after relation where both events are verbs). It could be worthwile to test whether including all links and using an annotator that annotates all relation types leads to better recall values on the TempEval-3 task. It is conceivable that the ranking scores are predictive for some more relation types than just `BEFORE` and `AFTER` (for example `IDENTITY`). Furthermore, one could modify the Commonsense Classifier to use events of any part-of-speech, not just verbs. The definition of what the event's arguments are would have to be redefined in this case. With these two changes, ranking scores could be learned for many more links in the data which can be expected to lead to better recall values when evaluating on all temporal links according to the TempEval-3 task.

6 Conclusion

Script knowledge has been shown to be beneficial for a number of natural language understanding applications, such as question answering and document summarization Miikkulainen (1995). In this thesis, it was tested whether a temporal ordering classifier can benefit from commonsense script knowledge.

The experiments do not provide strong evidence to affirm the hypothesis that using commonsense script knowledge additional to linguistic features leads to higher performance than only using linguistic features for temporal ordering of events. However, the results suggest that this is a promising approach to further pursue for two reasons: Firstly, using commonsense script knowledge without any linguistic features shows similar performance to linguistic features which shows that this information is predictive. Secondly, commonsense script knowledge might be predictive for different kinds of temporal links than linguistic features since it does not rely on events being in the same sentence. One should try to train the Commonsense Classifier on a much larger data set in order to learn more reliable script knowledge.

References

Abelson, R., & Schank, R. (1977). *Scripts, plans, goals and understanding.* Lawrence Erlbaum Associates, Potomac, Maryland.

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, *3*, 1137–1155.

Bethard, S. (2013). ClearTK-TimeML: A minimalist approach to TempEval 2013. *Second Joint Conference on Lexical and Computational Semantics (* SEM)*, *2*, 10–14.

Bethard, S., Martin, J. H., & Klingenstein, S. (2007). Finding Temporal Structure in Text: Machine Learning of Syntactic Temporal Relations. *International Journal of Semantic Computing (IJSC)*, *1*(04), 441– 458.

Chambers, N. (2013). NavyTime : Event and Time Ordering from Raw Text. *Second Joint Conference on Lexical and Computational Semantics (* SEM)*, *2*, 73–77.

Ferrucci, D., & Lally, A. (2004, September). UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, *10*(3-4), 327–348.

Frege, G. (1892). Über Sinn und Bedeutung. *Zeitschrift für Philosophie und philosophische Kritik*, *100*, 25–50.

Hinton, G. (1986). Learning Distributed Representations of Concepts. *Proceedings of the Eigth Annual Conference of the Cognitive Science Society*, 1–12.

Kolomiyets, O., & Moens, M. (2013). KUL: A data-driven approach to temporal parsing of documents. *Proceedings of the second joint conference on lexical and computational semantics (* SEM)*, *2*, 83–87.

Kolya, A. K., Kundu, A., & Gupta, R. (2013). JU_CSE : A CRF Based Approach to Annotation of Temporal Expression, Event and Temporal Relations. *Pro-*

ceedings of the second joint conference on lexical and computational semantics (SEM)*, *2*, 64–72.

Laokulrat, N., Miwa, M., Tsuruoka, Y., & Chikayama, T. (2013). *UTTime: Temporal Relation Classification using Deep Syntactic Features.*

Marneffe, M. D., MacCartney, B., & Manning, C. (2006). Generating typed dependency parses from phrase structure parses. *Proceedings of LREC*.

Marneffe, M. D., & Manning, C. (2008). Stanford typed dependencies manual.

Miikkulainen, R. (1995). Script-Based Inference and Memory Retrieval in Subsymbolic Story Processing. *Applied Intelligence*, *5*(2), 137–163.

Mitchell, J., & Lapata, M. (2010, November). Composition in distributional models of semantics. *Cognitive science*, *34*(8), 1388–429.

Modi, A., & Titov, I. (2014a). Learning Semantic Script Knowledge with Event Embeddings. *International Conference on Learning Representations (ICLR), 2014*.

Modi, A., & Titov, I. (2014b). Learning Semantic Script Knowledge with Event Embeddings. *Conference on Computational Natural Language Learning (CoNLL), 2014*.

Pustejovsky, J., Ingria, B., & Sauri, R. (2005). The specification language TimeML. *The language of time: A reader*, 545–557.

Regneri, M., Koller, A., & Pinkal, M. (2010). Learning script knowledge with web experiments. *Proceedings of ACL*(July), 979–988.

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Turian, J., Ratinov, L., & Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics.*(July), 384–394.

UzZaman, N., & Allen, J. (2011). Temporal evaluation. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers*, 351–356.

Uzzaman, N., Llorens, H., Derczynski, L., Verhagen, M., Allen, J., & Pustejovsky,

J. (2013). SemEval-2013 Task 1: TempEval-3: Evaluating Time Expressions, Events, and Temporal Relations. *Second joint conference on lexical and computational semantics (* SEM)*, *2*, 1–9.

Appendices

Appendix A: Example Input for Commonsense Classifier


```xml
<?xml version="1.0" encoding="UTF-8"?>
<scripts>
  <script docID="ABC19980120.1830.0957" linkID="0"
  relationType="BEFORE">
    <item aspect="NONE" eventID="e51" lemma="look" ner="O"
     polarity="POS" pos="VERB" sentenceID="0" tense="PRESPART"
     text="looking" tokenID="2">
      <ptcp sentence="Those observers looking for a battle
       between uncompromising representatives and very different
       ideologies will, in all likelihood, be disappointed."/>
      <ptcp lemma="battle" ner="O" pos="NN" sentenceID="0"
      syntDep="NN" text="battle" tokenID="5"/>
      <ptcp lemma="ideology" ner="O" pos="NNS" sentenceID="0"
      syntDep="NNS" text="ideologies" tokenID="12"/>
    </item>
    <item aspect="NONE" eventID="e53" lemma="disappoint" ner="O"
    polarity="POS" pos="VERB" sentenceID="0" tense="FUTURE"
    text="disappointed" tokenID="20">
      <ptcp sentence="Those observers looking for a battle
       between uncompromising representatives and very different
       ideologies will, in all likelihood, be disappointed."/>
      <ptcp lemma="observer" ner="O" pos="NNS" sentenceID="0"
      syntDep="NNS" text="observers" tokenID="1"/>
      <ptcp lemma="will" ner="O" pos="MD" sentenceID="0"
      syntDep="MD" text="will" tokenID="13"/>
      <ptcp lemma="likelihood" ner="O" pos="NN" sentenceID="0"
      syntDep="NN" text="likelihood" tokenID="17"/>
      <ptcp lemma="be" ner="O" pos="VB" sentenceID="0"
      syntDep="VB" text="be" tokenID="19"/>
    </item>
  </script>
</scripts>
```