Natural Language Generation of Tense and Aspect in a Narrative Context

Daan R. Henselmans

September 29, 2012

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Declaration

I hereby confirm that the thesis presented here is my own work, with all assistance acknowledged.

Daan R. Henselmans;

Msida, Malta, September 29, 2012.

Natural Language Generation of Tense and Aspect in a Narrative Context

M.Sc. Thesis of Daan R. Henselmans

September 29, 2012

Erasmus Mundus European Masters Program in Language and Communication Technologies

Universität des Saarlandes
University of Malta

Supervisors

Dr. Albert Gatt
University of Malta

Prof. Stephan Busemann
Universität des Saarlandes

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> *"In those days, in those far remote days, in those nights, in those faraway nights, in those years, in those far remote years, at that time the wise one who knew how to speak in elaborate words lived in the Land; Šuruppag, the wise one, who knew how to speak with elaborate words, lived in the Land."*
>
> —Instructions of Šurrupag, 3rd millennium BC

The art of storytelling is as old as humanity itself. Long before people sailed ships, played the fiddle, or sent satellites into orbit, they were telling each other stories. Oral traditions as well as the earliest known written texts revolve around narrative passages, times, events, and developments. Even today, the bulk of the written text that humanity produces everyday still comes in the form of a narrative; from postmodern literature to glossy magazines, hospital reports, and text messages, texts frequently involve temporal progression. Not a single culture on Earth does not have the concept of stories.

It is therefore ironic that time presents one of the major challenges in natural language processing. In a field which is almost as old as the digital computer itself, has branched into dozens of subfields, and is applied in all sorts of technology all over the world, both understanding and generation of time are still in an unruly stage of development, in spite of significant progress over the past couple of decades. Temporal expressions in language are subject to many subtle distinctions and a wide range of semantic implications. Many existing language processing systems involve time in some form or other, but narratives remain a very difficult subject.

This thesis aims to contribute to the field of natural language generation by investigating a framework whereby the generation of tense and aspect in narrative text can be approached empirically, in a machine learning setting. It addresses the following question: given information about the non-linguistic temporal context in which an event occurs, as well as about various semantic and grammatical parameters related to the way that event needs to be expressed linguistically, is it possible to accurately predict its tense and aspect properties?

Natural language generation systems are traditionally used to generate textual representations of non-linguistic data, such as weather reports from meteorological measurements. In this research, the data are obtained by extracting

event information from the TimeBank corpus, a corpus which consists of newspaper articles annotated with time line information, like the features of events and times described in the text, and the temporal relations between these times and events. The TimeBank corpus has found use in many natural language understanding applications, but it also has interesting, hitherto unexplored possibilities for generation.

The algorithm which is defined in this thesis is designed to automatically generate tense and aspect in newspaper-like articles, based on corpus information. The input of the algorithm is an event representation, with a number of features extracted from the TimeBank corpus, which contain information on the event, like its event class, lemma, relation to the speech time, and relation to other events in the text. Each event representation also contains the tense and aspect the event has in the corpus. A decision tree is built using the J48 decision tree classifier in the WEKA data mining workbank to determine the most appropriate tense and aspect in certain contexts. The used corpus information functions as a parallel to the information which would be available in a hypothetical NLG system of newspaper-like texts, in which this algorithm could be implemented. Obtaining event and time information for generation is feasible in text-to-text generation, which is used for automatic summarization or question-answering systems.

The thesis has the following aims and objectives:

- To find and extract relevant pieces information for generation of tense and aspect in the TimeBank corpus, and represent them as features.
- To identify the most useful features to perform generation of tense and aspect.
- To develop and optimize an algorithm which performs tense and aspect generation.

This dissertation is divided up into seven chapters. The following two chapters provide a thorough review of the field of natural language generation and the role of time and narrative in language respectively. The next chapter describes the methods used to develop the algorithm and conduct the experiment. It is followed by the results of the developed machine learning framework, along with an in-depth discussion and some concluding remarks.

Chapter 2

Natural Language Generation

Natural language generation (NLG) is a fast-moving subfield of computational linguistics and natural language processing (NLP). It is concerned with computer systems that are capable of representing non-linguistic input as a text in English or any other language understandable by humans. An NLG system usually takes abstract, non-linguistic information as its input, and uses knowledge about the language and the domain it functions in to generate reports, explanations, and other textual renderings of data.

NLG is a field with many direct applications and fascinating research opportunities, posing questions and giving insights related to artificial intelligence, cognitive science, and general linguistics. Some of the questions which arise are what constitutes a good text, how to select appropriate information in a given domain, and what steps are needed to get from abstract information to a text which represents it. To answer these questions, insights from different fields can be used: NLG systems apply linguistic knowledge to determine which type of expression is best in a given context, but also use methods from the field of artificial intelligence, like planning techniques and production systems, and increasingly involve statistical and machine learning methods (Reiter & Dale, 2000).

The applications of NLG include systems which automate routine writing in a certain domain fully or partially, and systems which represent raw data in an easily understandable format for people without the time or knowledge to interpret the raw data themselves. Many other NLP applications use NLG modules, including document summarization, machine translation, and dialogue systems. Ultimately, NLG ought to pave the road for fully language-sensitive human-computer interaction, allowing sharing more in-depth information and understanding on both parts.

NLG can be said to be a parallel field to Natural Language Understanding (NLU), another subfield of NLP. Whereas NLG is concerned with generating texts in human language, NLU systems take texts as their input, and extract abstract information which can be used for computing. Both of the subfields revolve around computational models of natural language, and make use of the same theoretical foundation. Since NLG converts abstract data into text, and NLU converts texts into abstract data, they can be said to have the same 'end points', with one functioning as a 'reverse' of the other. In terms of implementation, however, they are quite different; problems in NLG tend not to be problems in

NLU and vice versa. Methods used in either type of system cannot often be used in reverse to solve a parallel problem, even if the end product of an NLG system is the input of an NLU system and vice versa, and they can as such be said to be parallel in their functions (Reiter & Dale, 2000). For example, a major problem in NLG is ensuring that the produced text is easy to understand by humans, which has no functional parallel in NLU. On the other hand, a major challenge in NLU systems is representing ambiguous references in texts using their correct interpretation, whereas generating unambiguous text is, though far from trivial, relatively less difficult in a closed domain. Theoretically, a 'bidirectional grammar' which can be used to transform text into data and vice versa is possible, but it would perform tasks in a roundabout way which is not very feasible using current NLP methods (Reiter & Dale, 2000; Reiter, 2010).

NLG, as opposed to most other subfields of NLP, is characterized by the importance of choice-making (Reiter, 2010). NLG systems are required to take abstract data, and make numerous decisions at every level on how to best represent that information in a text. This ranges from high-level choices, like what information to include in a text at all, to low-level choices, like how to inflect a verb in a given context. These choices can be motivated by any number of factors, but they have to be made in an NLG system even when there is no single 'correct' choice (Reiter & Dale, 2000; Reiter, 2010). For example, a system might have to choose between two different sentences based on their linguistic correctness, as in example 2.1.

Example 2.1

1. *"John bounced down the stairway."*
2. * *"John bounce down the stairway."*

In this case, one of the two options is clearly better, since the second is grammatically incorrect. There is a much larger variation in types of decisions an NLG system has to make, though: whereas the decision is straightforward in the previous case, an NLG system might just as well have to make the choice between the two sentences in example 2.2.

Example 2.2

1. *"The lady strutted down the street."*
2. *"The lady shuffled down Main Street."*

1. *"Tesla is said to have invented a death ray shortly before his death."*
2. *"A death ray is said to have been invented by Tesla shortly before his death."*

In both examples, the two options are semantically closely related, and determining which sentence to choose is not a matter of grammar. There is a number of conceivable situations where both choices are, in fact, valid. The

first pair of sentences differ in terms of lexical choice, offering the same information but with different connotations, and the second pair differ in whether active or passive voice is used — they offer the same information, but lie focus on particular implications of the sentence. Determining the more easily readable or appropriate option requires taking into account human psychology and the effect different words and phrases will have on the reader. For both examples, selecting either option in an NLG system would be a subtle choice between different connotations and styles, and different results might be appropriate in different contexts.

Without a strictly 'correct' target, the success of NLG systems in its goals is more difficult to measure than that of most other types of NLP. NLG systems normally aim to produce text as close to texts written by humans in the same area as possible, and evaluation is usually performed in one of three ways. *Metric-based corpus evaluation* statistically compares the generated text to corpus text in the same domain, to see how close they are to one another, but this is not necessarily a measure of the quality of the generated text. Alternatives which provide more applicable measures at the cost of being more time-consuming are *Extrinsic evaluation*, which determines how effective generating texts were in accomplishing a certain goal when read, and *non-task-based human evaluation*, which involves humans reading the generated text and rating, editing or commenting on them (Reiter, 2010).

NLG can be categorized into two subfields: *content-to-text generation*, which generates text from non-linguistic input, and *text-to-text generation*, which generates texts from textual input (Barzilay, 2003). The latter is applied in automatic summarization, another subfield of NLP, and in question answering systems (Lapata, 2003). However, while summarization systems as a rule aim to make their output as close to similar texts written by humans as possible, not all of them use NLG components.

Automatic summarization systems generate summaries that can be classified as either *extracts*, which consist only of material copied from the input, or *abstracts*, which contain at least some material not present in the input. Extracts can be obtained using a shallow approach, which does not analyse sentences beyond the syntactic level. Instead, statistical methods can be used to determine the most important sentences in a text, and select those for a summary representing the full text (possibly syntactically edited). Abstracts require a deeper approach, which involves semantic information contained in the text and usually makes use of both NLU and NLG methods. Abstractive summarization can hence be seen as a form of text-to-text generation. Applying a deeper approach is difficult and computationally expensive, making the shallow extractive approach, which is more robust and less domain-specific, more popular (Mani, 2001). Only summarizations using a deeper approach require an NLG module to function, which means most of the systems currently in use do not. For a summary on state-of-the-art extractive and abstractive summarization systems, see Spärck Jones (2007).

Section 2.1 describes some of the major NLG applications that already exist. Section 2.2 describes the different tasks which an NLG system needs to perform to function appropriately, and the order of these tasks in the pipeline architecture, as decribed by Reiter and Dale (2000) and Reiter (2010). Section 2.3 goes into more detail on NLG systems using statistical methods specifically, since they will be used in this thesis.

2.1 NLG Applications

NLG has resulted in a large number of applications that generate text from abstract data. They are used either to present formal data to users in an accessible way, or to automate routine documentation in a certain domain, fully or partially. Automating such documentation is useful, since many people, such as doctors and programmers, spend large amounts of time on documentation of what they have done, despite that not being their main responsibility (Hüske-Kraus, 2003; Paris et al., 1995). NLG systems that aid with such tasks thus allow professionals to spend more time on more important business (Reiter & Dale, 2000; Reiter, 2010).

One main decision which has to be made when an NLG system is implemented is whether it will function as an independent text generator or as an authoring aid which offers suggestions. Though the former would be ideal, many of the tasks performed by an NLG tool are too important to assign to an automated system completely. An NLG tool that generates draft-like texts which are formed into a finalized text by an expert human author still allows work to be completed faster, while minimizing error. In some limited domains, like generating weather reports based on data, independent NLG is possible, but in others the resulting texts might not be of a consistently high enough quality to do without human intervention (Reiter & Dale, 2000).

A popular application of NLG is automatic generation of textual weather forecasts based on numerical weather data. Weather reports are relatively easy to generate because of the closed domain, the fixed structure of the texts, and the centrality of abstract numerical data. One of the earliest examples in weather report generation is FOG (Goldberg, Driedger, & Kittredge, 1994), which takes numerical weather forecasts of wind speeds, precipitation, and other meteorological phenomena over time, made by a supercomputer and annotated by a human forecaster, and generates a readable forecast for human use. An example of a textual weather forecast written using an NLG system, in this case one generated using FOG, is shown in the following example:

> The month was cooler and drier than average, with the average number of rain days. The total rain for the year so far is well below average. There was rain on every day for eight days from the 11th to the 18th.

FOG is used to generate marine forecasts, so the weather reports focus on wind and precipication, and do not include data on temperature. FOG is a multilingual system, and can be used to produce forecasts in both English and French. It does this by generating an abstract internal representation of the text, and mapping this to the output language automatically. The system has been in use commercially since 1993, and is one of the most often cited examples of successful NLG systems (Reiter & Dale, 2000).

Other well known NLG applications in the meteorological domain are:

MultiMeteo (Coch, 1998), another multilingual system which generated forecasts, and also provided the user with an interface to edit its output.

SumTime (Reiter, Sripada, Hunter, Yu, & Davy, 2005), which produced marine weather forecasts and is also used commercially.

RoadSafe (Turner, Somayajulu, Reiter, & Davy, 2008), which draws from large meteorological datasets and uses geographical information to aid road maintenance workers in determining which roads need attention at what time.

The SumTime system is especially notable, since users rated its resulting texts above reports written by humans for the same purpose. SumTime is a strictly rule-based system, and functions as a baseline for the performance of several statistical systems designed by (Belz, 2008) to produce similar results. The evaluation of statistical NLG systems as compared to rule-based ones will be discussed in more detail in section 2.3.

Another domain in which NLG has been applied is that of medicine. BT-45 (Portet et al., 2009; Gatt, Portet, et al., 2009) and its larger-scale successor BT-NURSE (Hunter et al., 2011) are systems which generate textual summaries of physiological data on infant patients in neonatal Intensive Care. Both systems produce summaries in English of clinical data on patients over a certain amount of time, to quickly inform doctors and nurses on the situation when time is an issue. This is an issue because there are large amounts of data on every individual patient, making it difficult for a medical expert to extract relevant facts. BT-45 uses techniques from intelligent signal processing to identify the most relevant information from 45 minutes of data on the patient's condition, and NLG to form said data into a coherent text, like the one cited in example 2.3. Similarly, BT-NURSE uses continuous physiological data gathered over time to automatically generate a textual summary of the data over a 12-hour nurse shift. BT-NURSE was evaluated by asking nurses to rate the output summaries for babies under their care, and it was rated to be helpful by a majority of the nurses, and understandable by 90% of them. These systems use and represent temporal data; the way they handle temporal information is discussed in detail in section 3.3.4.

Figure 2.3: *A sample* BT-45 *summary.*

> You saw the baby between 14:10 and 14:50. Heart Rate (HR) = 159. Core Temperature (T1) = 37.7. Peripheral Temperature (T2) = 34.3.
> Transcutaneous Oxygen (TcPO2) = 5.8. Transcutaneous CO2 (TcPCO2) = 8.5. Oxygen Saturation (SaO2) = 89.
> Over the next 30 minutes T1 gradually increased to 37.3.
> By 14:27 there had been 2 successive desaturations down to 56. As a result, Fraction of Inspired Oxygen (FIO2) was set to 45%. Over the next 20 minutes T2 decreased to 32.9. A heel prick was taken. Previously the spo2 sensor had been re-sited.
> At 14:31 FIO2 was lowered to 25%. Previously TcPO2 had decreased to 8.4. Over the next 20 minutes HR decreased to 153.
> By 14:40 there had been 2 successive desaturations down to 68. Previously FIO2 had been raised to 32%. TcPO2 decreased to 5.0. T2 had suddenly increased to 33.9. Previously the spo2 sensor had been re-sited. The temperature sensor was re-sited.

There are some other NLG systems in the medical domain beside BT-45 and BT-NURSE (see Hüske-Kraus (2003) for an overview), and a relatively large number of extractive summarization systems (Afantenos, Karkaletsis, & Stamatopoulos, 2005). The NARRATIVE ENGINE (Harris, 2008) (which despite what its name might suggest does not deal with sorting temporal information into a narrative) is a commercial NLG system which textually summarizes data acquired during an encounter between doctors and patients. An example of an older medical NLG system is TOPAZ (Kahn, Fagan, & Sheiner, 1991), which summarized blood cell counts and drug dosage data during intervals and at specific moments in time. NLG has also been applied in the medical domain to create systems that provide information to patients rather than medical staff, like PIGLIT (Binsted, Cawsey, & Jones, 1995), which generates personalized informational letters to cancer patients, and STOP (Reiter, Robertson, & Osman, 2003), a system which used individual data on smokers to write tailored letters to encourage them to stop smoking (although they were not more successful in causing people to quit than non-personalized letters).

NLG has been used to develop numerous other systems. Usually, a system using NLG methods serves to present information to a user in a straightforward manner, like most of the systems cited above. Because of the nature of the problems considered, a system usually functions within a given domain. Aside from the two examples described above, the ways in which NLG is currently applied include:

Summarization of other kinds of data, like stock reports (Kukich, 1983), biological data (Lester & Porter, 1997), turbine engineering data (Yu, Reiter, Hunter, & Mellish, 2007), and performance of subjects on languages tests (Williams & Reiter, 2008).

Assistance in writing documents, like multilingual software manuals (Paris et al., 1995), business reply letters (Coch, 1998) and clinical documents (Hüske-Kraus, 2003).

Inciting a user to do a certain thing, such as quitting smoking (Reiter et al., 2003) or buying a CD (Deemter & Odijk, 1997).

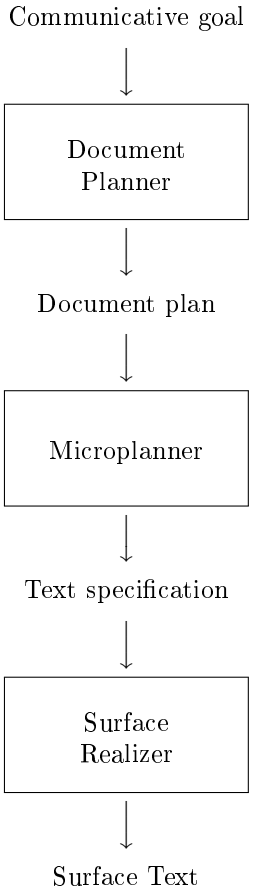
Assisting disabled users, for example by aiding deaf users in learning written English (McCoy, Pennington, & Suri, 1996) or verbally describing information in graphs to blind users (Ferres, Parush, Roberts, & Lindgaard, 2006).

Writing punning riddles (Binsted, Pail, & Ritchie, 1997; R. Manurung et al., 2008) and narrative prose (Callaway & Lester, 2002) to entertain and educate users.

2.2 Architecture of an NLG System

Engingeering a complex software system such as a full-fledged NLG system is simpler when the process is split up into separate tasks, especially if a team of people is involved in it. Reiter (1994) examines several previously developed NLG systems, and comes to the conclusion that all use similar methods, and

Figure 2.4: *The classic* NLG *system architecture proposed in Reiter (1994).*



correspond to a *pipeline architecture*. The pipeline architecture consists of different modules, each of which takes the output of a preceding module, along with additional information, as input. The output of one module is then used by the next, generally with minimal to no backtracking. This process continues until the final text is fully realized.

Reiter (1994) observe the systems they investigate to consist of three subtasks, with corresponding modules in the pipeline: *document planning*, which determines what information from a formal source needs to be included in a text and how to organize said information, *microplanning*, which is concerned with the linguistic expression of particular information, and *surface realization*, which takes the relevant information as determined by the other two modules, and generates an actual text as output. Each of these modules has their own subtasks, which are described in section 2.2.2. It is illustrated in figure 2.4.

Reiter and Dale's pipeline architecture is often seen as the 'standard' architecture of an NLG system, though there are alternatives. Mellish et al. (2000) found that, while most systems do use a pipeline architecture, there is a large variation in what modules are used in an architecture and how they interact.

They argue that the standard architecture is insufficient to represent all architectures found in different systems. The RAGS generation architecture (L. Cahill et al., 1999; Mellish et al., 2000) offers a formally defined framework for the interaction between different levels of an NLG system which allows accommodation of a wider range of architectures. This approach is more flexible in the types of systems it can account for, but uses a less rigid structure. Mellish et al. (2006) offer an extensive review of the RAGS architecture and its recent implementations.

There are alternative architectures which are not based on the pipeline model at all. Some NLG systems use a fully integrated approach, in which the task is presented as a set of constraints, with a realizer looking for a text which simultaneously satisfies all the constraints. This strategy has, for example, been applied in KAMP (Appelt, 1985), which uses a AI planner to generate referring expressions based on a number of communicative goals, and in H. Manurung, Ritchie, and Thompson (2000), who generate poetry based on a number of predefined poetic features.

For the sake of simplicity, I will outline the requisite tasks of an NLG system by the standard pipeline architectue of Reiter (1994), which remains a popular strategy. The following subsections deal with the input and output of an NLG system and with its subtasks. The final subsection of this chapter deals with the increasing use of statistical methods in NLG.

2.2.1 Input and Output

The first question to be posed when designing an NLG system is what the input and the output of the system should be. The targeted output is normally quite straightforward: a linear sequence of words and punctuation, that is, a text composed of natural language. As described by McDonald (1993), however, the input is more difficult to specify generally, as different systems are inconsistent in their input. Input can range from unanalysed numerical data, to structured objects within a formalism, to logical predicates. This phenomenon mirrors the process of NLU, where the input is clear, namely, text, but the output can vary greatly across applications. McDonald (1993) concludes that generation "may be considered to start at the first point where a speaker must appeal to her knowledge of language as she begins the process of carrying out some action through the use of language", but this is of more theoretical than practical value.

The standard pipeline architecture makes use of a more rigid definition of NLG input, which is described in Reiter and Dale (2000). They consider an NLG system as performing goal-driven communication, with every step an attempt to satisfy a *communicative goal*, which corresponds to the purpose of the text that is generated. NLG input can be represented as a four-tuple $\langle k, c, u, d \rangle$, with each element representing a separate source. An NLG system can then take input from each of these sources: k, the *knowledge source*', which is where the information is drawn from, c, the communicative goal itself, u, a *user model*, which characterizes the type of user for whom the text is generated and their preferences, and d, a *discourse history*, which keeps track of what has been said in the text so far.

The definition for NLG input proposed by Reiter and Dale (2000) is still quite general. The knowledge source in particular is only specified to be "typically encoded in one or more databases and knowledge bases", accounting for the

brunt of the variation between inputs. RAGS (L. Cahill et al., 1999) considers the form of information in some detail, but has little restrictions on how this information is represented in the input (Evans, Piwek, & Cahill, 2002). This has lead some researchers to argue for a stricter division between the control structure of a generator and the decisions made about the content it represents (Evans et al., 2002; Evans, Weir, Carroll, Paiva, & Belz, 2007). This would allow for more uniform generation components and hence easier comparison between different systems.

2.2.2 NLG Tasks

Reiter and Dale (1997) describes six tasks an NLG system is concerned with, and splits them into the three aforementioned modules. The tasks an NLG system has to perform are content determination, document structuring (or 'discourse planning), lexicalization, aggregation of messages, referring expression generation, and surface realization. Reiter and Dale (2000) divide surface realization up into two aspects: structure realization and linguistic realization. Reiter and Dale propose designating each of the tasks to one of the three modules from Reiter (1994), as summarized in table 2.1.

Table 2.1: *The modules of an* NLG *system and the associated tasks.*

MODULE	TASK
Document Planning	**Content Determination** *What messages should be communicated?* **Document Structuring** *In what order do the messages need to be presented?*
Microplanning	**Lexicalization** *What words and syntactic expressions should be used to express the messages?* **Aggregation of messages** *How should the messages be organized in sentences?* **Referring expression generation** *How to refer to entities?*
Surface Realization	**Structure Realization** *Organize the phrase specifications in sentences and paragraphs.* **Linguistic Realization** *Map the phrase information to words and syntactic constructs.*

The first module in the pipeline, document planning, is supposed to use the information source to generate a number of MESSAGES, or specific pieces of information, to be communicated. It takes the four-tuple $\langle k, c, u, d \rangle$ as its input. Its output is a *document plan*, a tree with messages as its terminal nodes, with the internal nodes specifying how the messages are grouped together. To achieve this output, Reiter and Dale (2000) propose this module is concerned with the task of *content determination*, deciding what messages should be included, and *document structuring*, building the tree that contains the messages.

In the case of text-to-text generation, it is likely that the sentences are already known from the input, and document structuring will result in an ordering

of sentences rather than a tree of messages. Microplanning and surface realization are not necessary in this case, but the system will need a *regeneration* module which performs surface repairs on the sentences, like adapting anaphora and introducing cohesion markers, to make them more fluent for the reader (Barzilay, Elhadad, & McKeown, 2002; Lapata, 2003).

The second module in the pipeline, microplanning, takes the document plan and uses linguistic knowledge to make decisions about sentence and phrasal structure, resulting in a *text specification*. The text specification is a refined version of the document plan, with *phrase specifications* (also known as *sentence plans*), which signify separate linguistic units like sentences, as its terminal nodes. Its internal nodes contain orthographic and typographic information with respect to the phrase specifications, like paragraph breaks. This module has to complete a number of tasks, namely *lexicalization*, deciding the words or phrases to be used to communicate the information, *aggregation*, deciding how the messages should be composed into sentences, and *referring expression generation*, determining how to denote entities (Reiter & Dale, 2000).

A phrase specification specifies a sentence, or possibly a different type of phrasal unit. The level of abstraction of this specification, however, can vary. A phrase specification can be expressed as an orthographic string, a fragment of canned text, and what Reiter and Dale (2000) refers to as *abstract syntactic structures*, and *lexicalized case frames*. The more abstract the phrase specification, the less overall work the microplanner has to do, but the more falls on the shoulders of the realizer (Reiter & Dale, 2000). This is touched upon in more detail in section 2.2.5.

The final module in the pipeline, surface realization, traverses the text specification tree, takes the phrase specifications at each terminal node, and converts them into units of text. There are two distinct aspects to this: *structure realization*, the process of mapping the data provided for a text into the structure of the target document given the mark-up language, and *linguistic realization*, mapping the phrase information into appropriate words and syntactic constructs given the target language. This stage of processing heavily depends on the level of abstraction of the phrase specifications, and is often carried out using an external system (Reiter & Dale, 2000).

There is some discrepancy over what NLG tasks belong in what module, especially concerning the microplanner. Reiter and Dale (1997) identify three tasks the microplanner is concerned with: lexicalization, aggregation, and referring expression generation, but L. Cahill et al. (1999) argue the tasks are not necessarily divided in this way. Tasks which Reiter and Dale (1997) identify to be part of the document planner are actually completed in the microplanner in some systems. For example, AlethGen (Coch, 1996) performs rhetorical structuring and sentence ordering in the microplanner. This variety is represented better in the RAGS architecture (Mellish et al., 2000). I will consider the tasks assuming the pipeline model, but this architecture is by no means an absolute.

2.2.3 Document Planning

The document planning module is responsible for the selection of information to be communicated, and how this information should be structured in the generated text. The result is a *document plan*, which is taken in by the other modules

to generate a complete text. The document planning module is concerned with two tasks: *content determination*, deciding what messages should be included, and *document structuring*, building the structure that contains the messages. The result is a document plan that can be passed on to the microplanner. The details of both tasks in document planning can be extremely different depending on the domain of the application and the type of texts which need to be generated (Reiter & Dale, 2000).

Because of the extreme diversity in types of information an NLG system has to represent and the large amount of possible discourse relations in a written text, both content determination and document structuring are still very specific to the domain of the application. This means that, to make an NLG system, it is necessary to implement methods for both content determination and document structuring that are specifically geared toward the type of output texts one expects to produce.

Content Determination

The task of content determination comes with a number of problems. Firstly, the data to be presented in the final document needs to be selected. This is important because in many NLG applications, the amount of available data is too large to include all of it in an easily understandable text. The more relevant parts need to be singled out, with what is relevant depending heavily on the goal of the application. Secondly, the selected data might have to be analysed and abstracted. This is necessary because the data a message represents might be too fine-grained to be presented in its entirety, or because the generated text might be bound by a certain word limit, or the interesting part of the data is a trend which exhibits only in an abstraction or generalization of the data (Reiter & Dale, 2000).

The FOG system (Goldberg et al., 1994) poses an example where abstraction is necessary. One forecast uses data gathered over 48 hours taking 76 different sample points into account. Considering variables for wind, temperature, visibility, weather, and freezing spray, the raw data may contain over 15,000 values. A forecast should communicate only the most notable trends in these data. Hence, it computes the changes and variation over time, reducing the information to a small number of significant events.

The type of data processing needed for content determination is not necessarily completely linguistic, as the FOG example shows. A system can use many different sorts of knowledge sources, which require different types of content determination. In fact, some researchers have argued that content selection should not be seen as part of NLG per se, but should instead be seen as an altogether different module, to make methodological comparisons easier (Evans et al., 2002, 2007).

Another factor which comes into play in content determination is user tailoring. Different NLG systems have different communicative goals and user models, and the resulting text should answer to these. FOG, for example, can generate marine forecasts, concerned mainly with wind speeds and directions, or public forecasts, focusing more on cloud cover and precipitation. Several NLG systems also produce different texts depending on the user. Dale et al. (1998) explore the possibilities of dynamic generation of descriptive texts about museum items, considering the wishes of the user. O'Donnell, Mellish, Oberlander, and Knott

(2001) generate hypertext taking into account the reading history of the user within the same context, dynamically adapting the user model. Bouttaz, Pignotti, Mellish, and Edwards (2011) generate textual descriptions of persons and projects in a virtual research environment based on the user's social context and policies created by the users.

Document Structuring

Irrespective of the content, a text needs to make sense to the user, and hence be coherent and easy to understand. The task of document structuring is concerned with structuring different messages with respect to each other in a specific format, like tree structures, graphs, or schemas. It takes into account not only on the messages' content, but also the relations they have to one another. There are various strategies to accomplish this, namely ones based on schemas, statistical methods, or heuristic methods.

The notion of *schemas* was introduced by McKeown (1985b). A schema is a structure consisting of a small number of messages, containing formal constraints on the types of information these messages and the relations that hold between them. The document plans is then constructed to answer to all constraints. Schemas are useful for texts with conventionalized patterns, where the order is relatively fixed. A schema can be designed by manually analysing an existing corpus of the type of text one is trying to generate, along with input from experts in the application's domain. FOG, for instance, uses a schema with constraints based on *data salience*, the relative significance of various types of weather information, for their marine forecasts. If there are more salient messages (such as warning status), they are placed earlier in the document plan than less salient ones, such as air temperature.

Rhetorical structure theory (Mann & Thompson, 1988) is used to define *rhetorical relations* between one message, called the *nucleus*, and another, called the *satellite*. The satellite can be an *elaboration*, an *exemplification*, a *contrast*, or a *narrative sequence* of the nucleus, among many other other options. This model can be used to perform document structuring based on schemas containing the rhetorical relations between messages (Reiter & Dale, 2000).

If a text requires a large number of messages with complex rhetorical relations, schemas quickly become insufficient. Data-driven statistical approaches are better suited to such tasks. Marcu (1997a) statistically derives weighted constraints between rhetorical relations and the ordering and adjacency of messages from a corpus of medical texts. He optimizes over these constraints, arguing that global coherence can be achieved if the local constraints are satisfied. Mellish, Knott, Oberlander, and O'Donnell (1998) propose using stochastic search methods as an alternative in the same task, using a genetic algorithm to select a locally optimal tree that is sufficiently coherent for people to understand. Statistical methods are useful in text-to-text generation, since rhetorical relations can be learned from a corpus. Lapata (2003) uses an unsupervised probabilistic model for document structuring to perform abstractive summarization.

Another approach to document structuring is through heuristic methods. In this case, the output is tailored to apply to the reader as well as possible. An example is SkillSum (Williams & Reiter, 2008), a rule-based system tailored toward users with low reading skills. An initial system was developed based on theoretical knowledge, which was then empirically improved through iterative

feedback of users on the results of prototype systems. Another, less user-specific approach to heuristic document structuring is taken in the text-to-text system described by Karamanis, Poesio, Mellish, and Oberlander (2004), who use the psycholinguistic theory of Centering to determine a sentence structure, rather than just optimizing over constraints.

2.2.4 Microplanning

The microplanning module is concerned with the linguistic expression of particular items of information. It deals with lexical and syntactic choice, referring expressions, and aggregation of messages. The microplanner makes decisions on how to express the messages it is fed from the document planner, and results in a text specification with phrase specifications at its nodes. Whereas the document planner relies on domain knowledge, and surface realization relies on linguistic knowledge, microplanning is concerned with the decisions which deal with the interactions between these different types of knowledge (Reiter & Dale, 2000).

Early NLG systems did not include a microplanning component, instead splitting up the generation process into two components, which, following Thompson (1977), were called the *strategic component*, which determines the content and strucutre, and a *tactical component*, which uses a grammar and dictionary to realize the text (McKeown, 1985a). During the 1990s, researchers came to the conclusion that a number of issues in NLG are neither primarily concerned with text and structure, nor with syntax and morphology (Reiter & Dale, 2000). The microplanner serves as a middle ground for tasks which belong in neither, so all the substantive decisions have been made when the text specification is passed to the surface realizer.

Lexicalization, aggregation and referring expression generation can be integrated in a microplanner in different ways. Wanner and Hovy (1996), for one, suggest a 'blackboard architecture', with no specific ordering of the three tasks. Their architecture places the document plan on a blackboard that is accessible by all three modules independently, allowing them to apply operations opportunistically in any order to transform the document plan into the text specification. This approach has some practical requirements, for instance that the three processes have a uniform input and output which they can all interact with, without unnecessarily constraining the decisions of different modules. Reiter and Dale (2000) propose an internal pipeline architecture for the microplanner instead, in which lexicalization, aggregation and referring expression generation are performed in that order. This results in a microplanner which is not as flexible, but easier to build.

Lexicalization

Lexicalization deals with the words and syntactic expressions which most aptly describe the information in the document plan. Every message in a document plan needs to be assigned both a syntactic structure and the correct words to communicate this information to the user. This may include making different decisions for different languages, as is the case with FOG, a multilingual system (Goldberg et al., 1994). There has been considerable debate about whether

the task of assigning words to messages belongs in the document planner, microplanner, or surface realizer (Reiter & Dale, 2000).

The simplest strategy to perform lexicalization is to apply a template to each message depending on its type. For example, Reiter and Dale (2000) describe a lexicalization function 'lexicalize' in their example system WEATHERREPORTER, which is ran with a message containing the type of rain, 'rainType'. The function, lexicalize(rainType) will simply return 'heavy' if the type of rain that the message describes classifies as heavy.

In many situations, there are several correct ways to lexicalize a message. In such cases, the lexicalization module has to make a choice regarding the best option. To lexicalize a message that describes a week-long rain spell in the WEATHERREPORTER system, for example, Reiter and Dale (2000) list the following options:

1. It rained for seven days from the 20th to the 26th.
2. It rained for seven days from the 20th.
3. It rained from the 20th to the 26th.
4. It rained on the 20th, 21st, 22nd, 23rd, 24th, 25th, and 26th.
5. It rained during Thanksgiving week.

These are only a few of the numerous ways this message can be expressed. The selection of the most appropriate available option is dependent on the conventions that hold in the application domain. This can be modelled either by rules based on examples and the opinions of experts, or by statistical modelling based on a corpus. Rule-based systems have used many different choice mechanisms to perform selection, including discrimination nets, decision trees, and systemic networks (Reiter & Dale, 2000). NITROGEN (Langkilde & Knight, 1998) is an example of a realization system which is primarily concerned with structure, but also statistically maps words onto lattices based on corpus data. It was later updated to include a forest ranking algorithm to perform the same task (Langkilde, 2000)

An alternative strategy, employed in Stede (1996) and Eldahad, McKeown, and Robin (1997) among others, is to divide information up in smaller chunks rather than fact-like messages, allowing a wider range of constraints on word selection. The input structure can then be repeatedly adapted to implement additional constraints. This allows a system to, for example, use a single word to express multiple predicates, and implement constraints on collocations. This can make communication more efficient: Eldahad et al. (1997) cite the example of reducing the sentence "The Jazz who extended their winning streak to 3 games, defeated the bulls" to "The Jazz defeated the Bulls for their third straight win". This is also useful in multilingual systems, since while the concepts included in a phrase might be the same, the target sentences are often not isomorphic in different languages (Reiter & Dale, 2000).

Aggregation

Simple lexicalized information based on the messages from the content determination module does not necessarily result in texts which are easy to read. The

simplest strategy to display messages would be to make each one an independent sentence. However, a complete text made up of such short, simple messages can be frustrating to read. If messages offer similar or related information, they can be construed into a single, more complex sentence, which offers the same information, but comes across as more natural.

Reiter and Dale (2000) describe a number of operations that can be performed in aggregation, which in their architecture is also concerned with the task of segmentation, specified by L. Cahill et al. (1999). These include adding in conjunctions or performing syntactic embedding when two messages are partially concerned with the same content, or have the same structure. Sentence aggregation is useful to make a text more concise, but makes texts syntactically more complex to comprehend. Therefore, its application has to be motivated by claims of professional writers, psychological research on text comprehension, or specific guidelines offered by style manuals. In general, only messages that are semantically related are easily understandable when they are aggregated (Reiter & Dale, 2000).

There are many statistical models to perform aggregation. Knight and Marcu (2000) use both bigrams and context-free probabilities in a system which selects between different sentence compressions. Daumé III, Knight, Langkilde-Geary, Marcu, and Yamada (2002) argue that a lexicalized model of syntax, commonly used in parsing, produces better results than using n-grams.

Referring Expression Generation

Whereas lexicalization provides linguistic expressions of concepts and relations, this module assigns expressions to entities. A referring expression, like a pronoun, proper name, or noun phrase, is used to identify a referent to the reader. There is a difference between preference in expression between an initial reference, which introduces an entity, and subsequent references, so a discourse model consisting of the previously mentioned entities needs to be maintained (Reiter & Dale, 2000). For subsequent references, especially pronouns, there are often multiple possible referents available. Referring expressions should always avoid ambiguity by being clear with respect to who or what is being mentioned, but on the other hand they also need to avoid redundancy, and make sure not to give explicit references more than the messages warrant (Dale & Reiter, 1995; Reiter & Dale, 2000). When there are multiple referents available, a referring expression must hence single out the appropriate option, while keeping the amount of information it provides minimal.

Referring expressions are used when referring to previously mentioned entities, but also when real-world entities are referenced, and an appropriate description is needed (if there are two chairs and one is red while the other is blue, for example). Dale and Reiter (1995) propose an algorithm based on the Gricean maxims for this problem, showing a preference for disambiguation based on certain qualities (most prominently colour). In interactive settings, priming toward a certain expression has been shown to effect change in preference (Gatt, Goudbeek, & Krahmer, 2011).

The discourse model attempts to model a representation of the entities a reader is currently considering, and which one a referring expression is most likely to refer to. A common observation is that a referring expression is less likely to resolve to an entity (less salient) if a long time has passed since it was

mentioned (Reiter & Dale, 2000). NLG systems have to take this into account when performing this task.

There have been rule-based and statistical approaches to performing referring expression generation. Reiter and Dale (1992) and Dale and Reiter (1995) describe an algorithm that iterates through a list of attributes, and adds attributes to an expression if they are needed to disambiguate the referring expression. Nenkova and McKeown (2003) use statistical methods to rewrite references in multi-document summaries. Recent work also saw an increase in context-sensitive referring expression generation, following Krahmer and Theune (2002) (Belz, Kow, Viethen, & Gatt, 2010).

The TUNA reference corpus (Gatt, Sluis, & Deemter, 2007) is used for statistical generation of referring expressions to images of both furniture and people in separate tasks. The corpus has been used for three generation challenges, with the task of generating referring expressions based on the properties of entities, resulting in new statistical methods for this purpose (Gatt, Belz, & Kow, 2009). One notable example is Di Fabbrizio, Stent, and Bangalore (2008), who take into account stylistic differences between speakers, and find that different feature sets are relevant for different speaker styles in realized referring expression generation. Hervás and Gervás (2009) use evolutionary algorithms to select features and realize referring expressions, although they find that case-based reasoning gives better results for realization.

2.2.5 Surface Realization

The surface realization module takes the information from the previous two modules, and uses it to generate a 'surface text' for the relevant information and domain, creating the final product of an NLG system. By the time a pipeline architecture has arrived to this point, the logical form of the text has been decided, but it has to be realized in a linguistic form. As described by Reiter and Dale (2000), there are two distinct aspects to this: 'structure realization' and 'linguistic realization'.

Structure Realization

Structure realization is the process of mapping the data provided for a text into the structure of the target document, organizing it in paragraphs and sections given the mark-up language. Linguistic realization maps the phrase information into appropriate words and syntactic constructs given the target language. Structure realization can be performed fairly easily because of the abundance of document preparation systems such as LaTeX, Microsoft Word, and internet browsers, which can be produced to present an NLG system to a user. Linguistic realization is hence more of an issue.

Linguistic Realization

The task of linguistic realization is partially dependent on the representation generated by the microplanner. Different NLG systems result in either more abstract text specifications, which give the document planner and microplanner more flexibility, but require more computation to be performed in surface realization, or more sophisticated text specifications, which require more work

to be done by the document planner and microplanner, in which case the output text determined more by the content than by standards. Different levels of abstraction require different levels of specification on the part of the realizer (Reiter & Dale, 2000).

Reiter and Dale (2000) describe a number of representations by which the surface realizer can generate the text. A *skeletal proposition* is an extremely abstract form, in which the logical form is comparable to formulas in predicate logic, and many tasks which are usually associated with the microplanner, such as lexicalization and referring expression generation, are performed by the surface realizer, based on additional external information. This can be convenient when the systems applies logical inference, and the text to be generated is just a single logical fact. A *meaning specification* is a less abstract specification, which supplies all the semantic information needed to form a coherent text to the surface realizer. A *lexicalized case frame* also maps the semantic content to base lexemes, but still requiring morphological processing. An *abstract syntactic structure* additionally specifies the basic grammatical properties of a sentence, which the realizer then applies syntactically by adding in the appropriate structure and function words. The least abstract messages in a logical form can be completely invariant, allowing the use of *canned text*, which requires no analysis in realization at all, though this usually still requires the realizer to apply orthography. *Orthographic strings*, are pieces of canned text which already contain fully orthographic information, and requires no interference at all.

In practice, many NLG systems require little in the way of surface realization, since much of text in many domains can be composed of canned text. There are several existing surface realizers available for the more complex realization tasks. While significant effort goes into supplying such systems with the appropriate format, building one's own grammar with a similar coverage is a massive amount of work, making the use of a pre-existent surface realizer usually a much more convenient choice (Reiter & Dale, 2000). Some of the realizers include KPML (Bateman, 1997), SURGE (Eldahad & Robin, 1996) and REALPRO (Lavoie & Rambow, 1997), which take into account different levels of abstraction and different grammatical theories.

More recent systems have seen a trend toward statistical realization, following NITROGEN (Langkilde & Knight, 1998), because of their wider coverage, which makes them less dependent on the domain. NITROGEN uses n-gram statistical information to localize syntactic constraints rather than apply syntactic rules. HALOGEN (Langkilde-Geary & Knight, 2002) is NITROGEN's successor, using a forest ranking algorithm to select the best realization. Another statistical realization system is OPENCCG (White, 2006), which is based on the Combinatory Categorial Grammar formalism (Steedman & Baldridge, 2011), a form of lexicalized grammar based on the syntactic type of expressions.

2.3 Statistical Methods in NLG

The standard pipeline model, as discussed in this chapter, is a widely used strategy in NLG. Traditional NLG systems are carefully handcrafted decision-making systems, with each decision made locally. This requires careful analysis of corpora and the assistance of experts in the field. Such methods have been and are still successfully applied, leading to advanced systems. BT-45 (Portet

et al., 2009), for example, uses pre-set rules in different modules to accomplish the task of generation. However, over the past few decades, methods in machine learning have opened up additional, altogether different options, which refrain from using handcrafted rule systems in favour of automated corpus analysis.

Statistical NLG can be performed in the context of the pipeline architecture, with each task being performed by an independent machine learning algorithm. For example, statistical NLG methods such as *collective content selection* (Barzilay & Lapata, 2005) and *entity grid ordering* (Barzilay & Lapata, 2008) have been developed to statistically apply content determination and document planning. SPaRKy (Walker, Stent, Mairesse, & Prasad, 2007) offers a trainable approach to lexicalization and aggregation. Bi-directional chart-based grammars, as first proposed by Kay (1996) can be used for the purposes of both parsing and generation. Such 'chart generation' has been used for statistical surface realization implementing various grammar models, like LFG (A. Cahill & Genabith, 2006) and HPSG (Carroll & Oepen, 2005; Nakanishi, Miyao, & Tsujii, 2005).

Other statistical approaches use corpora that contain direct mappings from input to output, resulting in fully integrated statistical NLG systems, which directly output the surface form of a text. Belz (2008) introduces pCRU, a statistical NLG system that generates weather reports in a form comparable to the output of the established SumTime system (Reiter et al., 2005). As opposed to the carefully constructed rules of the original SumTime generator, the pCRU generator in the same domain automatically converts the sentences in the corpus into derivation trees, which results in a set of chunks that can be used for generation. It then uses analysis of a corpus of weather reports and the data on which they are based to determine what phrasal chunks of texts occur in the context of what information. This results in sets of phrases to be used for things like wind directions, wind speeds, and time expressions, which along with information on what segment of a text is being generated, can be recombined into several alternative textual realizations of the data based on frequency counts of the different options in context. Multiple generation algorithms were used for this tasks (resulting in multiple sets of texts), allowing selection of the best results.

The approach taken by Belz (2008) offers a framework to compare statistical NLG systems with their handcrafted equivalent. In an evaluation of the pCRU system against the original SumTime system, the results of both systems were presented to and rated by experts. No significant difference in overall quality was perceived between the texts resulting from the handcrafted system and the best pCRU system. There was a preference for one system in their first evaluation, but a preference for the other by a similar margin in the second. Moreover, though statistical significance could not be shown, the combined results of comparisons to texts written by humans indicate that experts actually *preferred* the texts generated by the pCRU system over handwritten ones. This shows that, at least within the limited domain of weather reports, statistical NLG systems can perform just as well as or even better than handcrafted systems. Existing problems aside, these results provides an optimistic view of the future of statistical NLG systems.

A main disadvantage of handcrafted systems is that their development is a rigorous and time-consuming effort. The SumTime team estimates that approximately twelve months went into developing the SumTime system's mi-

croplanner and realizer, and an additional 24 on the surrounding activities like expert consultation. This poses a stark contrast to the pCRU system, which was built in less than one person-month (though SumTime has additional functionality; Belz estimates developing the pCRU system would take an additional two months) (Belz, 2008). All in all, this is an important advantage of statistical nlg: it allows systems to be developed faster (and hence cheaper) than the rule-based alternative. This applies to a system's initial development, but remains relevant throughout its run, since later adaptation and augmentation of rule-based systems also comes at a significant cost.

The question remains whether statistical generation is a viable option for more complicated texts. The weather reports generated by SumTime are very short and quite rigid in form, which generally means the decisions which have to be made to generate the text are relatively simple. Generating more extensive or irregular texts requires a more elaborate ruleset, which might be more of a problem with statistical generators, at least requiring larger corpora. On the other hand, many existing nlg systems are intended for very restricted domains. In this case, statistical nlg can be a useful option.

The dependence on corpora for statistical generation also comes with a potential advantage in terms of versatility. Rule-based systems often use algorithms specifically engineered to use the language associated with a particular domain. The fact that statistical systems can be retrained on different corpora makes already existing methods easily applicable in new contexts, which in turn makes a statistical system better equipped to deal with a wider variety of domains. This is illustrated by broad-range statistical realization systems such as HALogen (Steedman & Baldridge, 2011) and openCCG (White, 2006), which are largely domain-independent.

2.3.1 Using Corpora for Generation

Statistical nlg is heavily dependent on the existence of corpora for a given domain for decision making. Both abstract information and its linguistic representations must be available in large amounts for fully statistical nlg to be an option. This is not always the case, resulting in the potential disadvantage that much work will go into developing or adjusting a corpus for this purpose. It can also be an advantage, however, since it allows the use of resources even if they were not necessarily engineered for nlg problems. There are several strategies for acquiring corpus information.

The most straightforward option is creating a new annotated corpus specifically for generation. The tuna reference corpus (Gatt et al., 2007), used for statistical referring expression generation, was constructed through an online experiment, in which participants gave a description of a presented image. These descriptions were manually annotated to use them for corpus-based generation. Belz (2008) uses the SumTime-Meteo corpus (Sripada, Reiter, Hunter, & Yu, 2002), which was created for generation by the SumTime project from meteorological data and parallel sets of weather reports written by human forecasters. Since this type of corpus requires no annotated information, it is feasible to produce one for a single system, though still expensive.

Another possibility is bypassing the annotation of a corpus altogether, and applying factor analysis to a corpus to automatically infer what textual variations co-occur with what qualities of the expressed information. This is done

by Paiva and Evans (2005) for generation with certain stylistic features. They perform an experiment in which text is rated to have certain stylistic qualities, and generate a variation of a text in a certain 'stylistic dimension' through the fully integrated approach, with random variation in parameters, after which the resulting text can be evaluated with respect to the features of the target stylistic dimension. Generation is then repeated to learn the optimal parameters to achieve this dimension. Mairesse and Walker (2011) use a similar method to create PERSONAGE, a generator of which the parameters are trained on psychological findings on the linguistic qualities associated with certain styles. This results in texts of reliable quality which exhibit the qualities of a certain type of personality.

Lastly, data can be attained by reusing corpora which were developed for different ends than NLG. This is the strategy applied in NITROGEN (Langkilde & Knight, 1998), which uses annotated data from the Penn Treebank to perform statistical realization. This is the strategy applied in this thesis. The TIMEBANK corpus Day et al. (2003) (see chapter 4) was designed for statistical NLU applications, with the goal of deriving abstract temporal data from textual resources. To accomplish the current goal, I use the same information which is used for automatic annotation of texts, by analysing the abstract data and using it to predict the linguistic representation. Corpus-based NLG methods hence provide a new way of using existing resources for new applications.

This leaves open whether similarly annotated data is actually available from which to perform generation with a complete system. However, in the current case, this should not be a problem per se. Tools such as the TARSQI toolkit (Verhagen et al., 2005) are available with which texts can be automatically annotated in TIMEML (see section 3.3.3). This means that a tense and aspect generation module based on TIMEML will at the very least be usable in text-to-text generation, like automatic summarization and question-answering. Another possibility is automatic generation of narratives from temporal data through a visual interface, as done by (Elson, 2012).

2.4 The Issue of Time

The current chapter gives an overview of the general architecture of a NLG system. Although the topic of time is explored in greater detail in the following chapters, it is worth it to consider exactly what needs to be taken into account in the generation of a narrative. The decisions that have to be made to accurately communicate the time in a narrative are far more complex than just selection of tense and aspect. Time has an impact on all three of the modules, so I will briefly elaborate on at what points temporal information needs to be taken into consideration.

In document planning, time is an important factor in document structuring. Readers generally assume that events in a sentence are mentioned in chronological order (as discussed in the next chapter), but sometimes this is not the most natural expression for certain temporal information. Regardless, the order in which different information is presented in a text has a significant effect on the reader's understanding of what happened in what order, and document planning is therefore strongly affected by time. The order of the events in the narrative also has a strong effect on what tense, aspect, and temporal adverbials are most

appropriate. Time is usually not as important in content determination, though in systems like BT-45 and BT-NURSE, the frequency of certain events or the duration between them might be of influence with respect to whether they are notable at all.

The microplanning module is heavily affected by the time at which events took place. Aggregation is important for the same reasons as document structuring, since time might be expressed differently depending on the sentence structure of the document. The temporal expressions used in a sentence are not just dependent on the order of events mentioned in the overall document, but also on the events mentioned in previous sentences, and the existence of subordinate clauses. Lexicalization is also affected, since signal words and modal phrases are often necessary to clearly describe temporal relations. Time is also very important for referring expression generation, as many possible referents change over time. For example, the sentence 'the presidents of Zambia and Yugoslavia maintained a close friendship' contains a referring expression which can refer to different people depending on the time of the narrative, if to anyone at all. Temporal expressions generally have much in common with referring expressions, as discussed in section 3.3.2. The microplanning module ought to also decide on the most appropriate tense and aspect of verbs, given the temporal information and the order of events, and on temporal adverbs and adverbials.

After the microplanning module, all the substantial decisions of time need to have been made. Surface realization therefore ought to only be afflicted by time in a superficial way. Structure realization will realize the grammatical structures taking into account temporal adverbials and the modal verbs which were selected, using the grammatical rules for the language at hand. Similarly, linguistic realization will take into account tense and aspect information to inflect the parts of speech correctly.

For the current research, I focus on the task of tense and aspect generation for a hypothetical generator of narrative texts. I presume that all the tasks in document planning have already been completed, leaving a text specification with appropriate word ordering. Tense and aspect generation will be performed in the microplanner, which means that, depending on at what point in the pipeline the module is located, additional information from lexicalization, aggregation and referring expression is available. This hypothetical NLG system is described in more detail in section 5.1. The following chapter focuses on the nature of temporal information, and how it can be used in NLG tasks.

Chapter 3

Time and Narrative

3.1 Generating Narratives

Narratives make up the bulk of humanity's literatary history, and have been described as "the root metaphore for psychology" (Sarbin, 1986). Many of the texts which people read on a daily basis, such as newspaper articles, stories, and reports, are narratives, and yet generating a good narrative is a particularly difficult task in NLG. Labov (2011) defines narratives as "one way of recounting past events, in which the order of narrative clauses matches the order of events as they occurred", although this order can be interrupted by irrealis moods such as subjunctives and conditionals. By the same definition, a narrative always consists of at least two events. It usually begins with an *orientation*, which introduces and identifies the initial participants, time, and place at which the narrative took place, and often ends with a *coda*, a statement which returns the temporal setting to the present.

Narratives are a type of discourse distinct from other types, such as descriptions or arguments. Smith (2003) describes five modes of discourse: narrative, description, report, information, and argument. Narratives are distinguished by their temporally linear introduction of events and states, which contain temporal relations to one another. In reports, events and situations are related to the 'speech time' of the document, with time progressing forward and backward from that point. In descriptions, time is static, and progresses spatially as the text continues. Information and argument are atemporal, with information generally consisting of generic and generalizing sentences, and arguments of facts and propositions.

It is rarely the case, however, that narratives are fully sequenced: the chronological states and events in a narrative are frequently interspersed with descriptive and argumentative passages. The different modes make different types of contributions to a text, and different parts of a text usually have to be considered through various modes to make sense. However, a text normally has some predominant mode which represents the strategy used to understand the temporal relations within the text. The newspaper articles used in this research could be seen as narratives with descriptive passages, or as reports. The distinction is somewhat arbitrary when using statistical methods, since the temporal information which is present in the corpus texts is already annotated, and hence

independent of the mode in which they are considered. I will therefore consider all texts which describe two or more passed states or events as a narrative.

The most straightforward form of a narrative, however, remains one in which the events and states are related in chronological, linear order. In fact, temporal information can be derived purely from the ordering of events in a text: the sentence "Lars drank a carton of apple juice, put on his pants, and went to the store" does not mean the same as "Lars went to the store, put on his pants, and drank a carton of apple juice." This is due to *iconicity*, the notion that the order in which events are reported through language is the same as their chronological order, all other things being equal (Dowty, 1986). Narratives which adhere to this standard are called *iconic* narratives.

Iconicity provides a basic strategy to understanding the temporal relations within a narrative: it functions as a baseline by which to order the events in a text (Dowty, 1986). Any additional temporal information, such as tense, temporal adverbs, and other language cues, are evidence that adjustments need to be made to the baseline order. However, psycholinguistic research indicates that humans have a strong preference to see the described events as happening chronologically, and assume a linear event structure by default (Zwaan, Madden, & Stanfield, 2001). This basic approach to mapping events in time is known as the *iconicity assumption* (Dowty, 1986).

In practice, most narratives do not adhere strictly to iconicity, as evident in most newspaper articles and almost all of modern literature. There are many different ways to describe two events which have a temporal relation to one another, some iconic, and some not iconic. As an example, consider a narrative which deals with the events surrounding the speech Theodore Roosevelt held in Milwaukee in 1912. Roosevelt was shot in a failed assassination attempt, but decided to go on to hold the speech regardless. There are many ways to express this in English; a number of options is presented in the following example:

1. *"In 1912, Teddy Roosevelt was shot and held his campaign speech."*
2. *"In 1912, Teddy Roosevelt was shot, but he held his campaign speech afterward."*
3. *"Teddy Roosevelt was shot before holding his 1912 campaign speech."*
4. *"In 1912, Teddy Roosevelt held his campaign speech after being shot."*
5. *"In 1912, Teddy Roosevelt held his campaign speech, though he had been shot beforehand."*
6. *"In 1912, before Teddy Roosevelt held his campaign speech, he was shot."*
7. "In 1912, Teddy Roosevelt held his campaign speech, though he had been shot."

All of these sentences express the same narrative, and provide the same basic information, but the temporal specifications are different. Sentences 1 through 3 adhere to iconicity: they mention that Roosevelt got shot before mentioning his campaign speech. Sentences 4 through 7 do not: in each case, a temporal adverb or adverbial phrase, or a certain tense, is used to indicate an exception to iconicity. However, sentence 1, though iconic, does not communicate the narrative very well. This is because the situation is unusual enough that, despite

the sentence's iconicity, the knowledge that people usually do not hold speeches after getting shot could serve as semantic evidence that the events did not take place in the mentioned order, making sentence 2 a better alternative.

Otherwise, this example serves to illustrate there are many ways to accurately describe the sequence of events, and hence many correct decisions an NLG system could make to describe one sequence of events. Sentences 4 through 6, for example, each use a different aspect for 'being shot', partially dependent on whether the phrase is syntactically embedded or not. Sentence 2 uses the temporal adverb 'afterward' to holding the campaign speech, whereas sentence 5 achieves the same effect by attached the adverb 'beforehand' to the shooting. Sentence 7 does not use a temporal adverb at all, and instead brings across its message by shifting the perspective through the use of perfect tense. There are hence many ways to describe this simple narrative.

Despite the many available options, however, it is extremely easy to add misleading or confusing information. The relation between temporal cues and temporal relations is, after all, not at all specific. It is difficult to make uniform decisions with regards to temporal cues, because they often denote more than just when the event took place, and denote different things in different contexts. Consider, for example, the implications of tense in the sentences:

1. *"Mary runs one kilometer in five minutes."*
2. *"Mary ran one kilometer in five minutes."*
3. *"Mary is running one kilometer in five minutes."*
4. *"Mary was running one kilometer in five minutes."*

In terms of grammar, these sentences only differ in the tense and aspect of their main verb. Their meanings and the contexts in which they are the correct choice, however, are completely different. Sentence 1 implies Mary generally takes five minutes to run one kilometer, without denoting a specific event. Sentence 2 refers to the event of Mary running five kilometers some time in the past. Sentence 3 refers to Mary running a kilometer five minutes in the future. Sentence 4 could either refer to a point in the past during which Mary was running, or to the state of Mary's ability in the past (as in "Mary was running one kilometer in five minutes, then she broke her leg. Now that she's recovering, she's back to seven minutes"), but it could also refer to a point in the past which was five minutes *before* Mary ran one kilometer (as in "I was stuck in traffic. Mary was running one kilometer in five minutes. I needed to get her to the track field in time, so I ran a red light"). For an NLG system to select accurate tense and aspect in a narrative that describes one of these scenarios, it would need to have access to detailed information on the temporal relations and the sentential context.

Because of the ensuing complexity, NLG of narratives requires the use of temporal information at nearly every decision-making level. Document structuring is affected because of the iconicity assumption, with different orders possibly radically changing the meaning of a text. Lexicalization needs to be adjusted to allow for variation in tense and aspect, and temporal specifications adjust referring expression generation. This will be expanded upon in section 3.2.1.

The wide range of possible expressions for particular temporal relations and the wide range of meanings the use of a particular tense or aspect or a temporal

specification can have make the task of selecting temporal markers in an NLG system incredibly difficult. This makes narrative texts a specific challenge for NLG, as observed by Reiter, Gatt, Portet, and Meulen (2008) in the evaluation of BT-45: readers of the texts they generated still found that the human texts still had much better narrative structures than their generated counterparts.

The problem addressed in this thesis is a specific task in the microplanning module of generation of narrative texts: assigning tense and aspect to verbs based on known information about the temporal relations between events on a time line. This ties in with lexicalization, choosing the appropriate rendering of a lexeme, but it is conceivable that adding temporal information is an entirely separate task within microplanning which has received little attention in the literature so far. If lexicalization is restricted to the choice of verb, tense and aspect assignment can be performed as a separate task, based on the temporal properties of the event which is being expressed. A module performing this task might also benefit from taking information from the document structure, since, as example 3.1 illustrates, two events might be ordered differently, and have different tense and aspect depending on the ordering.

Where this thesis focuses on the role of verb tense and aspect in particular, this chapter is concerned with the various abstractions through which we can model time as it is used in language, and the computational applications that have been established using these abstractions. I will briefly describe the general mechanics of tense and aspect in English, and how Reichenbach (1947) linked their application to the description of time. I will outline the aspectual classes ascribed to verbs in context, complementary to their grammatical aspect, as has been described by Vendler (1957). Lastly, I will give an overview of time as it is handled in NLP, in particular event representation as time line data, as described by Allen (1983), and several applications dealing with time in NLU and NLG.

3.2 The Linguistics of Time

The ability to refer to things that are remote in time, described as *displacement*, is generally recognized to be one of the factors that distinguishes language from other types of communication (Hockett, 1960). Time is an intricate topic and there are various variables in language which affect temporal specifications. The non-verbal understanding of language across cultures amounts to a complete independent study in communication, namely that of chronemics (for an overview, see Bluedorn (2002)). For the current purpose, an understanding of temporal specifications in English event semantics, it is sufficient to say that time provides a framework to sequence events and specify their duration. A narrative text is a linguistic representation of a narrative, a sequence of events with specific durations. Verb tense, time adverbs, time adverbials and many pragmatic factors affect the placement of an event in a narrative, whereas verb aspect provides information about an event's duration, onset and completion status (Zwaan et al., 2001).

3.2.1 Tense and Aspect in English

A verb phrase is a direct description of an event in a narrative, and hence always takes place at some point in time. An event in a narrative relates to some specific points in time, and can also be specified to have a certain duration. Verb tense is used to describe the moment at which an event occurred relative to the time of speech. Together with temporal adverbs and adverbials and the iconicity assumption, this helps a speaker determine the time line behind a narrative, its chronology (Zwaan et al., 2001).

English has two absolute tenses: past and non-past (either present or future). A verb with absolute past tense refers to an event that took place before the moment of speech (like 'arrived' in "I *arrived* yesterday"), whereas non-past tense can refer to either an event that is ongoing at the moment of speech, or will take place in the future (like 'arrive' in "I *arrive* today" and "I *arrive* tomorrow"). To denote the future, the modal constructions 'will', 'shall' and 'be going to' are often used, but strictly this does not involve grammatical tense, as they can be used to denote an event in the present as well (like in "she *will* be at the store now", "I *shall* accept no disrespect now" and "I'm *going to* consider this answer correct").

Verb aspect is used to denote the duration of a verb, along with at what point it started and whether it is completed or not (Zwaan et al., 2001). The primary distinction in English aspect is that between progressive aspect and non-progressive aspect. A verb with progressive aspect denotes an ongoing process, marked by the presence of the auxiliary 'to be' before the present participle (like in "I *am* walking" and "I *was* walking"), as opposed to the non-progressive aspect which was used in all the examples in the previous paragraph. In the past tense, verbs can also be marked with the habitual aspect in two ways ('used to' and 'would' as used in "I *used* to live there" and "I *would* walk there everyday"). The perfect aspect, finally, is denoted by the auxiliary 'to have', and indicates that the event the verb refers to has already happened at a certain point in time, which is not necessarily the present. The perfect can hence be used in non-past tense, in past tense, or in a future construction (like in "I *have* left", "I *had* left", and "I will have left" respectively).

The perfect aspect can be combined with verbs inflected with absolute tense to form relative tense, relating not to the time of speech but to some other time (like "I will have left before you arrive" and "I had known her for years when she declared her love"). This can be used to denote a past perfect, a event which occurred before another event in the past, and a future perfect, an event which will have occurred before an event in the future. These complex relations are explained by Reichenbach's theory of tense (described in subsection 3.2.1).

Aside from tense and aspect, English verbs also have a *mood* and a *voice*, but these do not relate to time as strongly. A verb's mood can be either indicative, like all former examples, or imperative, which is used for commands and does not have tense or aspect. Aside from this, English uses the subjunctive moods of desire and counterfactuality, in cases where a verb expresses a situation which is not actually true (like subjunctive 'were' and counterfactual 'would ban' in 'if I *were* the president, I *would ban* grapes'). Of these, only the counterfactual subjunctive can express tense and aspect. A transitive verb can also express either active or passive voice, but this affects the syntactic structure of the sentence only. While all the examples given thus far were in the active voice,

the passive voice can use the same tenses and aspects in the same contexts, and exists independent of time.

Formal Theories of Tense and Aspect

Various formal systems have been developed to represent tense and aspect in English. An early and very influential model of tense is the one proposed by Reichenbach (1947). The existence of the past perfect motivated Reichenbach to explain temporal reference in terms of three parameters: the *speech time* (S), which occurs when the expression is uttered or written, the *event time* (E), denoting the time at which the event took place, and the *reference time* (R), an intermediary point with respect to which additional temporal relations may hold. The introduction of reference time is vital, because it allows temporal specification of an event as compared to a point in time which is not the time of speech.

These parameters have one of two types of relationship to one another: anteriority (marked by '<') or simultaneity (marked by ','). For example, $E, R < S$ denotes that the event time and the reference time occur simulaneously, and before the speech time, denoting simple past. The tense of a verb phrase corresponds to the relationship between R and S: Reichenbachian tense is 'present' in the case of R, S, 'past' in the case of $R < S$, and 'future' in the case of $S < R$. Relative tense corresponds to the relation between R and S: it is 'simple' in the case of R, S, 'anterior' in the case of $E < R$ and 'posterior' in the case of $R < E$. The possible Reichenbachian tense relations are shown in table 3.1, from Reichenbach (1947). Linking tense to temporal relations allows logical processing of time frames, and has found wide application in computation.

Table 3.1: *The Reichenbachian tense relations.*

Example	Temporal Relations	Tense Category	Traditional Tense
I see John.	S, R, E	simple present	simple present
I shall see John.	$S, R \quad E$	posterior present	simple future
I'll see John.	$S \quad R, E$	simple future	simple future
I saw John.	$R, E \quad S$	simple past	simple past
I have seen John.	$E \quad S, R$	anterior present	present perfect
I'll have seen John.	$S \quad E \quad R$	anterior future	future perfect
I had seen John.	$E \quad R \quad S$	anterior past	past perfect

Since Reichenbach (1947), numerous refinements of temporal logic have been developed. Prior (1967) includes modal operators to propositional calculus, which allows logical reasoning with temporal components. The event semantics of Davidson (1967) is based on those of Reichenbach, but treats events

as standalone entities linked to the representations of verb phrases. An indefinite number of things can then be asserted about the events and relations between them. Following Pnueli (1977), the modal version of temporal logic has found widespread use in computer science, particularly to keep track of ongoing programs working in parallel. Davidson's system, along with refinements by Parsons (1990), has gone on to be one of the main representations of time in semantics. Chapter 5 in Kamp and Reyle (1993) discusses the semantics of temporal reference in detail.

3.2.2 Aspectual Classes

It is notable that not all aspects are applicable to all verbs in English. It is possible to say "John is reading the book", but not to say "John is knowing the answer". This issue was addressed by Vendler (1957), who made the observation that there is a number of semantically different classes of verb, each of which is understood differently when it comes to time, and that each verb can be placed in one of these classes. This is known as a verb's *aspectual class*, *lexical aspect*, or, in the original German, *Aktionsart*. A verb phrase always has an aspectual class, though this is not always the aspectual class associated with the verb in isolation. Contextual factors such as arguments and adverbials can coerce a verb phrase into different classes (Bache, 1982).

Vendler (1957) originally sorted verbs in four classes: *processes* (originally *activities*), *culminated processes* (originally *accomplishments*, *culminations* (originally *achievements*), and *states*. A later paper by Comrie (1976) introduced a fifth class, the *point* (originally *semelfactive*). The different aspectual classes differ over three dimensions: *duration*, *telicity*, and *dynamicity* (Moens & Steedman, 1988)[1].

A durative verb denotes an event which took place over an extended amount of time, and can hence occur both in progressive and non-progressive aspect. Processes, culminated processes, and states are durative verbs. Processes and culminated processes differ in the dimension of telicity. Culminations and processes are telic, which means they have a natural 'end point', namely a change of state, which completes the event. Points and processes cannot be completed, and are hence atelic.

States differ from other durative verbs in the dimension of dynamicity. Dynamic verbs denote events, but states denote situations with duration that do not change naturally. All other types of verb are dynamic, and require energy to continue, but states do not change unless triggered by something conscious or external. Stative verbs are necessarily atelic, since they cannot have an end point of themselves.

A punctual verb, that is, one which is not durative, denotes an event that occurs instantly. Culminations and points are punctual, but differ in the domain of telicity: culminations trigger a change of state, but points do not. The change of state a culmination causes disallows the event from happening more than once, but points can be repeated indefinitely. For examples of all types of verbs, see table 3.2.

[1] I have chosen to use the terminology proposed by Moens and Steedman (1988), because it is more straightforward with respect to the relations between different types, which will be outlined in the next section, and because it makes the different aspectual classes easier to distinguish than the system Vendler (1957) proposes.

Table 3.2: *The five aspectual classes.*

	DURATIVE−	DURATIVE+	
TELIC+	**Culmination** (Achievement) 'reach', 'see', 'explode'	**Culminated process** (Accomplishment) 'eat a sandwich', 'read a book'	**State** 'know', 'love', 'contain'
TELIC−	**Point** (Semelfactive) 'spit', 'knock', 'hit'	**Process** (Activity) 'walk', 'dance', 'smile'	
	DYNAMIC+		DYNAMIC−

The aspectual class of a verb phrase affects what aspects it can have: dynamic durative verb phrases can occur in the progressive aspect, but culminations and states cannot (Vendler, 1957) (consider the culminated process "John was eating a sandwich" or the process "John was running", and the culmination "I was seeing John" or the state "I was knowing the answer"). Points can be expressed progressively, but denote an iteration if they do (consider "John was knocking") (Comrie, 1976). Aspectual class also affects whether certain temporal adverbs can be used. It is, for example, possible to use an atelic verb phrase to say "I danced for an hour", denoting a process, but it sounds odd to use a telic verb phrase in the same context, as in "John ate a sandwich for an hour", since this is a culminated process. Conversely, one can say "John ate the sandwich in five minutes", but not "I danced in five minutes" (Vendler, 1957; Comrie, 1976).

Moreover, the aspectual class has distinct semantic implications (Vendler, 1957; Comrie, 1976). For example, a telic verb phrase has a set beginning and end point, which the event must match for the verb phrase to accurately describe it. Hence, if John is eating a sandwich but stops before finishing it, he cannot be said to have eaten the sandwich. If I am dancing, on the other hand, I can stop at any point in time and will still have danced. The durativity of a verb phrase has a similar effect on the semantic connotations. Saying "it took John an hour to reach the town", a culmination, has different temporal implications than the same construction with a culminated process, like "it took John an hour to eat the sandwich": in the former, it is not implied that John was reaching the town at any moment in time before the end point, as opposed to the fact that John must have been eating the sandwich at any point during the hour.

It should be stressed that, while the verb phrases used as examples have general tendencies toward the aspectual classes they are listed with, this is by no means an exclusive relationship. The aspectual class of a verb phrase is something which depends on context, and is hence always a matter of interpretation, not something which is inherent in the verb (Bache, 1982). For example, if the verb 'walk' is given an object to form the phrase 'walk a mile', it becomes a culminated process rather than a process. Moreover, the aspectual classes can be considered abstractions of real-world events — it can be argued that there is no such thing as a punctual event, since all physical processes take time. Punctuality of a verb phrase can instead be taken to signify that it denotes events which are not interrupted by other events in the narrative, or that the narrative perspective does not dwell on its internal dynamics. Given this relativity,

it is more appropriate to judge sentences or the phrases within them by their aspectual class, than the verbs in isolation (Dowty, 1986).

Even if there is no exact relation between aspectual class and specific verbs, the 'preferred' class of a verb might still be useful to consider in generating the aspect of a verb in a text. Some verbs have strong tendencies toward one aspectual class, and this will affect how often they take certain aspects. 'Know', for example, a state, can be expected to occur in the progressive aspect very rarely, if at all. The verbs in the TIMEBANK corpus (Day et al., 2003) are also tagged with a 'class' (like 'occurrence' or 'perception, see chapter 4), which, though not specifically related to aspectual class, might indicate a tendency to one class or another. The connection between the verb and aspectual class can thus possibly be inferred by adding lexical knowledge and verb class to the decision algorithm.

3.3 Temporal Relations in Natural Language Processing

There is an enormous amount of texts that describe some sort of narrative, and hence incorporate temporal relations. This makes temporal relations a topic which has received consistent attention in NLP. Most of the existing systems dealing with time are NLU systems, but some NLG systems that incorporate temporal data have been developed recently. The following subsections describe Allen's interval algebra, a formal representation of temporal relations between events, and a selection of applications dealing with time in both NLU and NLP.

3.3.1 Time Lines and Temporal Relations

To study a narrative in a formal context, it is imperative to have a framework for the temporal relations between events. All narratives are made up of individual events with some sort of temporal relation to one another. Verbs describe such events, and there are many factors in a text that influence at what point in time an event is understood to have taken place. To establish a model of the relation between a verb's tense and aspect and the time of the event it denotes, it is necessary to be able to abstractly represent the time at which an event took place.

As it turns out, only few events in a narrative tend to be specified at a concrete reference point in time. For example, a newspaper article might specify the date of the main event it is concerned with, but this is still insufficient to map the event on a specific point in a time line: when another event has happened on the same day, but the exact hour of both is not specified, this leaves it ambiguous what temporal relation they have to one another. Moreover, a newspaper article will often forgo specifying the date and hour of surrounding circumstances, instead merely specifying their time as relative to other events: rather than the exact moment every event took place, the reader only knows which event came after which other event. This makes it difficult if not impossible to map the events described in a narrative to an absolute time line, where all events are marked with a specific timestamp. Such a model would not allow uncertainty of information, since to place an event on an absolute time line would require

assigning it a position with respect to all other events on the time line, rather than just the ones known.

Allen (1983) proposed a solution to this problem in the form of a time line which contains a list of events and absolute times in the form of intervals, and the temporal relation they have to one another. This is known as Allen's interval algebra. Allen's model was developed as an AI approach to processing temporal information in natural language, and has since become a widespread approach in NLP of time. It is central to the TIMEML markup language, on which the TIMEBANK corpus is based (Day et al., 2003). Some NLP applications which make use of Allen's interval algebra are described in sections 3.3.3 and 3.3.4.

Allen's interval algebra uses *temporal intervals* as its primitive, and *reference intervals* to denote the relations between different temporal intervals. A temporal interval t will always have a starting point $t-$, and an end point $t+$ which occurs after $t-$. Temporal relations between intervals can then be denoted by specifying the relation between the starting and end points of the two intervals.

Allen defines an event as a function from one situation to another. This poses a contrast with older systems, which tended to represent events as zero-width points on a time line. This produces some counterintuitive results when considered semantically. For example, in the case of a situation where a light is being switched on, there is a transition from a state where the light is off to one where the light is on. If the intervals correlating to these states do not contain the zero-width point, then at that point the light is neither on nor off. If they do contain the point, that means the light is both on and off (Allen, 1983). Taking temporal intervals rather than points as a primitive circumvents this problem.

The temporal structure of an interval algebra is hierarchical, which means that if an interval has a temporal relation to some other interval, it might be able to 'inherit' some temporal specifications from that interval. This use of time intervals allows the time of an event to be significantly vague, both with respect to absolute time and to other events. For example, if a sentence states "John burned his fingers while making pancakes this morning. He is moody.", 'this morning' refers to an interval in time, and so do 'burned his fingers' and 'making pancakes', but they operate at different levels in the hierarchy. 'This morning' refers to an interval which is bounded by specific points in time (for example, starting at sunrise and ending at noon at the day this was uttered), and 'making pancakes' starts and ends within that interval. Since 'burned his fingers' occurred during the event of 'making pancakes', its starting point must occur after, and its end point before, that of 'making pancakes'. It can be inferred from this that 'burned his fingers' must also have occurred in the interval 'this morning'. If 'is moody' is interpreted as not necessarily being a direct result of the events in the first sentence, it holds no relation to those, but since it is in present tense, it is known that its starting point occurs before the time of speech, and its ending point is undefined.

Allen's interval algebra allows the precise definition of any temporal relation which can occur in a narrative. The temporal relations are depicted in table 3.3 from Allen (1983). Six of the seven relations have an inverse, while the seventh is symmetrical, amounting to a total of thirteen types of temporal relations. For some purposes it might be beneficial to collapse some of these into more general categories. 'Starts' and 'finishes' and their inverses could, for example, be seen as a subset of 'during'. Allen's interval algebra allows parsing of a text where each interval and its relations can be added to a model. *Closure* can then be

applied on the model by adding in all temporal relations that can be logically inferred. Relations such as 'before' are transitive, so if event A is specified to occur before event B, which the model knows to occur before event C, closure would result in the additional relation that event A occurs before event C.

Table 3.3: *The thirteen possible temporal relations.*

Relation	Symbol	Symbol for Inverse	Pictoral Example
X before Y	<	>	XXX YYY
X equals Y	=	=	XXX YYY
X meets Y	m	mi	XXXYYY
X overlaps Y	o	oi	XXX YYY
X during Y	d	di	XXX YYYYY
X starts Y	s	si	XXX YYYYY
X finishes Y	f	fi	XXX YYYYY

Allen's model has thus far primarily been used for NLU of temporal relations. However, given a sizable corpus that is carefully annotated with temporal relations as defined in Allen's interval algebra, such as the TimeBank corpus (Day et al., 2003), it is entirely feasible to create an NLG system that generates text based on event data. To generate tense and aspect from time line data, it is important to consider how tense and aspect interact with time line information and temporal relations. This, among other things, is explored in the following subsection.

3.3.2 Automatic Inference of Temporal Relations

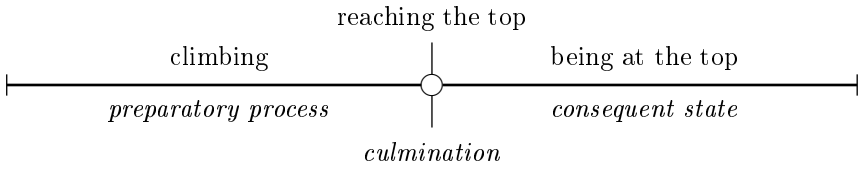
Time and Contingency

The relation between narratives and their formal time line representation is more complex than just mapping textual temporal specifications to time line events. Event descriptions which denote temporal relations tend to imply not just chronology, but also causal or consequential links. Some early work in NLP of temporal specifications includes the assertions put forward by Moens and Steedman (1988). They note that the sentence "At exactly five o'clock, Harry walked in, sat down, and took off his boots" is perfectly acceptable, while #"At exactly five o'clock, Harry took off his boots, sat down, and walked in", does not make sense. This indicates that the sentence's temporal content cannot be rep-

resented only by time line information. They propose an ontology to represent narratives with a focus on the role of culminations, establishing the surrounding events as either a preparatory process by which the culmination is realized, or a consequent state, which is a result of it. Certain linguistic constructions, such as 'when'-clauses, imply a *contingent* relation between different events, like a causal link or an enablement.

Moens and Steedman (1988) propose an algorithm with which the aspectual class of a verb phrase can be transformed into another aspectual class. For example, a point verb phrase may semantically not be considered progressive, but it can be coerced into an iteration, which can in turn be coerced into progressive, effectively changing the aspectual class. Their model is based on the notion of a *nucleus*, a structure consisting of a culmination, a preparatory process leading to that culmination, and a consequent state which is triggered by it. The nucleus is illustrated in figure 3.1.

Figure 3.1: *The three structures making up a nucleus.*



Culminated processes such as 'climbed the mountain' include all three of the components, and form a nucleus by themselves, as shown in figure 3.1. However, Moens and Steedman (1988) also argue that other aspectual classes, such as processes, cause readers to look for contingent relations in the vicinity. A reader assumes that a subsequent main clause contains the contingent event which fits in the nucleus. In the sentence "At exactly five o'clock, Harry walked in, sat down, and took off his boots", each main clause is a culmination. The subsequent state of each culmination is then automatically interpreted as the preparatory phase of the next culmination, adding contingent relations (of enablement) between the events. For the same reason, the second example, "At exactly five o'clock, Harry took off his boots, sat down, and walked in" does not make sense: there is a disagreement between the contingency implied by the temporal order and pragmatic knowledge. After 'Harry sat down', a reader looks to form a nucleus with the next clause, but 'walked in' cannot be a culmination of the state of sitting.

The temporal ontology of Moens and Steedman (1988) explains, among other things, changes in aspectual class depending on the situation. A complete semantic representation of a narrative is not only concerned with temporal information in the text, but strives to also draw contingent relations between states and events, combining them into nuclei. Taking this into account is necessary for a theoretical understanding of time line information. Moens and Steedman's model has lead to computational applications such as the one described in Marcu (1997b). Aspectual relations between different events are represented in TimeML by a specific link tag, which will be described in chapter 4.

Tense Resolution

From the tense of a phrase, it is possible to determine the relative location of the reference time with respect to the speech time, as described by Reichenbach (1947). The reference time is not specified by tense alone, so to be mapped to a specific point in time it needs to take information from context, either through temporal specification or from preceding sentences. Partee (1973) draws a parallel between tense resolution, where the reference time can refer to a point in time mentioned earlier in the text, and anaphor resolution, which refers to a previously mentioned entity. Building on this assumption, Webber (1988) proposes that the processing of a narrative makes use of a *discourse model*, which contains both information on the entities in the narrative and their properties and relations, and its events and situations along with their consequential and temporal relations.

Grosz and Sidner (1986) provide a model for anaphor resolution based on a *discourse focus*. The discourse focus marks the attention of a speaker when processing language, and shifts when it continues, favouring more salient entities. Webber (1988) draws on the parallel with anaphors to introduce the notion of *temporal focus*, which focus as a reference point for the mapping of tensed events of which the reference time is not specified. There are several possible temporal foci in a text at any point, but a temporal reference is only resolved to one of them.

The event structure proposed by Webber (1988) follows the tripartite nucleus structure of Moens and Steedman (1988). A newly mentioned event with a contingent relation to the temporal focus can either *maintain* that temporal focus, or provide a *local movement* of the temporal focus, either forward to the consequent state if the event culminates the previous one, or backward to the preparatory phase if it elaborates on the previous one. Which one of these is most appropriate depends on the speech time the tense indicates and on the context. An event which has no contingent relation to the temporal focus alters the discourse structure. Nakhimovsky (1988) outlines a similar theory which builds on the contingent relations of Moens and Steedman (1988) and the notion of temporal focus, describing the local shifts in focus as *micromoves* and shifts in focus as *macromoves*, respectively analogous to the local movements and changes in discourse structure in Webber (1988).

From the point of view of generation, this allows selection of tense based on knowledge of the speech time, the temporal focus, and the contingent relation a new verb phrase has with respect to the temporal focus. In example 3.2 from Webber (1988), an event E_a is described in sentence 1. The following event E_b, in sentence 2, occurred before E_a. If knowledge of contingency indicates that E_b is part of the preparatory phase of E_a, such as in sentence 2a, the temporal focus can be maintained if it is expressed. The only constraint on tense is then that the event time lies before the speech time, so both E,R–S (simple past) and E–R–S (past perfect) are options, as illustrated in sentence 2a. If, on the other hand, E_b simply precedes E_a without contingent relations, this indicates a change in discourse structure, and it must be expressed that the reference time lies before the event time, so the only option remains E–R–S, as illustrated in sentence 2b.

Example 3.2

1. *"John went to the hospital."*

2. a. i *"He took a taxi, because his car was in the shop."*
   ii *"He had taken a taxi, because his car was in the shop."*
   b. i #*"He broke his ankle, walking on a patch of ice."*
   ii *"He had broken his ankle, walking on a patch of ice."*

3.3.3 Natural Language Understanding Systems Involving Time

Several applications which make use of the large variety of strategies to deal with problems of tense and aspect have been developed within the field of NLU. The PUNDIT system (Passonneau, 1988) is an early NLU system which takes into account Allen's interval algebra and the aspectual operators described in Dowty (1979) to determine the temporal relations between events based on the tense and aspect of verbs in short reports of equipment failures on navy ships. Firstly, it establishes whether the clause is marked with a reference to an actual point in time, and whether that reference point is specific (referring to a particular event without the verb being modified by things like modals or frequency adverbials). It then classifies the described situation as states, processes or transitions, and feeds this into a time component along with its tense and information on whether the verb is perfective and progressive. Along with a record of the type of information verbs describe (concrete or less concrete), and Reichenbachian tense relations, this allows a deterministic method of defining temporal locations. PUNDIT does not take into account coercion as Moens and Steedman (1988) describe, but the framework theoretically allows for such distinctions.

Hwang and Schubert (1992) argue that Reichenbach's theory of tense is insufficient to model embedded clauses, because it necessarily defines the relationship of all three times with respect to each other. They propose the relationships in these cases to be hierarchical, and therefore better represented by a compositional account based on *tense trees*, which echo the hierarchy of temporal and modal operators in a sentence. A node in a tense tree can have up to three branches, for past, future, or perfect, which each add an additional reference point for events to be placed on the timeline. Moreover, the tree can contain horizontal *embedding links*, which link to the roots of trees which represent embedded verbs. This approach allows for a clear representation of complex temporal expressions, and is capable of representing the relations between different sentences altogether, with a tense tree for the whole text, though by itself it is not equipped to deal with non-iconic texts.

Hitzeman, Moens, and Grover (1995) also employ a hierarchical structure to model tense and aspect, but do this in the context of an HPSG unification framework. This hierarchy contains rhetorical and temporal relations, specified between two temporal expressions. It treats tense, aspect and temporal expressions as mutually constraining features. They use simple preference techniques such as semantic knowledge to determine to what other temporal expression a

relation refers to, and doesn't require full-fledged world knowledge to perform this task. They propose using an underspecified representation of the temporal structure in the case of ambiguity, so the appropriate references can be filled in when higher-level knowledge is available.

Corpus-based Applications

The development of TimeML (Pustejovsky, Castaño, et al., 2003) and the TimeBank corpus (Pustejovsky, Hanks, et al., 2003), described in detail in chapter 4, opened up many new possibilities for temporal NLU. TimeML is a markup language which marks linguistic items in the text as formal events, and describes the temporal relations that these events have to one another using Allen's interval algebra, as well as the aspectual relations in the tradition of Webber (1988) and subordination relations to account for the problems with embedding addressed by Hwang and Schubert (1992). The TimeBank corpus is the gold standard for TimeML-based applications.

The TimeBank corpus was designed to be able to statistically infer temporal relations in NLU, in particular for question-answering systems. As far as I know there is no corpus-based temporal question-answering system in operation as of yet. The work done so far has mostly been concerned with automatic annotation of other texts, by tagging the event and inferring the relations between them. This includes several modules, including event tagging, temporal tagging, and temporal ordering and anchoring of events. TimeBank functions as a gold standard for the systems developed for these modules, and they are generally evaluated by comparing their results to the TimeBank tags.

There are several event taggers available. EVITA (Saurí, Knippen, Verhagen, & Pustejovsky, 2005) is an event recognition tool which uses a basic strategy using linguistic knowledge, namely for parsing local context and extracting explicit morphological features, and statistical knowledge for word sense disambiguation. Llorens, Saquete, and Navarro-Colorado (2010) perform the same task, but show that results can be improved by adding information about semantic roles. An event recognition tool for Spanish, SIBILA (Wonsever, Rosé, Malcuori, Moncecchi, & Descoins, 2012), has also been developed, which functions on par with the results of EVITA, but to my knowledge all other languages lack applications for this purpose as of yet.

The temporal expression tagging task can be split up into two tasks: extraction, correctly identifying temporal expressions, and normalisation, assigning them a value in a standard format. There is quite a large number of applications available for this module; for an overview, see Strötgen and Gertz (2012). An early temporal tagger is TempEx (Mani & Wilson, 2000), a relatively simple rule-based system which uses TIMEX2 tags. It was developed into the widely used GUTime system[2], which is used in the TARSQI toolkit.

The TempEval-2 challenge (Verhagen, Saurí, Caselli, & Pustejovsky, 2010) was organized with a focus on temporal tagging, and resulted in five systems for English and three for Spanish, showing that normalisation is significantly more difficult than extraction, and also that the Spanish systems work better than the English ones. Current state-of-the-art systems include DANTE (Mazur & Dale, 2009), a system which tags in TIMEX2 and extracts using a JAPE (Java Annota-

[2] http://timeml.org/site/tarsqi/modules/gutime/

tion Pattern Engine) grammar for extraction and rule-based methods for normalization, and HeidelTime (Strötgen & Gertz, 2012), a domain-independent rule-based system which separates the source and resources like rules, and can therefore easily be adapted to different languages. Temporal expression tagging can be performed in different languages by automatic translation of the temporal expressions, and maintaining the same normalization rules (Saquete et al., 2006).

Automated temporal ordering is concerned with automatically inferring the temporal relations from a text annotated with events and times. An early application for this purpose was GutenLink[3] a rule-based temporal relation tagger[4], which took limited information about signals and subordination, and used a finite state transducer to determine the relation type (Verhagen et al., 2005). Mani, Wellner, Verhagen, Lee, and Pustejovsky (2006) use a Maximum Entropy classifier for learning, with the goal of inferring such temporal relations from plain texts. This results in prediction with an accuracy of 87.20%, which is significantly better than either a system based on hand-coded rules, or a system which used ME to learn from a baseline which is generated by hand-coded rules. To accomplish this, they merge the TimeBank corpus with another corpus, the Opinion Corpus, to increase its size. They also combat scarcity by using a temporal closure component named SputLink (Verhagen, 2004) which uses Allen's interval algebra to expand the time links specified in said corpora, explicitly specifying every inferrable temporal relation.

The first TempEval challenge was held in 2007 with the goal of automatically extracting temporal relations from texts which were annotated with event and time expressions (Verhagen et al., 2009). The process of reaching this goal was split up in three tasks, each of which could be performed and evaluated separately:

- **Task A: Assign temporal relations that hold between event and time expressions within the same sentence.**
- **Task B: Assign temporal relations that hold between the document creation time and event expressions.**
- **Task C: Assign temporal relations that hold between the main events of adjacent sentences.**

Six systems were built to complete these tasks, by different teams participating in the challenge. Evaluation was performed by comparison with the TimeBank corpus, on *strict* and *relaxed* grounds, with strict evaluation only counting exact matches and relaxed evaluation assigning some credit for disjunctions, like the system predicting `BEFORE-OR-OVERLAP` when the actual relation is `BEFORE`. The difference between the systems are not very large, with the best five F-measures of the systems ranging from 0.60 to 0.64 for task A, 0.74 to 0.81 for task B, and 0.57 to 0.64 for task C. The wvali team (Puçcaşu, 2007), which uses sentence-level syntactic trees and a bottom-up propagation of temporal relations followed by a statistical temporal reasoning system based on the words' tenses, performed the best in all three tasks.

[3] `http://www.timeml.org/site/tarsqi/modules/gutenlink/`

[4] Temporal relations are described using the `TLINK` tag in TimeML, which is described in detail in chapter 4

TempEval provides a framework for evaluating systems which infer temporal relations from TimeML-annotated texts. Bethard and Martin (2007) consider the tasks as pairwise-classification, and receive the second-highest accuracies on the tasks. Yoshikawa, Riedel, Asahara, and Matsumoto (2009) consider that the previous approaches mostly take the approach of pairwise comparison between events, neglecting logical constraints between temporal relations of different types. Instead, they use a Markov Logic model which simultaneously identifies the different relation types. This approach results in a score which is 2% higher for each of the tasks.

At the moment, the TARSQI toolkit[5] (Verhagen et al., 2005) is the only application which integrates the different components to perform fully automated temporal analysis of natural texts. It takes a modular approach, integrating GUTime, EVITA, GUTenLink, a subordination relation identifier called SLINKET, and SputLink, all in pipeline format, to identify and annotate events, times, and the temporal, aspectual, and subordination links between them. An updated version was released (Verhagen et al., 2009), which split the components into smaller tasks, and includes a visual component that combines the consistent temporal relations in a time line graph.

3.3.4 Temporal Relations in Natural Language Generation

Far less work relating to temporal relations has been performed in NLG than in NLU. To my knowledge, none of the existing work uses a machine learning approach to describe temporal relations, making it an interesting unexplored possibility. Narrative is a particularly difficult challenge in the area of NLG, and one of the major areas in which additional research is necessary (Reiter et al., 2008).

Oberlander and Lascarides (1992) evaluate a method of ordering event relations in generated text through defeasible reasoning. They see temporal structure as imposing constraints on discourse structure, since due to the iconicity assumption, the ordering of events in a text naturally implies a temporal relation between them. To address this issue, they propose a number of logical rules involving order in the text and temporal cue words, which provide a guideline as to when events can be ordered in certain ways. However, this strategy does not provide very natural results, and does not account for causal implicature as described by Webber (1988).

Dorr and Gaasterland (1995) actually describe an algorithm which generates tense, aspect, and temporal connecting words based on timestamped information. They use rules based on Allen's interval algebra, taking into account the time at which generation takes place and the time at which the events took place, to determine the tense in pairs of sentences. Vendler's aspectual classes are used to determine aspect. Both intervals and aspectual classes are considered to determine temporal expressions. This provides a basic strategy for tense and aspect generation.

A notable more recent example of an implemented NLG system involving tense and aspect is described by Elson (2012), who explores narrative discourse relations to develop a formal representation of story lines that can be used

[5] http://timeml.org/site/tarsqi/toolkit/

to automatically obtain structure and content information. Elson (2012) uses *Story Intention Graphs* (SIGs), which use formal relations to describe the intentional and temporal development in a story, to inferentially establish similarities and analogies between different stories and genres of stories. Elson (2012) introduces the SCHEHERAZADE system, which contains a generation component that assigns tense and aspect to verbs based on SIG data, as part of a feedback module to annotators.

SCHEHERAZADE's NLG module generates narratives through a rule-based system which considers the formal relations defined at each point, and then evaluates the time line from beginning to end, generating a sentence for each relevant time span. This results in a strictly linear, iconic narrative. The temporal relations between the formally defined events from the SIG are considered using Allen's interval algebra and Reichenbachian relations. The onset of the described event is used as an event time, and the story's perspective, set as a reference point, as its reference time. No lexical information was taken into account, though the system distinguishes between actions and statives. Following Dowty (1979), a lookup table was constructed to assign the temporal relation between events based on a reference interval. An alternate time line to refer to independent event sequences or counterfactuals was used to be able to model dialogue acts, as well as special modifications for subjunctives and conditionals. Based on the location of reference time and speech time, this allows the system to determine the appropriate tense and aspect to use in a given context.

Another example of representation of temporal data is provided by the BT-45 system and its successor BT-NURSE, described in section 2.1. These systems take abstract medical input over a certain time as its input data, and produces a summary of the patient's state. They involve statistically determined information based on reports written by humans and interviews with clinicians (Portet et al., 2009; Gatt, Portet, et al., 2009; Hunter et al., 2011). In BT-45, the statistical methods are applied to time in the *sequencing* mechanism used by the module for temporal abstraction. This mechanism uses a set of rules to determine whether any two events of a certain type answer to certain requirements, and occur within a certain time frame, in which case they can be merged into a single event denoting the entire sequence. For example, a set of upward spikes in a certain measurement that occur together within 10 minutes can be seen as a sequence, and treated as a single event (Portet et al., 2009).

Like in the algorithm of Dorr and Gaasterland (1995) and in SCHEHERAZADE, in the BT-45 system, tense is implemented based on the Reichenbachian model of time. However, it integrates the model of Webber (1988) and Portet et al. (2009), in which an event's reference time is equated to a previously established temporal focus. Additionally, the key events which started each paragraph were marked with explicit temporal references, and events which spanned long intervals of time also had temporal expressions denoting their duration. The results for BT-45 showed that automatically generated reports were as effective as visualizations of the data, but less efficient than similar reports written by humans.

One of the main findings of the experiments with BT-45 was that generating narratives involving temporal information is a particularly difficult task. Analysis of the generated texts as compared to the corpus texts showed that there was a large difference in the structures, and that this difference correlated with the differences in evaluation scores (Reiter et al., 2008). Problems occurred when

trends were evident on different time scales, since the system would focus on the events during a particular time in every paragraph, making it difficult to maintain an overview of the long-term progressions. Moreover, when temporal markers were inserted, it was often unclear to which event they referred, causing problems in temporal communication overall. Reiter et al. (2008) describe the problems with BT-45, and suggest a better temporal model, possibly in the form of a temporal ontology as per Moens and Steedman (1988) which functions separately from the domain ontology. Some of these issues were resolved in BT-NURSE, but generation of narrative texts remains a difficult problem in NLG.

The trends across existing systems indicate that a rule-based approach based on Reichenbachian tense is the most common way to generate tense and aspect, even if other modules in the system use machine learning methods to obtain the appropriate information for text generation. This thesis aims to apply a statistical approach to the generation of tense and aspect itself, in the context of a narrative.

Chapter 4

Materials

The experiment conducted in this thesis was performed by analysing feature data extracted from the TIMEBANK corpus, version 1.2. The TIMEBANK corpus contains much information which is theoretically usable from a generation perspective, like event class, lexical information, and temporal relations between events. The current chapter gives an outline of TIMEML, its uses, and its annotation scheme. In addition, it describes the TIMEBANK corpus and other corpora which have been annotated with TIMEML.

4.1 TimeML

Recent work in NLP has shown a growing interest in temporal information in documents, mainly for its use in many practical applications, such as question answering, machine translation, and discourse processing. Automatic inference of temporal expressions and event relations has hence become an active area of computational linguistics and semantics. TIMEML was developed to allow annotation of temporal information in narrative texts, for the purpose of automatically extracting information about narrative event structure. The existence of TIMEML-based corpora also allows statistical research in other fields of NLP, like generation, which is the approach taken in this thesis.

The TIMEML and TIMEBANK projects are a result of the TERQAS workshop and two subsequent AQUAINT workshops and projects, which had the aim of facilitating corpus-based research on the linguistics of time, particularly for question-answering systems (Pustejovsky, Castaño, et al., 2003). TIMEML is a robust, XML-based annotation scheme, which, unlike most prior attempts at event and temporal specification, represents the events and temporal expressions separately from the anchoring or ordering dependencies that may exist in a given text (Pustejovsky, Castaño, et al., 2003). TIMEML is mapped to the DARPA Agent Markup Language (DAML) Time Ontology, a collaborative and comprehensive effort toward standardisation of temporal relations which accounts for instants, intervals, durations, and other temporal concepts (Hobbs, 2002; Hobbs & Pustejovsky, 2003). It was built along the TIMEX2 guidelines for temporal expressions (Ferro, Mani, Sundheim, & Wilson, 2001; Ferro, Gerber, Mani, Sundheim, & Wilson, 2005), but adds several additional features, such as recognition of temporal prepositions and connectives, different classes

of event expressions, and dependencies between events and temporal expressions, or 'timexes' (Pustejovsky, Castaño, et al., 2003).

TimeML integrates the TIMEX2 annotation scheme with STAG, the temporal annotation language presented by Setzer (2001). This allows a wide range of event expressions, and both pairwise relations between events and relations between events and times to be marked up. The original TimeML, version 1.0, was designed to address four basic problems in event-temporal identification (Pustejovsky, Castaño, et al., 2003):

1. Time stamping of events (identifying an event and anchoring it in time);
2. Ordering events with respect to one another (lexical versus discourse properties of ordering);
3. Reasoning with contextually underspecified temporal expressions (temporal functions such as 'last week' and 'two weeks before');
4. Reasoning about the persistence of events (how long does an event or the outcome of an event last).

After considering the details of the representation granted by the initial system, several changes and extensions were added, the most significant of which was the logical separation of event descriptions and their temporal relations (Pustejovsky, Castaño, et al., 2003). The TimeBank corpus version 1.2, used in this thesis, was annotated using a more recent version of TimeML, namely version 1.2.1 (Saurí et al., 2006). This version has been adapted into an international standard for temporal annotation, ISO-TimeML, which allows a broader range of events to accommodate more languages (Pustejovsky, Lee, Bunt, & Romary, 2010; ISO 24617-1, 2012). ISO-TimeML has been used internationally to annotate most of the non-English corpora, although earlier corpora usually provide their own adaptation of TimeML. Initial descriptions of the TimeML specifications are given in Pustejovsky, Castaño, et al. (2003), and the specification for version 1.2.1 can be found online in Saurí et al. (2006).

4.1.1 Conceptual and Linguistic Basis

The temporal entities in TimeML are largely based on those proposed by Setzer (2001), namely *events*, *timexes*, and *relations*. Before giving an overview of the annotation scheme, it is useful to briefly address the nature of the different entities. Descriptions of the annotation and attributes of events, times, and relations in TimeML 1.2.1 are given in section 4.1.2.

Events

Events are tagged with the `EVENT` tag, and are taken to be *situations which happens or occur*. An event can either be punctual or durative. States or circumstances *in which something obtains or holds true* can also be classified as events, but only those which change in the time span of the text (Pustejovsky, Castaño, et al., 2003). For example, the phrase "New York is *on the east coast*" contains a state which holds true for a time span considerably longer than the span of a normal newspaper article, and hence is not considered an event, contrary to the state in "people *on board the air bus*" (Pustejovsky, Hanks, et al.,

2003). Events are generally expressed as tensed or untensed verbs, nominalizations, adjectives, predicative clauses, or prepositional phrases (Pustejovsky, Castaño, et al., 2003). Events are tagged with one of seven event classes. By TimeML version 1.2.1 (Saurí et al., 2006), an event can have multiple instantiations in a text, each of which is tagged with both tense and aspect.

Timexes

Times are tagged with the `TIMEX3` tag, and cover explicit temporal expressions. These are modelled on both the `TIMEX` tag from Setzer (2001) and the `TIMEX2` tag from Ferro et al. (2001). A timex can be a fully specified temporal expression, such as 'June 11, 1989', an underspecified expression, such as 'Monday', an intensionally specified expressions, such as 'last week', or a duration expression, like 'two years' (Pustejovsky, Hanks, et al., 2003).

Relations

The various relations that exist between the temporal elements of a document are tagged with `LINK` tags. They may exist both between events, and between events and times. TimeML recognises three different types of relations: `TLINK`, the temporal relations from Allen's interval algebra, `SLINK`, subordination relations which add information about whether a relation is subordinate to another, and `ALINK`, aspectual relations denoting contingent relations between an aspectual event and its argument (as in "John *started* to *run*"). Relations also include information on positive or negative polarity, and on recurring events (Pustejovsky, Castaño, et al., 2003).

4.1.2 Annotation

TimeML uses an XML-based annotation scheme, with the markup containing the temporal information. The initial annotation scheme is described in detail in Pustejovsky, Castaño, et al. (2003), and several versions have been published since. The most recent version was used to annotate version 1.2 of the TimeBank corpus. These changes can be found in the TimeML 1.2.1 specification[1]. The annotation guidelines for version 1.2.1 can be found in Saurí et al. (2006).

Annotation in TimeML is a time-consuming and expensive process. Some tools were developed to make the process easier for annotators. The 2003 TANGO workshops[2] provided the first graphical annotation tool for TimeML, by adding an extension, also called TANGO, to the CALLISTO annotation tool. This tool was used to produce the TimeBank 1.0, but came with some drawbacks: event, time and signal tags need to all be added through CALLISTO, before TANGO can be used to annotate the temporal, subordinate and aspectual relations between them. This results in a two-stage annotation process, which is time-consuming and can become confusing if a document contains large quantities of information. More recently, different annotation tools have been developed which tackle this problem, like BAT (the Brandeis Annotation Tool) (Verhagen, 2010) and CAT (the CELCT annotation tool) (Bartalesi Lenzi, Moretti, & Sprugnoli, 2012), both

[1] `http://timeml.org/site/publications/timeMLdocs/timeml_1.2.1.html`
[2] `http://www.timeml.org/site/tango/`

of which are intuitive and fast web-based annotation tools with many features, including TimeML annotation.

For the current purposes, it is sufficient to know with what tags the TimeML corpus is annotated, and where interesting information is located. I will therefore discuss the TimeML 1.2.1 tags, and represent some of the specifications for the tags in Backus-Naur Form (BNF) for clarification.

Event Annotation

TimeML 1.2.1 distinguishes between event *tokens* and event *instances*, which are marked with the `MAKEINSTANCE` tag. The reason for this is that in some sentences, like "John *taught* on Monday and Tuesday", one verb can represent two separate events. In this case, appropriate annotation results in two separate instances for 'taught'. Event tokens are marked within the text with the `<EVENT>` tag, as follows:

> All 75 passengers `<EVENT eid="1" class="OCCURRENCE">` died `</EVENT>`.

The BNF for this tag, from Saurí et al. (2006) is shown in figure 4.1.

Figure 4.1: *The* BNF *of an* ***EVENT*** *tag.*

```
attributes ::= eid class
eid ::= e<integer>
class ::= 'REPORTING' | 'PERCEPTION' | 'ASPECTUAL' | 'I_ACTION' |
'I_STATE' | 'STATE' | 'OCCURRENCE'
stem ::= CDATA
```

The `eid` attribute is an obligatory ID number by which the event can later be recognised. The event class is another obligatory attribute which descibes the type of the event. Verbs are often lexically ambiguous with regard to class, in which case the context is the deciding factor (Saurí et al., 2006). `stem` is an optional attribute which contains the stem of the tagged word. The properties of the seven classes are described in table 4.1, with explanations from Saurí et al. (2006) and examples from (Pustejovsky, Castaño, et al., 2003). Some of the event classes, like `STATE` are somewhat analogous to those proposed by Vendler (1957) and Dowty (1986), but the classification in TimeML is more focused on the event's role in the text than its temporal properties.

Tense, aspect, polarity and modality are represented by the `MAKEINSTANCE` tag. This is so that a sentence like "John taught today but he might not tomorrow", in which one event is instantiated with different attributes, can be represented. In most cases, events are only linked to one event instance (Saurí et al., 2006). A fully annotated `MAKEINSTANCE` tag looks as follows:

```
<MAKEINSTANCE eventID="e1" eiid="ei1" tense="past"
aspect="NONE" polarity="POS" pos="VERB"/>
```

The BNF for this tag, also from Saurí et al. (2006), is shown in figure 4.2.

All the attributes which come with an event instance are shown in this BNF. The `eiid` is an obligatory event instance number, distinguishing this instance from other instances (some of which might be of the same event). The `eventID` is also obligatory, and links the instance to the event it instantiates through its

Table 4.1: *The seven event classes in* TimeML.

EVENT CLASS	DESCRIPTION	EXAMPLES
REPORTING	Describes the action of a person or organisation declaring something, narrating, or informing about an event.	say, report, announce
PERCEPTION	Involves physical perception of another event.	see, hear, watch, feel
ASPECTUAL	Designates a complement and marks an argument with a temporal change.	begin, finish, stop, continue
I_ACTION	An intensional action; introduces an event argument from which we can infer something given its relation with the I-Action.	attempt, try, promise, offer
I_STATE	An intensional state; refers to an alternative or possible world.	believe, intend, want
STATE	A circumstance in which something obtains or holds true, which lacks dynamicity.	on board, kidnapped, love
OCCURRENCE	Other kinds of events which describe something that happens or occurs in the world. This class includes mainly culminations and culminated processes.	die, crash, build, merge, sell

Figure 4.2: *The* BNF *of a* ***MAKEINSTANCE*** *tag.*

```
attributes ::= eiid eventID tense aspect pos [polarity]
[modality] [signalID] [cardinality]
eiid ::= ID
{eiid ::= EventInstanceID
EventInstanceID ::= ei<integer>}
eventID ::= IDREF
{eventID ::= EventID}
tense ::= 'PAST' | 'PRESENT' | 'FUTURE' | 'NONE' | 'INFINITIVE' |
'PRESPART' | 'PASTPART'
aspect ::= 'PROGRESSIVE' | 'PERFECTIVE' |
'PERFECTIVE_PROGRESSIVE' | 'NONE'
pos ::= 'ADJECTIVE' | 'NOUN' | 'VERB' | 'PREPOSITION' | 'OTHER'
polarity ::= 'NEG' | 'POS' {default, if absent, is 'POS'}
modality ::= CDATA
signalID ::= IDREF
signalID ::= SignalID
cardinality ::= CDATA
```

eid. The pos tag indicates the part-of-speech of the event phrase, since not all event phrases contain a finite verb. The polarity attribute is an obligatory boolean which negates the event instance if set to NEG, in the case of sentences like "John did not *run* to work today". The modality attribute is only assigned if the instance is modified by a modal word such as 'would' or 'must', in which case it contains this word. The optional signalID attribute indicates a SIGNAL in the text which either motivates the existence of the MAKEINSTANCE, or indicates the value of its cardinality (in the case of sentences like "John read the book twice"), which is itself given as CDATA.

The tense and aspect tags contain the eponymous features of a verb phrase. If the pos indicates that the text is not a verb (gerunds are considered nouns), these will always be set to NONE. Finite verbs can have the PAST, PRESENT, or FUTURE tenses, combined with any of the four available aspects. An aspect set to NONE, in this case, refers to a simple non-progressive aspect. These tags can be applied regardless of the mood of the verb.

Some subordinate clause types in English, namely the infinitive, present participle, and past participle, contain non-finite verbs. Though such verbs lack a tense, the tense tag is in this case set to the clause type. If the verb is an infinitive, the aspect attribute can have any of its four values. If the verb is one of the participles, however, the aspect will always be set to NONE, even if a past participle is a part of a passive or perfective construction. The tense is set to PRESPART in the case of a present participle, and PASTPART in the case of a past participle. TimeML employs specific guide lines for annotators with respect to when a past participle should be seen as a verb, and when it should be seen as an adjective (Saurí et al., 2006).

Time and Signal Annotation

Linguistic expressions of time, as well as the speech time within the document, is tagged with the tag TIMEX3, to distinguish it from the TIMEX tags in Setzer (2001) and TIMEX2 tags in Ferro et al. (2001). This denotation is designed to be as compatible with the TIMEX2 annotation guidelines as possible (Pustejovsky, Castaño, et al., 2003). Unlike event tags, timexes are not instantiated separately; all the attributes are given within the tag in the text. A fully annotated TIMEX3 tag is shown here:

```
   <TIMEX3 tid="t35" type="TIME" value="1998-02-11T22:00"
temporalFunction="true" functionInDocument="NONE"
anchorTimeID="t30">10 p.m.  Wednesday</TIMEX3>
```

The BNF for this tag, from Saurí et al. (2006), is shown in figure 4.3.

The current experiment uses little information from the TIMEX3 tag. From a generation perspective, temporal specification is a task which is concerned with assigning tense and aspect as well as temporal expressions. The attributes given here therefore do not represent available information in most cases. However, it is at least possible that the aggregation module has already specified the presence of a temporal expression within a sentence, and the type attribute contains such elementary information that this should also be available. Hence, if an event has a relation to a timex within the same sentence, it is reasonable to assume that the type of this relation and of that time are available as information.

Figure 4.3: *The* BNF *of a* ***TIMEX3*** *tag.*

```
attributes ::= tid type [functionInDocument][beginPoint]
[endPoint][quant][freq]
[temporalFunction] (value | valueFromFunction)
[mod][anchorTimeID]
tid ::= ID
tid ::= TimeID
TimeID ::= t<integer>
type ::= 'DATE' | 'TIME' | 'DURATION' | 'SET'
beginPoint ::= IDREF
beginPoint ::= TimeID
endPoint ::= IDREF
endPoint ::= TimeID
quant ::= CDATA
freq ::= CDATA
functionInDocument ::= 'CREATION_TIME' | 'EXPIRATION_TIME' |
'MODIFICATION_TIME' | 'PUBLICATION_TIME' |
'RELEASE_TIME'| 'RECEPTION_TIME' | 'NONE'
default, if absent, is 'NONE'
temporalFunction ::= 'true' | 'false' default, if absent, is
'false'
temporalFunction ::= boolean
value ::= CDATA
value ::= duration | dateTime | time | date | gYearMonth |
gYear | gMonthDay | gDay | gMonth
valueFromFunction ::= IDREF
valueFromFunction ::= TemporalFunctionID
TemporalFunctionID ::= tf<integer>
mod ::= 'BEFORE' | 'AFTER' | 'ON_OR_BEFORE' | 'ON_OR_AFTER' |
'LESS_THAN' | 'MORE_THAN' | 'EQUAL_OR_LESS' |
'EQUAL_OR_MORE' | 'START' | 'MID' | 'END' | 'APPROX'
anchorTimeID ::= IDREF
anchorTimeID ::= TimeID
```

Another instance in which timex information is available, is if it denotes the time at which the document is created. Each document annotated in TimeML has a `TIMEX3` tag which denotes the creation time, marked by the value `CREATION_TIME` in the `functionInDocument` attribute. Timexes contain a unique ID number like event tags, the `tid`, which are included in the relation tags. By these means the relation with respect to the creation time and temporal expressions within the sentence can be obtained. The other attributes of the `TIMEX3` tag in TimeML 1.2.1 are elaborated upon in Saurí et al. (2006).

As regards signal tags, it is not inconceivable that a lexical decision has already been made to describe the relation between events. It is worthwhile to consider this information, even if it is just to check whether pre-existing signal information is valuable for generating. The `SIGNAL` tag itself contains just one attribute, which is a unique ID number like the `eid` and `tid`. It looks like this:

```
<SIGNAL sid="s3">after</SIGNAL>
```

The signal phrase related to a relation can be identified by identifying the text marked with the `signalID` attribute in the relevant `LINK` tag.

Annotation of Temporal Relations

Temporal relations are denoted through a `LINK` in TimeML. Three different types of relations are tagged: temporal relations (`TLINK`), subordination relations (`SLINK`), and aspectual relations (`ALINK`). The annotations of the different relations are quite similar, but they are semantically distinct.

A `TLINK` denotes information about temporal relationships between events, and between events and times. The relationship can be of a number of types largely based on Allen's interval algebra. A `TLINK` looks like this:

```
<TLINK lid="l2" relType="IDENTITY" eventInstanceID="ei338"
relatedToEventInstance="ei339"/>

<TLINK lid="l10" relType="BEFORE" eventInstanceID="ei305"
relatedToTime="t30"/>
```

The BNF for the `TLINK` tag is shown in figure 4.4.

The `lid` is the unique ID number for a link. A `TLINK` is established between two event instances or between an event instance and a time. The `eventInstanceID` contains the ID of the event instance which holds the temporal link. This event instance is being related to either the event instance of which the ID is contained by the attribute `relatedToEventInstance`, or the time of which the ID is contained by the `relatedToTime` attribute (though never both). A `TLINK` can also be held by a time instead of an event, in which case the ID is contained by the attribute `TimeID`. It can then only relate to an event through `relatedToEventInstance`. In the TimeBank corpus, the relation most often extends from an event, not a time.

The obligatory `relType` attribute contains the type of relation which the first entity has to the second. As such, if `eventInstanceID` is `ei1`, `relatedToEventInstance` is `ei2`, and `relType` is set to `BEFORE`, `e1` occurred before `e2`. `relType` can be one of fourteen different relations, shown in the BNF. These relation types are analogous to those of Allen (1983), and inherit both their transitivity and reversibility. Examples of what type of sentences contain

Figure 4.4: *The* BNF *of a* ***TLINK*** *tag.*

```
attributes ::= [lid] [origin] (eventInstanceID | timeID)
[signalID] [syntax]
(relatedToEventInstance | relatedToTime) relType
lid ::= ID
lid ::= LinkID
LinkID ::= l<integer>
origin ::= CDATA
eventInstanceID ::= IDREF
eventInstanceID ::= EventInstanceID
timeID ::= IDREF
timeID ::= TimeID
signalID ::= IDREF
signalID ::= SignalID
relatedToEventInstance ::= IDREF
relatedToEventInstance ::= EventInstanceID
relatedToTime ::= IDREF
relatedToTime ::= TimeID
relType ::= 'BEFORE' | 'AFTER' | 'INCLUDES' | 'IS_INCLUDED' |
'DURING' | 'DURING_INV' | 'SIMULTANEOUS' | 'IAFTER' | 'IBEFORE' |
'IDENTITY' | 'BEGINS' | 'ENDS' | 'BEGUN_BY' | 'ENDED_BY'
syntax ::= CDATA
```

these relations, from Saurí et al. (2006), are shown in table 4.2. The table shows the asymmetrical relations in pairs, to indicate how a relation can be inversed.

The `origin` attribute is optional, and shows whether the `TLINK` was generated manually or by closure. The `signalID` attribute is also optional, and if present provides the signal ID which marks this temporal relation if it is explicitly signaled in the text. The `syntax` attribute optionally holds syntactic information that was used to generate the link, if an automatic annotator was used.

Annotation of Subordination Relations

The `SLINK` tag contains a relation between a subordinating event and a subordinate event, to denote subordinate clauses. Thus, in the sentence "John *said* that he *taught* on Monday", the event instance linked to 'taught' will be a subordinate (of type `EVIDENTIAL`) of the event instance linked to 'said'. An SLINK looks like this:

```
<SLINK lid="l28" relType="MODAL" eventInstanceID="ei1991"
subordinatedEventInstance="ei1992"/>
```

The BNF for a `SLINK`, from Saurí et al. (2006), is given in figure 4.5.

The attributes are mostly the same as those in the `TLINK` tags explained before, but `relType` contains subordination relations rather than temporal relations in this tag. Moreover, subordination relations only occur between events. Unlike the `TLINK`, the relationships denoted here are neither transitive nor reversible. Some examples of these relations, from Saurí et al. (2006), are shown in table 4.3.

Table 4.2: *Temporal relations between entities in* TimeML.

RELATION TYPE	RELATIONS	CONTEXT
BEFORE AFTER	ei1 BEFORE ei2 ei2 AFTER ei1	*"The police looked into the* slayings (ei1) *of 14 women. In six of the cases suspects have already been* arrested (ei2)*."*
INCLUDES IS_INCLUDED	t1 INCLUDES ei3 ei3 IS_INCLUDED t1	*"John* **arrived** (ei3) *in Boston last* **Thursday** (t1)*."*
DURING DURING_INV	ei4 DURING t2 t2 DURING_INV ei4	*"John* taught (ei4) *for* twenty minutes (t2) *on Monday."*
SIMULTANEOUS	ei5 SIMULTANEOUS ei6 ei5 SIMULTANEOUS ei6 ei7 SIMULTANEOUS ei8 ei8 SIMULTANEOUS ei7	*"The unit said it can* **provide** (ei5) *no* assurance (ei6) *a* transaction (ei7) *will* occur (ei8)*."*
IBEFORE IAFTER	ei10 IBEFORE ei11 ei11 IAFTER ei10	*"All passengers* died (ei10) *when the plane* crashed (t11) *into the mountain."*
IDENTITY	ei12 IDENTITY ei13 ei13 IDENTITY ei12	*"John* drove (ei12) *to Boston. During his* drive (ei13) *he ate a donut."*
BEGINS BEGUN_BY	t3 BEGINS ei14 ei14 BEGUN_BY t3	*"John was* in the gym (ei14) *between* 6:00 PM (t3) *and* 7:00 PM (t4)*."*
ENDS ENDED_BY	t4 ENDS ei14 ei14 ENDED_BY t4	

Figure 4.5: *The* BNF *of an* ALINK *tag.*

```
attributes ::= [lid] eventInstanceID [signalID]
subordinatedEventInstance relType [syntax]
lid ::= ID
lid ::= LinkID
LinkID ::= l<integer>
eventInstanceID ::= IDREF
eventInstanceID ::= EventInstanceID
subordinatedEventInstance ::= IDREF
subordinatedEventInstance ::= EventInstanceID
signalID ::= IDREF
signalID ::= SignalID
relType ::= 'MODAL' | 'EVIDENTIAL' | 'NEG_EVIDENTIAL' |
'FACTIVE' | 'COUNTER_FACTIVE' | 'CONDITIONAL'
syntax ::= CDATA
```

Table 4.3: *Subordinate relations between entities in* TimeML.

RELATION	CONTEXT
ei1 MODAL ei2	*"John* promised (ei1) *Mary to* **buy** (ei2)) *some beer."*
ei3 EVIDENTIAL ei4	*"John* said (ei3) *he* **bought** (ei2)) *some wine."*
ei5 NEG_EVIDENTIAL ei6	*"John* denied (ei5) *he* **bought** (ei6)) *only beer."*
ei7 FACTIVE ei8	*"John* forgot (ei7) *that he was* **in Boston** (ei8)) *this year."*
ei9 COUNTER_FACTIVE ei10	*"John* forgot (ei9) *to* **buy** (ei10)) *some wine."*
ei11 CONDITIONAL ei12	*"If John* brings (ei11) *the beer, Mary will* **bring** (ei12)) *the chips."*

Modal relations contain references to possible worlds, mainly represented by either an `I_ACTION` or `I_STATE`. Evidential relations provide evidence for their subordinate event, and are usually either `REPORTING` or `PERCEPTION` events. Similarly, factives contain a presupposition of the veracity of their subordinate event. Negative-evidentials and counter-factives do the same as evidentials and factives, but introduce a presupposition of the non-veracity of their subordinate event instead. Conditionals denote straightforward conditional relations. These roles can be syntactically determined and have semantic implications, which means they could be determined in the document planner and also might have implications on tense and aspect.

Annotation of Aspectual Relations

Aspectual relations, marked by the `ALINK` tag, denote relations between an aspectual event and its argument event. Events can thus be placed in the context of the nucleus of WEBBER1988, possibly allowing an analogy with the temporal focus of Moens and Steedman (1988). Like the subordination relations, these are neither transitive nor reversible. In the TIMEBANK corpus, aspectual relations are relatively rare compared to subordination and temporal relations. A sample `ALINK` is presented here:

> *"John* started (`ei1`) *to* **read** (`ei2`)**)**.*"*
> `<ALINK lid="l54" relType="INITIATES" eventInstanceID="ei1"`
> `relatedToEventInstance="ei2"/>`

The attributes of an `ALINK` are exactly the same as those of the `SLINK`, so it is not necessary to show the BNF here. The only difference is that in these links, the `relType` contains an aspectual relation. The six options for the `ALINK relType` are shown in table 4.4, with examples from Saurí et al. (2006).

Table 4.4: *Aspectual relations between entities in* TIMEML.

RELATION	CONTEXT
`ei1 INITIATES ei2`	*"John* started (`ei1`) *to* **read** (`ei2`)**)**.*"*
`ei3 CULMINATES ei4`	*"John* finished (`ei3`) **assembling** (`ei4`)**)** *the table."*
`ei5 TERMINATES ei6`	*"John* stopped (`ei5`) **talking** (`ei6`)**)**".
`ei7 CONTINUES ei8`	*"John* kept (`ei7`) **talking** (`ei8`)**)**".
`ei9 REINITIATES ei10`	*"John* resumed (`ei9`) **reading** (`ei10`)**)**".

Additional Tags

The introduced tags make up the basis of TIMEML. The TIMEBANK corpus contains the documents annotated in normal TIMEML, but also contains an annotation with some extra tags, like sentence markers, document-level tags, and entity tags. I will use this annotation, to extract sentence tags in order to

compare the sentential positions of events in a text. Sentence tags are put at the beginning and end of every sentence in the form of <S> and </S>. A fully annotated sentence from the TimeBank 1.2 corpus with sentence tags is shown in example 4.6. It is possible to add additional tags using different markup languages, as done in the ModeS corpus (Guerrero Nieto & Saurí, 2012) (see next subsection).

Figure 4.6: *A sentence annotated in* TimeML *from* TimeBank *1.2.*

```
<S>
<SIGNAL sid="s95">For</SIGNAL>
<TIMEX3 tid="t94" type="DURATION" value="P40Y"
mod="EQUAL_OR_LESS" temporalFunction="false"
functionInDocument="NONE">nearly forty years</TIMEX3>
, the United States has
<EVENT eid="e26" class="REPORTING">said</EVENT>
categorically it would not
<EVENT eid="e99" class="I_ACTION">tolerate</EVENT>
totalitarian
<EVENT eid="e97" class="STATE">rule</EVENT>
in its own backyard.

<MAKEINSTANCE eventID="e26" eiid="ei431" tense="PRESENT"
aspect="PERFECTIVE" polarity="POS" pos="VERB"/>
<MAKEINSTANCE eventID="e99" eiid="ei432"
tense="NONE" aspect="NONE" polarity="NEG" pos="VERB"
modality="would"/>
<MAKEINSTANCE eventID="e97" eiid="ei433" tense="NONE"
aspect="NONE" polarity="POS" pos="NOUN"/>

<TLINK lid="l17" relType="DURING"
eventInstanceID="ei431" relatedToTime="t94"
signalID="s95"/>
<SLINK lid="l41" relType="EVIDENTIAL"
eventInstanceID="ei431" subordinatedEventInstance="ei432"/>
<SLINK lid="l42" relType="FACTIVE"
eventInstanceID="ei432" subordinatedEventInstance="ei433"/>
</S>
```

4.1.3 TimeML Across Languages

The English TimeBank corpus, annotated with TimeML, was developed as the gold-standard corpus for corpus-based research on the linguistics of time (Pustejovsky, Castaño, et al., 2003). Since the initial applications, several adaptations have been made to TimeML to facilitate annotating different languages than English. TimeML annotation was adapted to cater to the grammar of Italian (T. Caselli, 2010), German (Spreyer & Frank, 2008), Korean (Im, You, Jang, Nam, & Shin, 2009), Romanian (Forăscu, Ion, & Tufiş, 2007), French

(Bittar, 2011), Portuguese (Costa & Branco, 2012), Spanish (Saurí & Badia, 2010) and Catalan (Saurí & Badia, 2012).

Most of these adaptations to TimeML were used to annotate corpora for different languages. The French TimeBank (Bittar, 2011), Spanish TimeBank (Saurí & Badia, 2010), and Catalan TimeBank (Saurí & Badia, 2012) are made up of annotated newspaper articles, like the original TimeBank corpus, but containing different articles. Spanish TimeML was also used to create the ModeS TimeBank (Guerrero Nieto & Saurí, 2012), which contains annotated historical texts in 18th century Spanish, describing a sea crossing. The Ita-TimeBank, made up of annotated Italian texts, is under development and will soon be released (V. Caselli T. Bartalesi Lenzi, Sprugnoli, Pianta, & Prodanof, 2011).

Translating the existing corpora is arguably cheaper and easier than annotating completely new texts. Some corpora are translated versions of annotated English texts, like the Romanian Ro-TimeBank (Forăscu & Tufiş, 2012), a direct translation of the annotated TimeBank 1.2, and the Portuguese TimeBankPT, a translation of the annotated data of the first TempEval challenge. An algorithm to map the tags onto a German parallel corpus has been developed, but so far it has not been used to create a corpus (Spreyer & Frank, 2008).

The availability of these cross-linguistic corpora make the TimeBank corpus a good candidate to address the problem of this thesis statistically. Any methods used can easily be adapted to analyse other TimeML corpora regardless of language.

4.2 The TimeBank Corpus

The TimeBank corpus version 1.0 was the first implementation of TimeML, and it has received numerous updates since. TimeBank was envisioned as a gold-standard human-annotated corpus marked up for events and temporal relations and expressions (Pustejovsky, Hanks, et al., 2003). As TimeML was updated, so were the annotations of the TimeBank corpus. Version 1.2 of the TimeBank corpus uses TimeML version 1.2.1.

Documents in the TimeBank corpus come from a variety of sources, namely ABC, CNN, PRI, VOA, NYT, and the Wall Street Journal. In the initial phase of TimeBank, these were annotated by five different annotators who also took part in the development of TimeML 1.0. Some events and signals, and some of the event attributes, were tagged automatically in the preprocessing phase, and human annotators then completed the annotation by checking the results and adding additional signals, events, time expressions, and the appropriate links. Annotation took roughly 1 hour for a 500-word document (Pustejovsky, Hanks, et al., 2003). Subsequent phases of annotation involved researchers updating these early versions to reflect the changes in TimeML, up to the current version, and checking the accuracy and completeness of the annotations (Pustejovsky, Littman, Saurí, & Verhagen, 2006).

The TimeBank corpus has been used to develop numerous algorithms for automatic annotation in TimeML, which were described in section 3.3.3 in the previous chapter. Most notably, the TARSQI toolkit was developed, which automatically annotates a text with events and the relations between them.

While the original intended goal was to develop a question answering system which could handle narrative, to my knowledge no such application exists as of yet.

4.2.1 Corpus Statistics

Version 1.2 of the TimeBank corpus contains 183 news articles which have been annotated with events, times, and temporal relations following the TimeML 1.2.1 specification (Pustejovsky et al., 2006). The count of each of the TimeML tags is listed in figure 4.5.

Table 4.5: TimeBank *1.2 corpus statistics.*

TAG	COUNT
EVENT	7935
MAKEINSTANCE	7940
TIMEX3	1414
SIGNAL	688
ALINK	265
SLINK	2932
TLINK	6418
Total	27592

The utility of the TimeBank corpus version 1.1 was analysed by Boguraev and Ando (2005) from the perspective of a reference corpus usable as a resource to automatically annotate other corpora. They notice some qualitative problems with inconsistency in the annotation, and quantitative problems concerning the size of the corpus. Their comments were taken into account in version 1.2 of TimeML, but some of the problems they draw attention to still apply.

Most notably, the TimeBank corpus is relatively small, since it was originally not envisioned as reference corpus, but more as an exercise in applying the annotation guidelines. With 183 documents containing 18.6 thousand tokens, and no training or test portions, the TimeBank corpus is an order of magnitude smaller than other standard training corpora like the Penn Treebank corpus, which contains more than one million words (Boguraev & Ando, 2005). This seems a problem considering the large quantity of variables which are annotated, and makes whatever inconsistencies are still present in the corpus much more of a problem.

Another quantitative problem Boguraev and Ando (2005) lay out is that counting the different event and relation types shows that they are highly unevenly distributed. This is still applicable in version 1.2.1. The event class statistics shown in table 4.6 show that some event types are much more frequent than others. The same goes for temporal and subordination relations, as shown in table 4.7 and table 4.8. Aspectual relations, shown in table 4.9, are relatively more evenly distributed, but are themselves rare enough to pose problems. Tense, shown in table 4.10, is slightly more evenly distributed, allowing for statistical analysis. Aspect, unfortunately, is heavily biased, as shown in table 4.11. The number of non-verbs which have tense and aspect in this context are nouns and adjectives marked with event tags which occurred in a verb phrase, and were not considered in this research.

Table 4.6: TimeBank *1.2* ***EVENT class*** *statistics.*

CLASS	COUNT
OCCURRENCE	4215
STATE	1117
REPORTING	1028
I_ACTION	681
I_STATE	584
PERCEPTION	48
Total	7935

Table 4.7: TimeBank *1.2* ***TLINK relType*** *statistics.*

RELATION TYPE	COUNT
BEFORE	1408
IS_INCLUDED	1357
AFTER	897
IDENTITY	743
SIMULTANEOUS	671
INCLUDES	582
DURING	302
ENDED_BY	177
ENDS	76
BEGUN_BY	70
BEGINS	61
IAFTER	39
IBEFORE	34
DURING_INV	1
Total	6418

Table 4.8: TimeBank *1.2* ***SLINK relType*** *statistics.*

RELATION TYPE	COUNT
MODAL	1276
EVIDENTIAL	1160
FACTIVE	397
COUNTER_FACTIVE	48
CONDITIONAL	45
NEG_EVIDENTIAL	6
Total	2932

Table 4.9: TimeBank *1.2* ***ALINK relType*** *statistics.*

RELATION TYPE	COUNT
INITIATES	94
CULMINATES	30
TERMINATES	55
CONTINUES	70
REINITIATES	16
Total	265

Table 4.10: TimeBank *1.2 tense statistics.*

TENSE	OVERALL	VERBS
NONE	2793	240
PAST	2152	2064
PRESENT	1415	1244
INFINITIVE	782	754
PRESPART	361	355
FUTURE	284	268
PASTPART	152	151
Total	7940	5076

Table 4.11: TimeBank *1.2 aspect statistics.*

ASPECT	OVERALL	VERBS
NONE	7315	4486
PERFECTIVE	415	381
PROGRESSIVE	190	189
PERFECTIVE_PROGRESSIVE	20	20
Total	7940	5076

The subset of verbs in the corpus events warrant additional attention, since these are the items on which the classifier will be trained. Table 4.12 shows that, like the tense and aspect themselves, the distribution between tense and aspect is heavily biased. Since none is by far the most frequent aspect, it makes sense that every tense occurs far more often with this aspect than any other. The statistics, however, show an even stronger tendency than that: the vast majority of the verbs with other aspects occur in present tense, and those which do not are mostly in past tense. This results in the majority of tense-aspect combinations having very few or even zero values.

Table 4.12: TimeBank *1.2 statistics of tense and aspect combination frequencies in verbs.*

	none	perfective	progressive	perfective progressive	*Total*
past	1961	83	19	1	2064
present	795	270	162	17	1244
infinitive	754	0	0	0	754
prespart	354	1	0	0	355
future	258	5	5	0	268
none	215	20	3	2	240
pastpart	149	2	0	0	151
Total	4486	381	189	20	5076

While problematic, it can be expected that some of the relation types, tenses, aspects, and tense-aspect combinations are less frequent than others. Their use in language is generally not subject to an equal distribution; for example, it can be assumed that the imperfective aspect is more commonly used than the

perfective, because the latter refers to a more specific temporal relation which might be used less often in a narrative. The same goes for the `BEFORE` relation as compared to `BEGINS`: most texts describe more events which simply occur before another event than events which begin another event. Such trends are amplified in a contained domain such as newspaper texts, because the texts are written according to a certain format, with specific tendencies in both style and the type of information that is described.

The low frequencies of some of the event classes and relation types does pose a possible problem in classification, since it could make low-frequency values virtually useless in a corpus-based approach. These problems therefore have to be accounted for in the research, meaning that most likely the model will benefit from some low-frequency information being ignored or simplified.

On the other hand, the low frequencies of most of the tense-aspect combinations is somewhat of a double-edged sword. On the downside, it completely rules out the generation of rare constructions, even if they are theoretically feasible. There is no linguistic reason a sentence like "He *will have been postponing his grocery shopping* for five years tomorrow" should not be generated, but since there is no instance of a future perfective progressive in the corpus, it cannot be generated statistically based on the available tense-aspect combinations. On the upside, this will make generation of the aspects and combinations which do occur frequently easier. For example, by excluding certain combinations, the generator will be more likely to select a frequently occurring combination, which is generally more likely to be the best choice in natural language. In short, while the lack of rare constructions will limit the versatility of an algorithm trained on these data, it might result in an increase in overall accuracy.

Chapter 5

Methods

The TimeBank corpus contains much information which is theoretically usable from a generation perspective, like event class, lexical information, and temporal relations between events. I will attempt to generate the tense and aspect based on this information, using machine learning methods, and evaluate the results by comparing the results of generation to the tense and aspect of the original verb phrase. Three different systems will be created, one to generate tense, one to generate aspect, and one to generate combinations of tense and aspect. This will shed light on whether statistical generation of tense and aspect based on these criteria is feasible, what features are actually useful in this task, and which algorithm is the best for this task.

This section describes the hypothetical NLG system for which the tense and aspect generation algorithm was designed, with a theoretical consideration of what information would be available to perform the task, along with the software, machine learning algorithms, and feature selection algorithms that were used, and what information was extracted from the TimeBank corpus to perform classification. The next chapter contains the obtained results.

5.1 A Hypothetical Narrative NLG system

The purpose of this study is to develop an algorithm which selects tense and aspect for an NLG system which generates narrative texts. It is worthwhile to consider what such a system might look like, and what information will be available to perform this task. The task of tense and aspect generation in such a system can be considered as an independent part of the microplanning module.

The theoretical NLG system takes a set of events, times, and temporal and aspectual relations that hold between them as its input, and represents them in a coherent text. Every event has one of the TimeML event classes and a polarity. Times to which events relate can be considered to come in a format similar to timexes, with most of the them presented as concrete, grounded points in time. The system's input also contains the creation time of the document, in similar format, though that may be derived automatically from the time at which the system is run. Temporal and aspectual relations are denoted in the input as links between two events, with a TimeML temporal or aspectual `LINK`. Temporal relations can also hold between an event and a time. In essence,

this creates a set of data analogous to the event instances (with event data), timexes, temporal links, and aspectual links from TimeML, but without the content phrase, tense, aspect, or modality, though with semantic information.

For the purpose of this study I will presume a document planner can be used to structure the data in a document plan, although document planning of narratives is an interesting problem in its own right. The document plan takes the form of an ordered hierarchy of events and times. This functions as the input for the microplanner. Using the pipeline architecture of Reiter and Dale (2000), the microplanning module consists of the tasks of lexicalization, aggregation, and referring expression generation. Maintaining the pipeline, tense and aspect generation has to be performed at some point alongside those tasks.

No matter where in the microplanner tense and aspect generation takes place, it has the event classes, polarities, semantic content, temporal, and aspectual links available from the document plan. This is not quite enough for tense and aspect generation, however, since the events by themselves have not been assigned a part-of-speech yet. The lexicalization module will determine whether the event should be represented as a verb, noun, or an adjective. An event does not necessarily need to be represented as a verb to have tense and aspect. Events represented as nouns, for example, can use a copula, which will have to be assigned tense and aspect, as in the sentence "When the area behind the dam falls, it *will be a lake*", in which 'the lake' functions as a state. This is beyond the scope of the current research, however, so I will assume only verbs need to be assigned a tense and an aspect. Nevertheless, tense and aspect generation will have to take place after lexicalization, so that it knows which events to generate tense and aspect for.

Placing the task after lexicalization also provides additional sources of information. Lexicalization generates the base lexeme for the verb, a possibly relevant variable, and could also introduce signal words and explicit modal auxiliaries for verbs. Aggregation can also provide additional knowledge, by sorting the information in sentences and introducing subordination, which could also be relevant for tense. Referring expression generation is a task comparable to tense and aspect generation, but they ought not be particularly relevant to one another. I therefore suggest placing tense and aspect generation after lexicalization and aggregation, to have access to as much additional information as possible.

This theoretical NLG system also has a module which generates temporal expressions, a staple for any narrative. I will make few assumptions regarding the level of specification of temporal expressions at this point, but if both the times and the event relations are represented in the document plan, and aggregation has been completed, information is available on what sentences will express a time, and also what type of temporal expression it is. Hence, I can include these as possible variables in the tense and aspect generation module of this system, alongside the temporal relation type to the creation time.

All in all, this leads to a large selection of information which can be used to perform tense and aspect generation. Not all of it might be as useful for generation, nor easy to provide in an actual generation system, but it is worth investigating which of these variables are the most useful. By extracting the relevant event information from the TimeBank corpus, along with the tense and aspect used for the event verb phrase, machine learning methods can be used to 'predict' the tense and aspect, making it into a straightforward classifi-

cation task. The different sources of information described and where it can be extracted from in the TimeBank corpus is reported in table 5.1.

Table 5.1: *Temporal information in an* NLG *system and its analogous* TimeBank *information.*

MODULE	INFORMATION	ANALOGUE
System input	*Event class, polarity, and content.*	`EVENT`, `EVENTINSTANCE`
	Temporal relations between events.	`TLINK`
	Aspectual relations between events.	`ALINK`
	Relations of events to speech time.	`TIMEX3` (`'CREATION_TIME'`)
	Concrete times, dates, and durations.	`TIMEX3` (other functions)
Document Planning	**Content Determination**	
	What time expressions are used.	`TIMEX3`
	Document Structuring	
	Order of events and time expressions in text.	`EVENT` and `TIMEX3` order
Microplanning	**Lexicalization**	
	Event part-of-speech.	`MAKEINSTANCE pos`
	Event lexemes.	`EVENT` content
	Signal words to describe relations.	`TLINK signal`
	Words used for event modality.	`MAKEINSTANCE modality`
	Aggregation of messages	
	Sentence position of events and time expressions.	`EVENT` and `TIMEX3` sentence position
	Event subordination relations.	`SLINK`
	Tense and Aspect Generation	
	The described module, which uses all the abovementioned information.	
	Temporal Expression Generation	
	Probably uses similar information.	
	Referring Expression Generation	
Surface Realization	**Structure Realization**	
	Linguistic Realization	

5.2 Machine Learning Methods

The problem of tense and aspect generation can be seen as a classification task. Each of the event instances in the text is an *instance* of a certain *class* (the class being the tense or aspect depending on what is being modelled), and the *classifier* is an algorithm which determines the class of an instance, depending on the values of its *features*. Event instances in the TIMEBANK corpus are annotated with feature information, and have either a tense, an aspect, or a tense-aspect combination as their class. Machine learning methods can be used to train and evaluate a classifier for tense and aspect, based on the available features of each instance. The values for these features, the contextual information in this case, can be extracted from the TIMEBANK corpus and labelled. Since the classes themselves are also annotated in the corpus, the task is one of supervised learning.

There are numerous strategies for supervised learning. The main two choices which have to be made are what features to use for the system, and what algorithm works best. The classifier's evaluation is based on the accuracy and F-measure of the prediction, two measures which will briefly be described in section 5.4. Several techniques can be used to evaluate the accuracy of a classifier; I will use 10-fold cross-validation, in which the training set is divided into ten subsets of equal size, and each subset functions as a test set once, with the union of all other subsets used for training. The overall accuracy is the average of the accuracies found by each of the evaluations over the test sets. Kotsiantis, Zaharakis, and Pintelas (2006) give an excellent overview of classification techniques and evaluation.

To create and evaluate a classifier, I use the WEKA (Waikato Environment for Knowledge Analysis) data mining software, version 3.6.0 (Hall et al., 2008). WEKA is a unified workbench which allows easy access to state-of-the-art machine learning techniques. It allows quick selection and comparison of different machine learning methods on data sets, including many algorithms for classification, through a graphical user interface. A dataset with a large number of features can be loaded into the interface, after which the ones which need to be used for classification can be selected, allowing testing of the effectiveness of different features. The first version of WEKA was developed in 1992, and with numerous versions released since, it has achieved widespread acceptance in the field of machine learning (Hall et al., 2008). WEKA allows classification through multiple types of methods, of which decision trees and rule learners are of interest in this research the most because of their comprehensibility and adaptability to NLG systems.

5.2.1 Decision Trees

Decision trees are trees that classify instances by making decisions based on feature values. For a general overview of decision trees and their functionality, see Murthy (1998). Decision trees are based on the assumption that different classes differ in at least one of their features (Kotsiantis et al., 2006). A decision tree contains a set of *internal nodes*, which contain a branch of *child nodes* each signified by a particular feature value of the instance. The *leaf nodes*, the final nodes of the tree, contain a class for that instance (Murthy, 1998). Each internal node represents a particular feature in an instance to be classified, and

its branches each represent a value for that feature. Classification of instances starts at the root node, and splits based on a single feature at every node.

Decision trees provide an intuitive and elegant strategy for classification. One of their most useful traits is their comprehensibility; it is very straightforward to see how a decision tree performs classification. This means that they can easily be adapted into rules for an NLG system to follow. This strategy has been applied before in numerous NLG applications. For example, Cheng, Poesio, Henschel, and Mellish (2001) used decision trees to determine NP modifiers in museum descriptions, and then implemented those trees as a module in the ILEX system (Oberlander, O'Donnell, Knott, & Mellish, 1998), which adaptively generates online descriptions of museum objects. Dale and Viethen (2009) and Viethen (2011) use the WEKA J48 decision tree classifier with the TUNA corpus to learn the correspondences between referring expressions and corpus data. In addition to resulting in usable systems, these studies grant insights into the heuristics of corpus-based features.

The problem of constructing an optimal binary decision tree consists of finding the feature which best divides the training data at the root node, and then repeating that task for the subset of the data at each internal node (Kotsiantis et al., 2006). This has to be resolved unless all the instances in the subset at a certain node belong to the same class, in which case a leaf node can be created for that class. Constructing an optimal decision tree is an NP-complete problem. Several strategies have been applied to construct near-optimal trees, but studies have concluded that there is no single best method (Murthy, 1998).

One of the most famous algorithms for building decision trees is the C4.5 algorithm (Kotsiantis et al., 2006; Quinlan, 1993). It chooses one feature which splits its subset of instances most efficiently at every node, and automatically creates a subtree for all of the feature values. This method is efficient and requires little time. Lim, Loh, and Shih (2000) evaluate a large number of classification algorithms, and conclude that C4.5 has a very low error rate, and is one of the fastest algorithms they tested. As such, it makes a very efficient classification algorithm. An open implementation of C4.5 named J48 is used in WEKA, and will be used as a main method in this research.

If decision trees are automatically generated based on training data, it is possible that the tree performs well on the training data, but has problems with accounting for unseen data. In this case, the tree is *overfit*, that is, too specifically geared toward the training data (Kotsiantis et al., 2006). *Pruning* is a strategy to prevent overfitting, by removing branches of the tree which add little to the accuracy of the evaluation set. This essentially trades in accuracy of the model with respect to the training data, in the hopes that a simpler tree will give a better account of unseen instances in the test data. It is beneficial in some learning situations, dependent on the complexity of the decision task (Elomaa, 1999). Different decision tree algorithms have different pruning methods. The ones in WEKA generally perform pruning automatically, with the option to disable it. *Grafting* is complementary to pruning, and adds branches at points in the tree where evaluation shows a leaf does not bear good results (Webb, 1999). The J48GRAFT algorithm in weka is an implementation of C4.5+, a modification to C4.5 with grafting added.

While J48GRAFT takes more time than J48 when confronted with the data, both produce excellent results in little time when compared to. They therefore represent the main strategies by which classification is performed. The

performance of J48 and J48GRAFT is shown and compared in chapter 6.

5.2.2 Rule Learners

A rule learner functions by searching for a rule which classifies a part of the instances, then continues searching for a rule to classify a part of the remaining instances, until all instances are accounted for. Fürnkranz (1999) provides an overview of rule-based methods in classification. The difference between rule learners and decision trees is that, while decision trees evaluate the quality of each of the subtrees at a feature value node, rule learners only evaluate the quality of the instances covered by the rule. A rule learner therefore needs to be able to generate rules with a high reliability (Kotsiantis et al., 2006). Rule learners, like decision trees, are attractive because of their high comprehensibility.

One well-known rule generation algorithm is RIPPER (Cohen, 1995). It functions by repeating a *grow phase*, which adds conditions to a rule until it has 100% accuracy, and a *prune phase*, which removes superfluous conditions to prevent overfitting. An implementation of RIPPER is present in WEKA as JRIP. Another well-known system present in WEKA is PART (Frank & Witten, 1998), which functions by generating partial decision trees and extracting a single rule from that tree. This combines the benefits of both rule-based and decision tree methods. Unlike C4.5 and JRIP, PART introduces one rule at a time, and avoids post-processing. PART functions better than JRIP on the data used, and its results are on par with the ones resulting from the decision trees, so it will be used as a rule-based alternative to J48 and J48GRAFT.

5.3 Data Preparation

To use machine learning methods, the information from the TIMEBANK corpus must be in an appropriate format for WEKA to handle. Every instance with tense and aspect, that is, every verbal event instance, has to be presented as a set of features with a tense or aspect class. The features of every instance were designed to mirror the information available in a narrative NLG module, as described in section 5.1. Hence, I created an *instance file* containing all the event instances in the TIMEBANK corpus, with their qualities and relations as features. Running the full program to extract the information from the corpus and generate the .arff file takes roughly one minute. The resulting instance file contains 5077 instances in total.

The task of creating the instance file can be split up in two stages. Both were performed with a single program written in Python. In the first stage, I extracted all the relevant event information from the TIMEBANK CORPUS. Iterating over all documents in the corpus, I added every event instance to a dictionary of event instances, every timex to a dictionary of timexes, and every link to a dictionary of links. Every document in the corpus hence was assigned a triple of dictionaries. This granted an abstract representation of all the information in any given document.

In the second stage, I generated an instance file. The types of features (which need to be mentioned explicitly in .arff format[1]) were added first, and

[1] .arff is a text format used to represent features and classes in an attribute/value notation, not unlike CSV. A description of the version of .arff used in this research is given at

then for every event instance in every document, I drew information from the dictionaries generated in the first stage. I then added the feature values and class for every instance to the instance file.

In the first stage of the program, for every document in the TimeBank corpus, the following information was extracted:

1. Events and their event class, along with their position in the document relative to other events, and the number of the sentence they occur in within the document;
2. The content of the events (the words tagged with them);
3. The content of the signals;
4. The event instances and their part-of-speech, polarity, modality, tense and aspect;
5. Times and the position of the sentence they occur in within the document;
6. For every individual event instance, dictionaries containing:
   a. The event instances to which it has a temporal relation, the relation type, and, if applicable, the signal;
   b. The event instances to which it has a subordination relation and the relation type;
   c. The event instances to which it has an aspectual relation and the relation type;
   d. The times to which it has a relation, the relation type, and the type of the time.

   Two more variables were obtained by processing the data:
7. The lemma of the event content. These were obtained by running the morpha morphological analyser (Minnen, Carroll, & Pearce, 2001) on the event content;
8. The root event of each sentence, which match the root word in a dependency tree. These were obtained by converting the TimeML files to plain text, parsing the file with the lexicalized dependency parser implemented in the Stanford Lexicalized Parser version 2.0.3 (Klein & Manning, 2003), and extracting the roots from the resulting dependency structures. The event instances were matched with the root by iterating over the sentences recognized by the parser and the ones annotated in the corpus, and comparing the content word of each event to the root word. If the content of multiple events in a sentence matched the root word, a logical algorithm was used to select the correct one. Because the sentence structure annotated in the corpus did not always adhere to that recognized by the parser, some minor modifications had to be made to the TimeBank corpus (mostly adding full stops and sentence markers) to align the sentences. Not every sentence has a root event, because some of them are not actually grammatical sentences.

`http://www.cs.waikato.ac.nz/ml/weka/arff.html`

Each temporal relation between events was added to the dictionary twice: both as it was marked in the corpus and in reverse. As such, for the following `TLINK`, relations 1 and 2 would both be added to the dictionary:

```
<TLINK lid="l1" relType="BEFORE" eventInstanceID="ei10"
relatedToEventInstance="ei12"/>
```

1. `ei10 BEFORE ei12`
2. `ei12 AFTER ei10`.

Each link between relations has an inverse. The inverses of all relation types are all shown in table 5.2.

Table 5.2: *`TLINK` reversals.*

RELATION TYPE	REVERSAL
BEFORE	AFTER
AFTER	BEFORE
INCLUDES	IS_INCLUDED
IS_INCLUDED	INCLUDES
DURING	DURING_INV
DURING_INV	DURING
SIMULTANEOUS	SIMULTANEOUS
IAFTER	IBEFORE
IBEFORE	IAFTER
IDENTITY	IDENTITY
BEGINS	BEGUN_BY
ENDS	ENDED_BY
BEGUN_BY	BEGINS
ENDED_BY	ENDS

The second stage served to generate a list of instances and their feature values, which was written to an .arff file, by performing specific operations to obtain or compute the different feature values. The features for every instance, and the information which is represented by them, are shown in table 5.3. The columns in this table show the feature, a description, the percentage of instances in the text for which this feature is not set to 'none', the number of distinct feature values, and the amount of feature values which occurred only once. Each type of temporal relation was also used to add two eventDistance features, also shown in the table.

Table 5.3: *The features which were used for each instance, derived from the event instance `eii10` of event `e1`.*
*Each feature is described along with its rate of assignment, or the percentage of the instance which had a non-empty value for this feature (*A*), the amount of distinct values the feature takes (*D*), and the amount of those values which is unique, or occurred only once in the corpus (*U*).*

FEATURE	DESCRIPTION	A	D	U
`word`	The lemma of the word contained in `e1`.	100%	916	393

class	The event class of eii10.	100%	7	0
polarity	The polarity of eii10.	100%	2	0
speechtime	The relation type of the TLINK eii10 has to the document creation time, if that TLINK exists.	18.85%	6	0
modality	Boolean about the presence of modality in eii10.	100%	2	0
modalityval	The value of the modality in eii10, if any.	4.79%	16	6
timebool	Boolean about the presence of a TIMEX3 in the same sentence.	100%	2	0
timeinsentence	The type of relation eii10 has to a time in the same sentence, if any.	12.88%	11	0
timetype	The type of that time.	12.88%	5	0
signalbool	Boolean about the presence of a TLINK with a signal.	100%	2	0
signal	The signal used if eii10 has a TLINKs with a signal.	9.57%	45	14
signalrel	The relation type of the TLINK with the signal.	9.57%	16	3
has_subordination	The type of subordination relation eii10 has, if any.	44.67%	7	0
is_subordination	The type of subordination relation another event instance has to this one, if any.	15.86%	6	0
has_aspectual	The type of aspectual relation eii10 has, if any.	4.43%	6	0
has_aspectualbool	Boolean about the presence of an aspectual relation eii10 has	100%	2	0
is_aspectual	The type of aspectual relation another event instance has to this one, if any.	0.75%	6	2
is_aspectualbool	Boolean about the presence of an aspectual relation that another event has to eii10.	100%	2	0
eventneighbour-	The type of temporal relation eii10 has to the event instance mentioned directly before it in the text, if any.	16.15%	15	0
eventneighbour+	The type of temporal relation eii10 has to the event instance mentioned directly after it, if any.	18.28%	15	0

closestrelation-	Like eventneighbour-, but iterates to the beginning of the document until it finds the closest event to which it has a temporal relation.	33.68%	15	0
closestrelation+	Like eventneighbour+, but iterates to the end of the document until it finds the closest event to which it has a temporal relation.	30.88%	15	0
sentenceneighbour-	The type of temporal relation eii10 has to the root of the previous sentence, if any.	4.55%	8	0
sentenceneighbour+	The type of temporal relation eii10 has to the root of the next sentence, if any.	5.14%	11	2
sentenceclosest-	Like sentenceneighbour-, but iterates backward like eventneighbour-.	8.21%	9	1
sentenceclosest+	Like sentenceneighbour+, but iterates forward like eventneighbour+.	7.45%	12	3
sentencerel0	The type of temporal relation eii10 has to the root of the sentence it is in. IDENTITY if eii10 is the root.	46.43%	15	2
eventBEFORE-	A numeric feature: the number of events between eii10 and the closest event instance before it which ei10 has a relation of type BEFORE to.	5.5%	19	7
eventBEFORE+	A numeric feature: like eventBEFORE-, but looks ahead instead of backward.	8.9%	19	7
eventAFTER-	Like eventBEFORE-.	7.78%	19	5
eventAFTER+	Like eventBEFORE+.	7.29%	19	10
eventINCLUDES-	Like eventBEFORE-.	0.96%	11	5
eventINCLUDES+	Like eventBEFORE+.	2.23%	17	9
eventIS_INCLUDED-	Like eventBEFORE-.	4.53%	27	14
eventIS_INCLUDED+	Like eventBEFORE+.	2.38%	14	4
eventDURING-	Like eventBEFORE-.	0.57%	5	0
eventDURING+	Like eventBEFORE+.	0.22%	4	1
eventDURING_INV-	Like eventBEFORE-.	0.18%	19	10
eventDURING_INV+	Like eventBEFORE+.	0.43%	3	1
eventSIMULTANEOUS-	Like eventBEFORE-.	8.35%	26	8
eventSIMULTANEOUS+	Like eventBEFORE+.	7.76%	24	12
eventIAFTER-	Like eventBEFORE-.	0.55%	11	6
eventIAFTER+	Like eventBEFORE+.	0.3%	6	3
eventIBEFORE-	Like eventBEFORE-.	0.22%	6	3

eventIBEFORE+	Like eventBEFORE+.	0.49%	11	7
eventIDENTITY-	Like eventBEFORE-.	4.69%	37	13
eventIDENTITY+	Like eventBEFORE+.	7.13%	37	17
eventBEGINS-	Like eventBEFORE-.	0.2%	5	1
eventBEGINS+	Like eventBEFORE+.	0.28%	4	2
eventENDS-	Like eventBEFORE-.	0.67%	7	3
eventENDS+	Like eventBEFORE+.	0.12%	4	1
eventBEGUN_BY-	Like eventBEFORE-.	0.28%	3	1
eventBEGUN_BY+	Like eventBEFORE+.	0.32%	4	1
eventENDED_BY-	Like eventBEFORE-.	0.08%	3	1
eventENDED_BY+	Like eventBEFORE+.	0.06%	6	3
tense	The tense of ei10.	95.27%	7	0
aspect	The aspect of ei10.	11.64%	4	0
tenseaspect	The tense and aspect combination of ei10.	95.77%	20	2

The features represented in table 5.3 denote an excessive amount of information. Many types of information were added in different ways, to see what data and what representation of the data is most effective for generation. The features in the list are all available for the machine learning task, but not all of them need to be used. They represent strategies to perform classification. How to determine what features are the most valuable is discussed in subsection 5.3.3.

The selection of these features can be justified in light of the hypothetical NLG system discussed in section 5.1. Many of the features, such as `class` and `speechtime`, are quite straightforward to obtain in the NLG system. Some others, namely the `has_aspectual` feature and its counterparts, are directly available from the input of the system, which includes which events have aspectual relations to which. Other features, namely `word`, `signal`, and `modalityval`, can be presumed to have been generated in the lexicalization module. The document planner and aggregation module order events and times and organize them in sentences. This allows measurement of the distance between two events, as performed by the `eventneighbour`, `closestrelation`, `sentenceneighbour`, and `sentenceclosest` features. These features represent different strategies of modelling the temporal context of an event, performed at different levels of granularity, the details of which are discussed in subsection 5.3.2. The division in sentences also allows features which are concerned with what expressions are present in the event in which the sentence occurs, such as `timeinsentence`, and the relation an event has to the grammatical root of this sentence, as in `sentencerel0`. As such, these features are all theoretically available for the task of NLG, and can be used to generate decision trees and rules for the system.

5.3.1 Simplification of Information

Extracted information about features like the modality of a verb and the signal which denotes a relation often represents very detailed knowledge. As shown in table 5.3, there are 16 distinct values the `modality` feature can have, 6 of which occur only once. This includes phrases like 'having to', and in some cases nonsense for this purpose, such as 'delete'. It might therefore be the case that

this information is too specific for the hypothetical NLG system to be able to provide at the point this module is run.

The same problem exists for `signal`, which has 45 distinct values, 14 of which are unique, like 'at least until' and 'in anticipation of'. Basing a classifier on feature values which occur only once or twice is likely to result in overfitting, since the one instance with this value will be classified by that value regardless of other features, without any reliable evidence that this is actually a good strategy. If the same algorithm is used to classify previously unseen instances which use the same signal, this likely will not result in accurate classification. Moreover, it might not be realistic from a generation perspective to assume such specific signals and modal verbs are a readily available resource.

A possible solution for this problem is to simplify the values. For `signal`, this can be done by not including the signal word itself, but instead the relation it denotes, resulting in a simplified 'counterpart' feature which is based on the same corpus information, `signalrel`. It is more likely that the NLG system has already decided which relations will be expressed through signals at the point of tense and aspect generation, than that it has already selected a phrase to denote the signal. Alternatively, since these cases might still be too rare and result in overfitting, the feature could be simplified even further, containing only whether it has a temporal relation with a signal or not.

The modality cannot be simplified to a type of relation, because it is an attribute of the event instance itself. Therefore, representation as a boolean is the only option to simplify the value of modality. The same limitation applies to `has_aspectual` and `is_aspectual`, which is extracted from tags that only denote a single aspectual relation. Therefore, these three features were given a simplified boolean counterpart feature.

Simplification to booleans is useful to increase accuracy for relatively rarely occurring linguistic phenomena such as modality and aspectuals. Even though they do not contain unique values, boolean counterparts were added as an option for the relatively rare timex features as well. This results in these features being represented in three different ways, one which denotes the type of timex, one which denotes the type of relation to it, and one boolean about their presence. Additional candidates for boolean counterparts would be the `sentenceneighbour-`, `sentenceneighbour+`, `sentenceclosest-`, and `sentenceclosest+` features, which are all quite rarely assigned a value, and some of which sometimes have unique feature values. The current version of the instance file does not contain booleans for these features, though their use would be interesting to investigate.

Another set of features which can be simplified is those which represent information about temporal relations. As shown in table 4.7 in section 4.2.1, the values of these features are distributed very unevenly. Since temporal relations are transitive and some of the relations are largely similar, this can be resolved by folding the less frequent types into the more frequent less specific ones. The types of temporal relations which can be subsumed by larger sets are shown in table 5.4, and the resulting frequencies of the different relation types are shown in table 5.5.

Table 5.4: *TLINK relType simplification.*

RELATION TYPE	SUPERSET
DURING	IS_INCLUDED
DURING_INV	INCLUDES
IAFTER	AFTER
IBEFORE	BEFORE
BEGINS	BEFORE
BEGUN_BY	AFTER
ENDS	AFTER
ENDED_BY	BEFORE

Table 5.5: *Simplified TLINK statistics.*

RELATION TYPE	COUNT
BEFORE	1680
IS_INCLUDED	1659
AFTER	1082
IDENTITY	743
SIMULTANEOUS	671
INCLUDES	583
Total	6418

5.3.2 Representation Strategies for Temporal Relations

One of the more difficult parts of presenting the information present in the TimeBank corpus is translating the temporal relations to features. Unlike aspectual and subordination relations, of which an event can only have one, it is possible for an event to have temporal relations of different types to multiple events in the text. I applied three different strategies to represent `TLINK` information in the feature file. For each strategy, the relations to an event instance which occurs later in the text are added separately from the ones to events which occur earlier in the text, because, due to the iconicity assumption, it cannot be assumed that the relations are processed the same regardless of their position.

The first strategy, *event recognition*, is represented by the `eventneighbour` and `closestrelation` features. The `eventneighbour` features are obtained by a simple mechanism: the program checks whether an event instance has a `TLINK` to the event instance directly before it, and directly after it. The `closestrelation` feature does the same, but prioritizes having a non-empty value over remaining within its direct context. It therefore iterates over the events until it finds an event it has a relation to, both forward and backward. This strategy provides two ways of obtaining information about temporal relations without performing any selection.

The second strategy, *root recognition*, is represented by the `sentenceneighbour` and `sentenceclosest` features. The features resulting from the previous strategy do not distinguish syntactic information. As such, the events they refer to could be contained in a subordination or temporal expression, which would mean they have a different role in the narrative than other events. The current strategy uses a parser to retrieve the syntactic root

of the previous and following sentences, and marks it as a *root event*, which ought to contain the main event that the sentence denotes. This should result in a selection of events which have a more consistent function, and possibly makes it easier to give the current instance a virtual place in the narrative. The `sentenceneighbour` feature only considers the events occurring directly before and after it, whereas the `sentenceclosest` feature performs an iteration. The algorithm used for this strategy also allows the feature `sentencerel0`, which contains the temporal relation to the root of the sentence itself.

The theoretical drawback of these two strategies is that they can only consider a single relation. It is possible for an event istance to have multiple different relations. The third strategy, *relation recognition*, is an attempt to account for more relations, by representing all the different relation types the event instance has as separate features. This strategy is represented by the `eventBEFORE-` through `eventENDED_BY+` features in table 5.3, a set which will be collectively referred to as the `eventRELATION` features. Each relation type is assigned four numerical features. If the event instance has a relation of that type, the features are assigned the distance to the event it has that relationship with, in terms of the order of events in the document. This is measured both forward and backward, so if the 10th event in a document has the relation `BEFORE` to the 8th event in the same document, and the `INCLUDES` relation to the 12th event in the same document, the `eventBEFORE-` features is assigned the value -2, and the `eventINCLUDES+` feature is assigned the value 2. This strategy results in a more complete representation of the temporal relations of an event, but it results in a large amount of mostly unassigned features.

The feature structure allows for these three strategies to be applied separately or at the same time, providing different insights about the same situation. It is also possible to use one of the features which belong to a strategy, but not all of them.

5.3.3 Feature Selection

The performance of a classification system is largely dependent on the features used to perform the task. Feature selection is therefore an important part of optimizing performance. Although the C4.5 algorithm has an automatic feature selection algorithm built in, correlated and irrelevant features can degrade the performance. I attempted two different methods to perform feature selection: controlled ablation and contextual merit.

Ablation

The first feature selection method is controlled ablation. To optimize the performance of the system, I started out with the results of the full feature set, and systematically removed features, comparing each result to the last to improve the accuracy and F-measure. For this purpose, the full set of `eventRELATION` features was considered as one feature, since they generally did not improve the system much. It is not impossible that some of the `eventRELATION` features might be beneficial for the classifier combined with certain other features, but it has not been observed. The ablative method generally results in a contained feature set of 10 to 15 features, about half of the original set.

Through controlled ablation, a number of features were consistently removed. The `word` feature always weakened the performance of the algorithm, most likely due to its large number of unique values. Both the decison trees and rule learners trained on feature sets that included `word` based selection mostly on its value, largely obscuring the other features. Removal of the `eventRELATION` features also consistently resulted in an improvement of the results, probably due to the features' low rate of assignment, large range of values, and high number of unique values. These features are never included in the sets which result from this method.

Contextual Merit

The second method used for feature selection is the contextual merit algorithm of Hong (1997). This algorithm computes the features which are most likely to indicate the difference in different classes, by comparing each feature of an instance of a certain class to a large number of features of instances which do not belong to that class. Perner (2001) shows that limiting the feature set in this way is an effective way of increasing the performance of C4.5.

The contextual merit algorithm computes the value of a feature k based on the distance D_{ij} between two instances i and j. Assume there are N instances in the set, with N_s symbolic or categorical features and N_n numeric features in every instance, and $N_f = N_s + N_n$, the full set of features. The dataset, consisting of N instances with N_f features, can then be represented by an array of feature values of size N_f and a vector C of length N containing the class labels.

Contextual merit is determined by comparing all the features of an instance i with class C_i to the features of all instances which have any other class C_j. All instances with other classes together make up the set of 'counterexamples' of i, $\overline{C}(i)$. The total distance D_{ij} between an example instance i and one of its counterexamples j is determined as follows:

$$D_{ij} = \sum_{k=1}^{N_f} d_{ij}^{(k)}$$

In this formula, the component distance $d_{ij}^{(k)}$ denotes the difference in the feature value X_k of a feature (or 'component') k between instances i and j. For a symbolic feature, the component distance is a binary:

$$d_{ij}^{(k)} = 0 \text{ if } x_{ki} = x_{kj}$$
$$d_{ij}^{(k)} = 1 \text{ if } x_{ki} \neq x_{kj}$$

For a numerical feature, the component distance is dependent on the difference in value between the two features, as shown here:

$$d_{ij}^{(k)} = \min(|x_{ki} - x_{kj}|/t_k, 1)$$

The value t_k functions as a threshold for the relevance of the difference between the two feature values. If the difference between the two is equal to or greater than t_k, the component distance is simply set to 1. If it is smaller, the component distance is set to the proportion of the distance with respect to

t_k. The threshold t_k is a variable, and its value depends on the feature which is being evaluated. In this research, t_k is set to half the range of the feature values for each feature, which is the most commonly used strategy (Hong, 1997; Perner, 2001).

The total distance D_{ij} between instance i and counterexample j, defined before, is the sum of all component distances between them. Since the counterexamples which have too large a total distance with respect to instance i only provide noise to the data, only the fraction with the lowest total distance is used to compute the contextual merit. Out of the set of counterexamples $\overline{C}(i)$, the $K = \log_2 |\overline{C}(i)|$ instances with the lowest distance to i are retained. This is an arbitrary choice, but justified by experimental results (Hong, 1997).

The contextual merit m_k of a feature k in class C_i correlates to the amount by which the feature contributed to the total distance between i and j. Hence, the contextual merit is defined as such:

$$m_k = \sum_{i \in C_i} \sum_{j \in C_j} w_{ij} d_{ij}^{(k)}$$

In this formula, $w_{ij} = 1/D_{ij}^2$ if j is one of the K most closely related counterexamples, and $w_{ij} = 0$ if not.

This algorithm computes the contextual merit for every feature in every class in turn, determining which features are most influential to classify to that particular class. The computation of the contextual merit over multiple classes is treated as a series of two-class problems, with contextual merit for every class c in the set of possible classes P computed for $C_i = c$ and $C_j \in P \setminus c$. However, the features selected for the classification algorithm need to be optimal over the overall feature set, so the contextual merits for each feature are averaged into an overall contextual merit M_k using the following formula:

$$M_k = \sum_{c \in P} \frac{|c|}{N} m_k^{(c)}$$

In this formula, $|c|$ denotes the number of instances with class c, and N is the total number of instances. Hence, the classes which occur more often are given more importance, in the hope that this will maintain the overall accuracy of the contextual merit while accounting for all classes.

A well-known problem with contextual merit is that features with a large variety of values are automatically assigned higher scores than those with lower varieties. This was evident in the initially obtained results, where, for example, the feature `word` was assigned a very high merit. This was because every sparse or unique word would have a component distance of 1 to practically all other classes. This is fair in a normal classification setting, since in this case `word` is an unambiguous indicator of its class, but it does not reflect linguistic reality, and a high number of different lemmas is no indication that. Moreover, the system performs worse with a feature like `word` in it, because it is considered earlier in the algorithm than other features, most of which are then ignored afterward, despite not functioning well in the test set due to overfitting.

To avoid this problem, I used the methods outlined in Hong, Hosking, and Winograd (1995) to normalize the feature merits. In addition to calculating the normal contextual merits for each feature, I randomly permute the values

10 times, and compute the contextual merit of a feature set in which the normal feature assignment is replaced by the random permutation. Each feature merit can then be normalized by dividing it by the average of the merits of its randomized counterpart. This ensures that features which have a large merit purely by virtue of having many unique values are not overrepresented.

Normalization accounts for interaction between features to some extent. In the temporal spectrum, a lot of the information is closely related, and the same conclusions could be drawn from different features. This is already a problem when computing contextual merit, since information which is represented in different ways has an artificially lowered merit. A feature such as `word` does not have an equivalent, but the value of a feature like `sentenceneighbour-` might well be derived from the same event as `eventneighbour-`. In this case the total distance increases, which effectively lowers the relative contribution of both features, and hence their feature merits. If the feature is randomly permuted, each instance which still contributes to the feature merit will have its contribution artificially lowered in the same way, so the merits of the permuted feature in the denominator will be lowered by the same ratio. Normalizing the merit then negates the detrimental effect. However, the problem seemed to subsist even with normalized features, albeit to a lesser degree.

Another problem is related to the large range of values, and their unequal distribution. The component distance for features is dependent on the total range of the features, which, in the TimeBank corpus, spans from 15 to 202 values. The feature with the most extreme range, a distance of 202 events, is `eventIDENTITY-`. This feature then has a t_k value of 101. However, despite this high range, 75% of the instances with a non-zero value for this feature have a distance of 10 or less, with frequency increasing as the distance lowers. If two instances of different classes are otherwise only different with respect to this feature, and the difference between their values for the feature is 1, this means both the event distance and total distance between those instances are 1/101. This comparison then contributes d_{ij}/D_{ij}^2 to the contextual merit of this feature, which is equivalent to the t_k in this case, that is, 101. A symbolic feature in the same situation, however, with a component distance of 1, would only contribute 1 to the contextual merit, while not necessarily being less useful for classification. This type of situation is not uncommon in the feature set, and it results in `eventIDENTITY-` consistently being one of the features with the highest feature merit, despite not producing very good results.

This problem could possibly be resolved by discretizing the numerical values, as outlined by Hong (1997). However, the problem of interdependence between features would remain. Even with normalized feature values, the results of contextual merit were dissatisfying in the context of tense and aspect classification. Therefore, the only results reported are obtained through controlled ablation.

5.4 Evaluation

The classifier is both trained and evaluated on the dataset. Evaluation is performed by comparing the result of the classifier for each instance to the class it actually has. In the classifier for tense, the tense of every instance is predicted. The accuracy is one of the main metrics, and simply denotes the ratio of classifications which were correct within the overall set. If c is the number of correct

classifications and i is the number of incorrected classifications, the accuracy can be computed like this:

$$accuracy = c/(c+i) * 100\%$$

This number isn't very informative about the performance of the model with respect to all the different classes. *Precision* and *recall* are used to describe the quality of the performance of the system with respect to a specific class. Every class has a number of *true positives* (tp), instances which were classified to that class correctly, *false positives* (fp), instances which were classified to the class but actually belong to a different one, *true negatives* (tn), instances which belong in the class but were not classified to it, and *false negatives* (fp), instances which were not classified in the class and did not belong to it. Precision (P) and recall (R) can then be computed like this:

$$P = tp/(tp+fp)$$
$$R = tp/(tp+fn)$$

Precision denotes the ratio of correctly classified instances compared to the total number of instances that were classified in that class. Recall denotes the ratio of correctly classified instances compared to the total number of instances which were supposed to be classified in it. Hence, for each class, precision represents how many of the classified instances were correct, and recall represents how many of the correct instances were classified.

Depending on the application, either precision or recall might be more important. This research, however, aims to optimize the overall result. This can be represented by the F-measure, a harmonic mean of precision and recall. The F-measure is computed like this:

$$F\text{-}measure = 2(PR)/(P+R)$$

WEKA automatically evaluates a classifier it generates, and has the option of using any of the basic evaluation methods. I use 10-fold cross validation, in which the data is partitioned in ten complementary sets. Each of the sets is used as a validation set once, by training the classifier on the nine remaining sets, and evaluating the resulting model with respect to the validation set. The accuracy, rate of true and false positives, precision, recall, and F-measure are all automatically measured by WEKA for each class. An overall result for these scores is produced by taking the average over all classes. For all systems tested, I will report the accuracy and the average precision, recall, and F-measure, mostly aiming to optimize the accuracy and the F-measure.

5.5 Classifiers

Using the methods outlined in this section, I will develop three classifiers: one for tense in isolation, one for aspect in isolation, and one for tense and aspect combined. These three classifiers represent two different strategies to resolve the problem: either tense and aspect are resolved separately and then combined, or one algorithm determines the tense and the aspect at the same time. The strategies have different theoretical allure.

The main benefit of isolated classification is that each algorithm has to deal with fewer classes, and hence all of the classes will occur more often than if they were generated at the same time. This is especially beneficial for rare tense-aspect combinations: in a combined system, there would possibly not be enough data to form rulesets to classify to the combination. The chance of success is greater when, although the tense is relatively rare, and the aspect is relatively rare, each of them occurs often enough to be classified independently. Another potential benefit is that the output of one system could function as a feature in the other. This latter possibility is not tested in the experiments, but might provide interesting future research.

The main benefit of combined classification is that tense and aspect interact: one feature which points toward a certain aspect given a certain tense might not do so when it is combined with a different tense. As shown in table 4.12, tense and aspect are very unevenly distributed with respect to each other, and many combinations never occur at all in the corpus. Using combined classification would ensure that the resources are used to classify to combinations which actually occur in the texts. As such, combined classification would be particularly beneficial for the tense-aspect combinations which occur at some significant frequency. However, it would probably go up at the expense of the less frequently observed combinations.

The next chapter will describe the performance of both the isolated classifiers and the combined classifier, for different algorithms and with different feature sets. It also contains a discussion of the different types of classifiers and the utility of the individual features for classification.

Chapter 6

Results and Discussion

This chapter contains the results of the experiments conducted to determine which feature sets are best and what the optimal performance of the algorithm is like. After an initial review of feature selection, it is subdivided into three parts: one on classification of tense, one on classification of aspect, and one on classification of the combination of tense and aspect. The first two represent different tasks in isolated generation, while the third is an attempt of combined generation. Classification is performed with the decision tree classifiers J48, and J48GRAFT, and the rule learner PART. The trees and rulesets are pruned with a confidence factor of 0.25, with the exception of the decision tree classifiers for isolated aspect, because pruning resulted in the neglect of progressive and perfective tense. J48 also performs grafting on the tree.

Since there are no previous results by which to measure the quality of this system, the base result to which the scores can be compared will be that of the ZEROR classifier, WEKA's baseline classifier, which classifies every instance into the most frequent class. This is equivalent to using any of the other algorithms without any features with which to perform classification. For evaluation of the feature sets, the results of the algorithm when given the full set of features will be provided for comparison.

6.1 Feature Selection

The features were selected through the controlled ablation method described in section 5.3.3. Analysis started with the full feature sets, and features were gradually removed and consequently left out if their removal improved the results of the system. Ablation was performed on two starting sets: one with the full set of features from table 5.3, and one which was limited to values which were assigned with a value at least in 1% of the instances, and of which not more than a quarter of its feature values was unique.

While these features could be valuable for the task at hand, and improve the accuracy of the algorithm, they represent information which is not readily available. This is a problem when put in the context of the hypothetical NLG system this algorithm would be implemented in. For these features, it is likely that many values will be observed in the test set which were not observed in the training set. Decision trees only work well when all possible values of their

features are known in advance (the possible nominal values are in this case specified in the .arff file to be used as features in WEKA). That requirement is unrealistic in the context of natural language. Hence, a second, 'practical' feature set, which excludes the very sparsely occurring features and those which have many unique values, is used as an alternative, to avoid unseen feature values. With a larger corpus available, this might not be an issue, in which case some of these features could be moved back into the full set. For every algorithm and strategy, ablation was performed twice: starting with the full feature set, and with the practical one.

Some of the features, like `word`, actually weaken the performance of the system when added, because their inclusion incurs overfitting. If `word` is included, the tree resulting from `j48` is mostly made up of nodes classifying the instance directly based on the lemma, ignoring the other features and decreasing the overall performance. On the other hand, the `signal` feature is used quite consistently in the optimal systems and the information it contains seems to be very valuable. So as not to lose the potential worth of these features, all of the impractical features except `word` are represented by a feature holding more general values based on the same feature. The features which were excluded from the practical sets of features are shown in table 6.1.

Table 6.1: *The features which are excluded from the practical set.*

FEATURE	REPLACEMENT
`word`	*None*
`modalityval`	`modalitybool`
`signal`	`signalrel` `signalbool`
`is_aspectual`	`is_aspectualbool`

`eventRELATION`	`eventneighbour`
	`closestrelation`
	`sentenceneighbour`
	`sentenceclosest`

All of the features were considered thoroughly for each of the tasks, including the impractical ones, so as to see whether they might be of benefit in future systems which are trained on larger corpora. Since the `word` and `eventRELATION` features were not beneficial in any task, they are excluded from the overviews of feature use. For every task, the selection with the optimal results for J48 (O-J48), J48GRAFT (O-J48GRAFT), and PART (O-PART) is shown, along with the practical set P. The practical set generally obtains near-optimal results for all three algorithms.

6.2 Isolated Tense Generation

The first task performed was generation of tense in isolation. The four feature sets for generating tense are shown in table 6.2, and the results for these feature sets are shown in table 6.3, with the system that produced the best results for the practical set, the J48GRAFT classifier, marked in bold. The detailed results for each class in this optimal system are shown in table 6.4. Some excerpts from

the resulting J48GRAFT tree is shown in figure 6.1.

Table 6.2: *The feature selection of the practical system and the optimal feature selection for each algorithm to classify into tense.*

FEATURE	P_t	O-J48$_t$	O-J48GRAFT$_t$	O-PART$_t$
`class`	☑	☑	☑	☑
`polarity`	☑	☑	☑	☑
`speechtime`	☑	☑	☑	☑
`modality`	☑	☑	☑	☑
`modalityval`		☐	☐	☐
`timebool`	☐	☐	☐	☐
`timetype`	☑	☑	☑	☑
`timeinsentence`	☐	☐	☐	☐
`signal`		☑	☑	☑
`signalrel`	☑	☐	☐	☐
`signalbool`	☐	☐	☐	☐
`has_subordination`	☐	☐	☐	☐
`is_subordination`	☑	☑	☑	☑
`has_aspectual`	☑	☑	☑	☑
`has_aspectualbool`	☐	☐	☐	☐
`is_aspectual`		☐	☐	☑
`is_aspectualbool`	☑	☑	☑	☑
`eventneighbour-`	☐	☐	☐	☐
`eventneighbour+`	☐	☐	☐	☐
`closestrelation-`	☑	☑	☑	☑
`closestrelation+`	☑	☐	☐	☐
`sentenceneighbour-`	☐	☐	☐	☐
`sentenceneighbour+`	☑	☑	☑	☑
`sentenceclosest-`	☐	☑	☐	☐
`sentenceclosest+`	☑	☐	☐	☐
`sentencerel0`	☑	☑	☑	☑
Total	14	13	12	13

6.2.1 Discussion

From table 6.2 a very clear trend can be observed as to which features are the most consistently useful when classifying tense. No optimal feature set could be obtained without the features `class`, `polarity`, `speechtime`, `modality`, `timetype`, `is_subordination`, `has_aspectual`, `is_aspectualbool`, `closestrelation-`, and `sentencerel0`. Most of these are also located near the top of the tree, which indicates they are the most important.

The boolean `modality` feature is preferred over the nominal feature `modalityval`. This makes sense, since all verbs which occur in a modal phrase are marked with the `none` tense in TIMEML. The nominal feature `has_aspectual`, on the other hand, performs better than its boolean counterpart `has_aspectualbool`, which shows that content information is useful in this context to classify tense.

Of the features which are excluded from the practical set, it seems that `signal` information is very useful. This is of note in case a resource is available

Table 6.3: *The results of different algorithms and feature sets to classify into tense.*

FEATURE SET	A	P	R	F	SIZE
J48					
Optimal	64.18%	0.632	0.642	0.618	336
Practical	64.18%	0.626	0.642	0.617	272
Full set	59.40%	0.572	0.594	0.571	3196
J48GRAFT					
Optimal	64.46%	0.634	0.645	0.62	980
Practical	**64.38%**	**0.629**	**0.644**	**0.619**	**788**
Full set	61.31%	0.592	0.613	0.591	63610
PART					
Optimal	64.46%	0.626	0.645	0.621	156
Practical	63.2%	0.607	0.632	0.61	162
Full set	58.61%	0.561	0.586	0.559	614
ZEROR					
None	40.66%	0.165	0.407	0.235	

Table 6.4: *The results of the* J48GRAFT *tense classifier using the practical set.*

CLASS	P	R	F
`past`	0.681	0.818	0.743
`future`	0.579	0.205	0.303
`infinitive`	0.555	0.788	0.651
`present`	0.632	0.479	0.545
`none`	0.971	0.979	0.975
`prespart`	0.415	0.268	0.325
`pastpart`	0.313	0.033	0.06
Weighted average	0.629	0.644	0.619

Figure 6.1: *Some excerpts from the practical* J48GRAFT *classifier tree for isolated tense generation.*

```
modality = none
| sentencerel0 = none
| | is_subordination = none
| | | speechtime = none
| | | | signalrel = none
| | | | | class = occurrence
| | | | | | is_aspectualbool = none
| | | | | | | timetype = none
| | | | | | | | closestrelation- = none
| | | | | | | | | polarity = pos:  infinitive (476.0/233.0)
| | | | | | | | | polarity = neg:  present (14.0/6.0)
| | | | | | | | closestrelation- = after
| | | | | | | | | polarity = pos
| | | | | | | | | | sentencerel+ = simultaneous:  past (0.0|5.0/1.0)
| | | | | | | | | | sentencerel+ != simultaneous
| | | | | | | | | | | sentenceneighbour+ = simultaneous:  past (0.0|5.0/1.0)
| | | | | | | | | | | sentenceneighbour+ != simultaneous:  infinitive (78.0/53.0)
⋮
| sentencerel0 = identity
| | speechtime = none
| | | class = occurrence
| | | | is_subordination = modal:  infinitive (0.0|217.0/41.0)
| | | | is_subordination != modal:  past (525.0/153.0)
| | | class = reporting:  past (473.0/56.0)
| | | class = i_state
| | | | sentenceneighbour+ = identity:  past (0.0|37.0/4.0)
| | | | sentenceneighbour+ != identity
| | | | | sentencerel+ = identity:  past (0.0|61.0/9.0)
| | | | | sentencerel+ != identity:  present (64.0/17.0)
⋮
| | speechtime = before:  past (406.0/69.0)
| | speechtime = after:  future (21.0/3.0)
| | speechtime = is_included:  present (120.0/15.0)
| | speechtime = includes:  present (78.0/3.0)
| | speechtime = simultaneous
| | | sentenceneighbour+ = identity:  past (0.0|61.0/7.0)
| | | sentenceneighbour+ != identity
| | | | sentencerel+ = identity:  past (0.0|102.0/13.0)
| | | | sentencerel+ != identity
| | | | | closestrelation- = is_included:  past (0.0|101.0/14.0)
| | | | | closestrelation- != is_included
```

in which annotated signals occur more frequently. If `signal` information is not available, the `signalrel` feature (which only has 2 unique values) can take over some of its functions, resulting in a slightly smaller accuracy. Some other features, namely `closestrelation+` and `sentenceclosest+` also appear to somewhat make up for the information which could otherwise be obtained through the `signal` feature. The other restricted features do not actually increase the efficiency of the algorithm, indicating that boolean values are generally more useful for very scarcely assigned features. The exception is `has_aspectual`, which increases the performance of the PART classifier, but only when used in conjunction with its boolean counterpart.

Table 6.3 shows the accuracy (A), precision (P), recall (R), F-measure (F), and the size, that is, the number of nodes in the decision tree or the number of rules in the case of PART, of the classifiers which use the different feature sets. There is a major difference between the results obtained by the classification algorithms and by the ZEROR results, so it can be concluded the use of these features is very beneficial for the task. Feature selection itself is also important, with all algorithms performing significantly better when subjected to the selected feature sets than when given the full set.

There is little difference between the different algorithms, although J48GRAFT produces slightly better results than J48, at the expense of an increase in tree size. PART produces results which are as good as those of J48 when using the unrestricted feature set, but its performance drops with the restricted set. Overall, J48GRAFT seems to be the most efficient classifier for tense, though if time is an issue J48 can be used instead.

The performance of the practical J48GRAFT classifier with respect to all tenses are shown in table 6.4. The general trend is that more frequent values are classified with both higher precision and recall, which is to be expected, due to more availability of data, and because optimization of the weighted average F-measure favours more frequently occurring classes. One exception to this is `none`, which performs unusually well compared to the others. This is because the only verbs which are classified as `none` in TIMEML are the ones in a modal phrase, making that a reliable feature by which to classify this tense.

`pastpart` is a particularly difficult feature to classify, with a very low recall. This is possibly due to the lack of some relevant features, since the difference between past participles and past simples is dependent not just on temporal features, but also largely on grammatical features. Consider, for example, "He has written the book" and "He wrote the book". The first example is a past participle, and the second is a simple past, but both could occur in the exact same temporal context. As such, adding additional syntactic or semantic information which is not found in the TIMEBANK corpus might be of benefit here. A possible solution for this problem would be to classify participles independently of tense.

Overall, automatic tense classification seems to be a difficult task, but not an impossible one. It is very clear which features are important for this task. Selection of the proper features results in a major improvement in the classifier's accuracy.

6.3 Isolated Aspect Generation

The second task which was performed was classifying aspect in isolation. The four feature sets which are used for this task are shown in table 6.5. The results for classification are shown in table 6.6, with the practical feature set that produces the best results marked in bold. The specifics of the performance of this algorithm are shown in table 6.7. Some excerpts from the resulting tree are shown in figure 6.2.

Table 6.5: *The feature selection of the practical system and the optimal feature selection for each algorithm to classify into aspect.*

FEATURE	P $_a$	O-J48 $_a$	O-J48GRAFT $_a$	O-PART $_a$
class	☑	☑	☑	☑
polarity	☐	☐	☐	☑
speechtime	☑	☑	☑	☑
modality	☑	☐	☑	☐
modalityval		☑	☑	☐
timebool	☐	☑	☑	☐
timetype	☑	☑	☑	☑
timeinsentence	☑	☐	☐	☐
signal		☑	☑	☑
signalrel	☐	☐	☐	☐
signalbool	☐	☐	☐	☐
has_subordination	☑	☑	☑	☐
is_subordination	☐	☐	☐	☐
has_aspectual	☐	☐	☐	☑
has_aspectualbool	☐	☐	☐	☐
is_aspectual		☑	☑	☐
is_aspectualbool	☐	☐	☐	☐
eventneighbour-	☑	☐	☐	☐
eventneighbour+	☐	☐	☐	☐
closestrelation-	☑	☑	☑	☑
closestrelation+	☑	☑	☑	☑
sentenceneighbour-	☐	☐	☐	☐
sentenceneighbour+	☐	☐	☐	☐
sentenceclosest-	☐	☑	☑	☑
sentenceclosest+	☐	☐	☐	☑
sentencerel0	☑	☑	☑	☑
Total	10	12	13	11

6.3.1 Discussion

Section 4.2.1 mentioned the main problem with generating aspect: about 4 out of 5 verbs in the TIMEBANK corpus are simple non-progressive, and hence marked with the `none` tag. This results in a very high accuracy even with the ZEROR classifier, by simply assigning all the features to the `none` class. This is undesirable, as, even if comparatively rare, progressive and perfective aspect do occur frequently in texts, and cannot simply be substituted for non-progressives.

Table 6.6: *The results of different algorithms and feature sets to classify into aspect.*

FEATURE SET	A	P	R	F	SIZE
J48 (UNPRUNED)					
Optimal	88.18%	0.842	0.882	0.85	738
Practical	87.63%	0.829	0.876	0.842	472
Full set	86.27%	0.81	0.863	0.831	1854
J48GRAFT (UNPRUNED)					
Optimal	88.36%	0.845	0.884	0.848	1574
Practical	**88.14%**	**0.838**	**0.881**	**0.844**	**960**
Full set	88.22%	0.824	0.882	0.834	37002
PART					
Optimal	88.65%	0.854	0.887	0.844	73
Practical	88.16%	0.826	0.882	0.837	40
Full set	87.53%	0.806	0.875	0.832	414
ZEROR					
None	87.51%	0.806	0.875	0.832	

Table 6.7: *The results of the* J48GRAFT *aspect classifier using the practical set.*

CLASS	P	R	F
`none`	0.892	0.986	0.937
`perfective`	0.359	0.072	0.122
`progressive`	0.618	0.111	0.188
`perfective_progressive`	0.0	0.0	0.0
Weighted average	0.838	0.881	0.844

Figure 6.2: *Some excerpts from the practical* J48GRAFT *classifier tree for isolated aspect generation.*

```
speechtime = none
| class = occurrence
| | sentencerel0 = none
| | | has_subordination = none:  none (1259.0/110.0)
| | | has_subordination = evidential:  none (6.0)
| | | has_subordination = modal
| | | | eventneighbour- = none
| | | | | timeinsentence = none
| | | | | | closestrelation- = none:  none (86.0/8.0)
| | | | | | closestrelation- = after:  none (5.0)
| | | | | | closestrelation- = identity:  none (3.0/1.0)
| | | | | | closestrelation- = simultaneous:  none (4.0)
| | | | | | closestrelation- = is_included:  none (0.0)
| | | | | | closestrelation- = before
| | | | | | | modality = yes:  none (0.0|14.0/1.0)
| | | | | | | modality != yes:  perfective (1.0)
⋮
| class = reporting
| | timetype = none
| | | eventneighbour- = none
| | | | has_subordination = none:  none (20.0)
| | | | has_subordination = evidential
| | | | | sentencerel0 = none:  none (82.0/2.0)
| | | | | sentencerel0 = identity:  none (378.0/6.0)
| | | | | sentencerel0 = simultaneous:  none (16.0)
| | | | | sentencerel0 = before:  none (5.0)
| | | | | sentencerel0 = after
⋮
speechtime = after
| has_subordination = none:  none (64.0)
| has_subordination = evidential:  none (4.0/1.0)
| has_subordination = modal
| | class = occurrence:  none (5.0)
| | class = reporting:  none (0.0)
| | class = i_state
| | | eventneighbour- = simultaneous:  none (0.0|14.0)
| | | eventneighbour- != simultaneous
| | | | closestrelation- = is_included:  none (0.0|10.0)
| | | | closestrelation- != is_included
| | | | | closestrelation- = simultaneous:  none (0.0|32.0/2.0)
```

Therefore, the feature sets were designed to maintain the high accuracy while at the same time accounting for the less frequent aspects, and optimization was focused on increasing the F-measure.

The optimal features shown in table 6.5 are interesting when compared to the features which were optimal for tense, shown in table 6.2. It appears that some of the main features, namely `class`, `speechtime`, `signal`, and `closestrelation-` are useful for classifying both tense and aspect. Other features which were universally important to classify tense, namely `polarity`, `is_subordination`, `has_aspectualbool`, and `sentenceneighbour+`, are important for tense but hardly used to classify aspect. Note that, whereas `is_subordination` and `has_aspectualbool` were used to classify tense, aspect classification places more importance on its counterparts, `has_subordination` and `is_aspectual`. The feature `closestrelation+` is also useful for aspect, but not so much for tense.

The features which were excluded from the practical set seem to have a beneficial effect on aspect generation. Unlike with tense, however, their use does not remain when they are swapped out for a more general similar feature, with the exclusion of `modality`. This indicates that the impractical features were mostly useful because of their unique values, rather than their content, which means it is a good thing not to include them, so as not to overfit the training data. Other than that, the practical feature set is quite similar to the optimal ones. It is of note that `timetype` and `eventneighbour-`, while not useful in any of the optimal systems, improves the system when they are left out, though that might be a result of the relative scarcity of some of its values. While the reported practical set had the highest observed F-measure, various other sets would produce very similar results, with variation mainly in the precision and recall of the `perfective` and `progressive` classes. It would be interesting to see how the algorithm would react to a larger number of assigned values in the impractical features, and if these features would still improve the system in that case.

Accounting for the rare classes was difficult due to a lack of available training features, but for all systems, both the optimal and the practical set result in a significantly higher F-measure than that of the zero classifier. For the decision tree classifiers, any subtree which led to an unusual aspect was automatically pruned to improve the accuracy of the system, resulting in an empty tree equivalent to the zero classifier, so no pruning was performed for these. However, the `none` aspect is still very predominant in the trees, as evident in figure 6.2. Only very specific circumstance were generally more likely to result in the non-standard aspects. J48GRAFT obtained the best results with the practical set, although the lack of pruning resulted in a large tree size. If the speed and the size are an issue, PART is the best choice, since it functions without disabling pruning.

The performance of J48GRAFT with respect to each of the aspects is shown in table 6.7. While the F-measure of the simple non-perfectives is very good, the perfectives and progressives are usually not classified accurately. While precision for these is decent, recall is very low, which is a sign of overfitting. The perfective progressive, of which there are only 20 instances in the corpus, is never classified at all. This was a consistent problem with all classifiers that were tested: though some of them classified a few instances into perfective progressive, none did so correctly. To be able to accurately predict the less frequent aspects, a larger corpus is most likely needed.

All in all, aspect is difficult to classify using the TimeBank corpus, even if it is just because of the scarcity of the nonstandard options. The algorithm will perform well in most circumstances, but err in the unusual cases. As of yet, this seems to be mostly a problem of overfitting, though, which means that it might very well be a problem which can be fixed using a larger corpus. Corpus-based generation of aspect seems to be very possible.

6.4 Combined Tense and Aspect Generation

The third algorithm takes tense and aspect in unison, and attempts to classify instances to a combination of their tense and aspect. This increases the number of classes, and hence decreases the average class frequency. There were 20 different combinations of tense and aspect in the corpus. The four feature sets which were used for classification are shown in table 6.8, and the results are shown in table 6.9, with the algorithm which produced the best results using the practical set, j48, marked in bold. Some excerpts from the resulting tree are shown in figure 6.3.

Table 6.8: *The feature selection of the practical system and the optimal feature selection for each algorithm to classify into tense and aspect.*

FEATURE	P $_{ta}$	O-J48 $_{ta}$	O-J48GRAFT $_{ta}$	O-PART $_{ta}$
class	☑	☑	☑	☑
polarity	☑	☑	☑	☑
speechtime	☑	☑	☑	☑
modality	☑	☑	☑	☑
modalityval		☐	☐	☐
timebool	☐	☐	☐	☑
timetype	☑	☑	☑	☑
timeinsentence	☐	☐	☐	☐
signal		☐	☑	☑
signalrel	☑	☑	☐	☑
signalbool	☐	☐	☐	☑
has_subordination	☐	☐	☐	☐
is_subordination	☑	☑	☑	☑
has_aspectual	☐	☐	☐	☑
has_aspectualbool	☑	☑	☑	☐
is_aspectual		☐	☑	☑
is_aspectualbool	☑	☑	☑	☐
eventneighbour-	☐	☐	☐	☑
eventneighbour+	☑	☑	☑	☑
closestrelation-	☑	☑	☑	☑
closestrelation+	☐	☐	☐	☑
sentenceneighbour-	☐	☐	☐	☐
sentenceneighbour+	☐	☐	☐	☐
sentenceclosest-	☑	☑	☑	☐
sentenceclosest+	☐	☐	☑	☐
sentencerel0	☑	☑	☑	☑
Total	13	13	15	19

Table 6.9: *The results of different algorithms and feature sets to classify into a combination of tense and aspect.*

FEATURE SET	A	P	R	F	SIZE
J48					
Optimal	60.38%	0.55	0.604	0.558	227
Practical	**60.38%**	**0.55**	**0.604**	**0.558**	**227**
Full set	55.62%	0.488	0.556	0.509	4417
J48GRAFT					
Optimal	60.52%	0.553	0.605	0.558	1402
Practical	60.38%	0.55	0.604	0.557	675
Full set	57.37%	0.502	0.574	0.523	54977
PART					
Optimal	60.34%	0.555	0.603	0.559	227
Practical	59.77%	0.537	0.598	0.552	164
Full set	54.16%	0.475	0.542	0.495	606
ZEROR					
None	38.63%	0.149	0.386	0.215	

Table 6.10: *The results of the* J48 *combined classifier using the practical set.*

TENSE	ASPECT	P	R	F
`past`	`none`	0.652	0.826	0.729
`future`	`none`	0.56	0.217	0.313
`infinitive`	`none`	0.553	0.786	0.649
`present`	`perfective`	0.143	0.019	0.033
`present`	`progressive`	0.561	0.142	0.227
`none`	`none`	0.868	0.977	0.919
`present`	`none`	0.574	0.541	0.557
`prespart`	`none`	0.386	0.316	0.348
`pastpart`	`none`	0.281	0.107	0.155
`past`	`perfective`	0.0	0.0	0.0
`none`	`perfective`	0.0	0.0	0.0
`present`	`perfective_progressive`	0.0	0.0	0.0
`past`	`progressive`	0.0	0.0	0.0
`future`	`perfective`	0.0	0.0	0.0
`none`	`perfective_progressive`	0.0	0.0	0.0
`none`	`progressive`	0.0	0.0	0.0
`pastpart`	`perfective`	0.0	0.0	0.0
`prespart`	`perfective`	0.0	0.0	0.0
`future`	`progressive`	0.0	0.0	0.0
`past`	`perfective_progressive`	0.0	0.0	0.0
Weighted average		0.55	0.604	0.558

6.4.1 Discussion

The optimal feature sets are remarkably similar to those of tense. In fact, using this practical set provides good results for isolated tense as well, with a

Figure 6.3: *Some excerpts from the practical J48 classifier tree for combined tense and aspect generation.*

```
modality = none
| sentencerel0 = none
| | speechtime = none
| | | is_subordination = none
| | | | signalrel = none
| | | | | class = occurrence
| | | | | | closestrelation- = none
| | | | | | | timetype = none
| | | | | | | | eventneighbour+ = after
| | | | | | | | | polarity = pos:  infinitive_none (34.0/18.0)
| | | | | | | | | polarity = neg:  future_none (2.0)
| | | | | | | | eventneighbour+ = none
| | | | | | | | | polarity = pos:  infinitive_none (362.0/155.0)
| | | | | | | | | polarity = neg:  present_none (10.0/6.0)
| | | | | | | | eventneighbour+ = is_included:  past_none (13.0/7.0)
| | | | | | | | eventneighbour+ = before:  past_none (27.0/17.0)
| | | | | | | | eventneighbour+ = simultaneous:  prespart_none (18.0/12.0)
| | | | | | | | eventneighbour+ = identity:  past_none (22.0/13.0)
| | | | | | | | eventneighbour+ = includes:  past_none (6.0/4.0)
| | | | | | | timetype = date
| | | | | | | | is_aspectualbool = none
| | | | | | | | | has_aspectualbool = none:  past_none (60.0/33.0)
| | | | | | | | | has_aspectualbool = yes:  infinitive_none (3.0/1.0)
| | | | | | | | is_aspectualbool = yes:  prespart_none (10.0/2.0)
⋮
| sentencerel0 = after
| | signalrel = none
| | | class = occurrence
| | | | closestrelation- = none:  past_none (3.0/1.0)
| | | | closestrelation- = after:  infinitive_none (37.0/18.0)
| | | | closestrelation- = identity:  infinitive_none (0.0)
| | | | closestrelation- = simultaneous:  infinitive_none (0.0)
| | | | closestrelation- = is_included:  infinitive_none (0.0)
| | | | closestrelation- = before:  infinitive_none (0.0)
| | | | closestrelation- = includes:  infinitive_none (0.0)
| | | class = reporting:  past_none (5.0/3.0)
| | | class = i_state:  future_none (4.0/3.0)
| | | class = i_action:  past_none (7.0/3.0)
| | | class = state:  future_none (4.0/2.0)
| | | class = aspectual:  past_none (4.0/2.0)
| | | class = perception:  infinitive_none (1.0)
| | signalrel = is_included:  past_none (3.0/1.0)
| | signalrel = after:  past_none (2.0)
| | signalrel = before:  infinitive_none (0.0)
| | signalrel = conditional:  infinitive_none (0.0)
| | signalrel = simultaneous:  future_progressive (1.0)
| | signalrel = initiates:  infinitive_none (0.0)
| | signalrel = includes:  infinitive_none (0.0)
| | signalrel = modal:  infinitive_none (0.0)
⋮
| sentencerel0 = is_included
| | is_subordination = none:  prespart_none (29.0/18.0)
| | is_subordination = modal:  infinitive_none (1.0)
| | is_subordination = evidential:  infinitive_none (3.0/2.0)
| | is_subordination = conditional:  prespart_none (0.0)
| | is_subordination = factive:  prespart_none (0.0)
| | is_subordination = counter_factive:  prespart_none (0.0)
modality = yes:  none_none (242.0/32.0)
```

J48GRAFT classifier resulting in a 64.24% accuracy and an F-measure of 0.618. This makes sense, given that due to the high frequency of the standard aspect, and the difficulty with predicting the others, most of the combined classification will be concerned with tense more than with aspect.

Similarly to tense, there is a very clear trend with respect to the most consistently useful features, and they are mostly the same. The exceptions to this are `sentenceclosest-`, which was not used for most tense classifiers, but was in all the optimal sets for aspect (though not in the practical set), and `sentenceneighbour+`, which was important for aspect but not for the combined classifier. The order in which the different features occur in the tree is also very similar to that of tense. It is notable that, while both the practical set for tense and all the sets for aspects benefit from the `closestrelation+` feature, this was not the case for the combined classifier. It is also of note that, while the tense classifier used the `has_aspectual` feature, the combined classifier functions better with its boolean alternative. The optimal `j48` set also performed better without the `signal` feature, and, like tense, without `modalityval`, leading it to be identical to the practical set.

The results for the combined classifier are shown in table 6.9. It is evident that the features are very useful, with the full set producing much better results than the zero classifier, and feature selection resulting in another significant improvement. Again, the different algorithms produce very similar results, though unlike the previous two tasks, J48 performs the best on the practical set by a small margin. Overall, combined classification seems to be a better strategy than isolated classification: the best accuracy is 60.38%, which compares favourably to the product of the best accuracies in isolated classification, 56.74%.

The results for each of the classes are shown in table 6.10. A large subset of the classes is never used, and of those which are, a significant subset has very low recall. These results do not improve when pruning is disabled. Notably, the vast majority of the features with very low recall use the nonstandard aspects. It is evident that the problems which occurred with isolated aspect classification, and to a lesser degree that of classification of the `pastpart` tense, persist in the case of combined classification.

On the other hand, it seems that combining tense and aspect alleviates the problem to some degree. Although the majority of the classes shown in the table are never classified, a look at table 4.12 shows that these classes in fact only make up 158, or 3.1%, of the instances in the corpus together. Of the 9 remaining classes which are classified, 4 still have very low recall, but even these scores are generally higher than the recall their most difficult components have in isolation. Isolated aspect classification for `progressive`, for instance, results in a recall score of 0.111, but the more specific `present_progressive` has a recall score of 0.142. In other words, through combined classification, the algorithm at least classifies a subset of otherwise unwieldy classes correctly.

In a similar vein, the results for the participles improved when compared to the performance of the isolated tense classifier. The `prespart` class is classified with an F-measure of 0.325 in isolation, but `prespart_none` is classified with an F-measure of 0.348. Since all but one of the present participles in the corpus occur without aspect, this makes sense. Similarly, the `pastpart` class, the verbs of which occur without aspect in all but two of the cases, is classified with an F-measure of only 0.06 in isolation. `pastpart_none` undergoes a radical

improvement to 0.155.

These results are to be expected to a degree, as mentioned in section 4.2.1. Theoretically, the very low frequencies at which many of the tense-aspect combinations occur limits the versatility of the algorithm, since the classes which represent them will most likely never be selected in classification. In practice, however, the low-frequency tenses and aspects already pose problems by themselves. Performing combined classification improves the results for the frequent combinations, leaving only infrequent combinations, rather than both infrequent tenses and infrequent aspects, as major problems. This leads me to believe that combined classification is a more effective strategy for tense and aspect generation than isolated classification.

6.5 General Discussion

The results obtained in these experiments are not quite good enough to be implemented in a full-fledged NLG system for narratives, but they provide an optimistic view of the possibility. Isolated tense classification provides decent results in both tense and aspect for all classes but one, and this class is the one which occurs the least frequently in the corpus. Isolated aspect classification performs very well with respect to the most frequent class, although the other three classes, which also occur quite rarely in the corpus, are not handled well, resulting in a very low recall, or, in one case, never being classified at all. Combined classification seems to solve this problem to some degree, with significantly better results for some subsets of the problematic classes as compared to their isolated counterparts. Combined classification also leads to the highest overall accuracy, indicating that it is a better strategy than generating tense and aspect separately.

There is no precedent research in corpus-based generation of tense and aspect. Since rule-based systems do not have a gold standard available by which the accuracy of their tense and aspect can be evaluated, this makes comparison of the current results to those of other systems difficult. Rule-based systems are normally evaluated by either metric-based corpus evaluation, in which the resulting texts are compared to existing texts written by humans, extrinsic evaluation, in which it is determined how effective texts were at accomplishing a certain goal, or non-task-based evaluation, in which resulting texts are read and rated by humans. These methods cannot be applied in the current context, since they require full-fledged texts, whereas the presented classifiers only generate tense and aspect for an event. That said, it might well be the case that the algorithm would perform better in a complete NLG system than the results suggest, since the same event can often be expressed by multiple tense-aspect combinations. This means that even of the instances which were not 'correctly' classified, some likely result in tense-aspect combinations which are clear and grammatically correct, especially since, due to the nature of the classifier, the mismatched instances are usually classified as combinations which occur in the same context, and are therefore more likely to be semantically close.

The closest equivalent to which the algorithm can be compared are the parallel NLU systems developed for the TempEval challenges. The challenges concern a number of tasks related to identifying temporal relations in text, two of which are determining the temporal relation between two main events in

consecutive sentences (task E) and determining the temporal relation between two events where one event syntactically dominates the other (task F). These tasks represent classification problems which are resolved with machine learning methods and a corpus derived from TimeBank, and likely have to deal with some similar issues. The best results F-measures for these tasks at the TempEval-2 challenge were 0.58 for task E and 0.60 for task F, with average results over all the submitted systems of 0.53 and 0.55 respectively (Verhagen et al., 2010). The current results for combined classification of tense and aspect, with an F-measure of 0.577, are on par with the average results for these tasks.

As far as isolated and combined tense and aspect generation goes, it seems evident that combined generation is more useful, at least given the current domain and data. Aside from the higher overall accuracy, combined generation accounts better for scarcely occurring tenses and aspects. The isolated classifiers have poor recall for infrequent classes, which means that, if they were to be implemented, very few of the instances which are supposed to have that infrequent class will actually be assigned it. Using combined classification, at least the more frequent subsets will perform better. This strategy does make the assignment of the other combinations, which are very infrequent, even less likely, but in the current context those would not be assigned correctly regardless.

The results of isolated aspect generation were not very solid. While the addition of features and their subsequent selection could be used to increase the F-measure, the benefits were minor, and similar results could be obtained using a variety of feature sets, with very different optimal sets for J48 and PART. This places it in stark constrast with the tense and combined classifiers, which exhibited a clear trend toward the utility of certain features. Moreover, all three of the impractical features were generally useful in aspect classification, but for two of them that utility did not carry to their simplified counterparts, which likely means the only merit of these features was in their unique values. As such, the observed increase could actually be explained as the result of random variation rather than actual benefit of the feature sets.

Although some of the features might still have some amount of merit for aspect classification, it could well be the case that the TimeBank corpus does not actually contain very useful information with respect to aspect. If this is the case, other information, like syntactic or semantic information about the sentence in which the verb occurs, might be of more benefit than the currently used temporal information. It could also be the case, however, that aspect is difficult to classify without the context of tense, because aspect by itself does not define a distinct position on the time line. A progressive verb might describe a past, an ongoing, or a future event. The same goes for the other aspects; even a perfective verb might unambiguously refer to an event in the future, as in the case of "The company will have existed for fifty years next week". Whether the low recall of aspect is due to an irrelevance of the used information, dependence on tense, or some other reason is an interesting question, and one which warrants more research if aspect is to be successfully generated.

The tendency toward unsatisfactory results in the less frequently occurring classes indicates that part of the problem lies with the size of the corpus. As mentioned before, the TimeBank corpus is very small compared to other corpora which are successfully used for statistical generation. It is virtually impossible to build a classifier which functions fully for classes like the `present_progressive` aspect, which only occurs a total of 20 times in the cor-

pus. This is likely one of the causes of the overfitting which seems to occur for at least one of the classes no matter which strategy is used, evidenced by low recall values. Another is the genre-specificity that results from training only on newspaper texts, which tend to have a very rigid format, and likely favour certain values for tense and aspect. If other types of narratives were to be used for training, certain classes would most likely occur more frequently, although mixing genres might also make the algorithm less reliable.

6.5.1 Feature Utility

The set of optimal features for the different classifiers provides some interesting insights. Some features are of particular note, since they were used in every optimal and practical set for all three classifiers. These are `class`, `speechtime`, `timetype`, `closestrelation-`, and `sentencerel0`. Other features which are generally useful are `modality`, `polarity`, `signal`, `has_aspectual`, and `sentenceclosest-`. This subsection examines some of these features more closely, because their utility likely hints toward the type of information which could be used to improve the results of the system. The most obvious features which point toward possible improvements to the system are `class`, `sentencerel0`, and `signal`.

The inclusion of `class` in every optimal set is important to note, because it is the only feature used in classification which has to do with the semantic content of the verb which is being classified. Much work, mainly that derived off Vendler (1957) and Dowty (1979), stresses the tendency of specific verbs, like 'read', or 'knock', to be used with particular aspects (see section 3.2.2), but no such information was available from the TimeBank corpus. Lemma information was originally included in the hopes that information about the aspectual class might be statistically inferred from its aspect frequencies, but most lemmas occurred too sparsely to contain that information, and their inclusion obscured the other features. To my knowledge, no public database of verbs and their aspectual classes is available, but it might very well be the case that additional semantic information from an external resource such as WordNet (Miller, 1995) could be of use in increasing the algorithm's performance. This is especially true since the inclusion of `class` increases the performance of tense as well as aspect classification, although the importance for tense might have more to do with tendencies in the domain than with linguistic qualities: for example, newspaper texts are very likely to contain `REPORTING` verbs, like 'said' or 'announced', in simple past tense.

Like the `class` feature, the consistent importance of `sentencerel0` is very interesting, since it is the only feature which is directly inferred from the syntactic structure of the sentence in which an event occurs. Syntactic information is not annotated in the TimeBank corpus, but many of the applications which automatically infer temporal relations from text make extensive use of it. The high value of this feature indicates that results might benefit from a more thorough syntactic representation of the sentence in which the event will occur. This is difficult in a standard generation system, because the syntactic structure is usually generated automatically in the sentence realizer. Obtaining more syntactic information would be possible by constructing a proto-syntactic structure more thoroughly in the microplanner, leaving less work for the realizer. It could also be argued that the pipeline architecture is actually not a very good strat-

egy to generate narrative texts, because it does not allow interaction between the syntactic and semantic modules, in which case a fully integrated approach might be used to gain better results.

The `signal` feature is a tragic hero, which was a part of all but one of the optimal sets, but consistently had to be excluded from the practical sets due to the uniqueness of many of its values. It was the only impractical value which actually increased performance, which indicates that this information is valuable. Its utility likely comes from idiomatic expressions, which makes it particularly useful to classify aspect, which can have different values regardless of event time. It could be the case that `signal`'s large number of unique values were accountable for its utility, but this seems unlikely when it is considered that the `word` feature has many more unique values, and its inclusion only weakened the performance of the classifier. A bigger corpus would be the primary way in which larger frequencies of occurrence of the signal values could be obtained, although this might also result in the inclusion of additional rare signal words, and hence not solve the problem of overabundance of unique values. It would be worthwhile to see if signal information is as useful in case of a higher number of instances which have non-zero values for the features, possibly after pruning the rarest values.

The utility of the other frequently useful features is, it seems, generally either easy to explain, or likely a result of style, be it generic or specific to the the newspaper format. An example of the former, an easily explainable feature, is `speechtime`, the instances of which frequently have past tense or perfective aspect when its value is `BEFORE`, present tense or progressive aspect when its value is `INCLUDES` or `IS_INCLUDED`, and future tense when its value is `AFTER`. A likely example of the latter, whose merit is likely a result of the style of newspapers, is `timetype`, with verbs which are temporally connected to a date or duration being more likely to occur with past tense and non-progressive imperfect aspect.

Whether linguistic or stylistic preference is the most likely explanation for continued inclusion of features is not necessarily the same for tense and aspect: `modality`, for example, only occurs without tense, so verbs which have it always have `none` tense. For aspect, on the other hand, `modality` is likely useful because generic (and perhaps domain) style dictates verbs in a modal phrase do not often occur with progressive or perfective aspect. While interesting to examine, these features do not seem to provide evidence for any particular information the algorithm would benefit from, although the many style-dependent features do indicate that the quality of performance might decrease if the training and evaluation texts contain texts of mixed genres.

6.5.2 Representation Strategies for Temporal Relations

Section 5 described the different possible strategies for feature representation of the TimeML temporal relations in some detail. After evaluating the results, it is a lot more clear which of the representations was better suited to be used in classification. Two of the three strategies, event recognition and root recognition, which looked in their immediate surroundings for the closest events, proved to be valuable very often. Relation recognition, on the other hand, which considered each temporal relation type as a separate pair of features, did not increase the accuracy of any system.

The utility of the event recognition and root recognition is very similar, but event recognition seems to be useful slightly more often. This is most likely at least partially a result of the feature being assigned a value more frequently. The `closestrelation-` and `sentenceclosest-`, both of which iterate backward through the events, are the two most useful features among them. They are used concurrently in many algorithms, which means that, while they represent the same information, their manner of represention adds utility to the feature. The `closestrelation+` and `sentencerelation+` features were also quite frequently used. This is good to know, since apparently they all represent different types of information.

The utilities of the features which only looked at their neighbours, interestingly, are very different to one another. The features which looked to their neighbour later in the text were useful, whereas the features which looked toward their backward neighbour were never in the optimal sets. Apparently, the temporal relation to the event directly after an event is a useful variable in determining its most appropriate tense. Why this is the case for relations to the neighbours later in the text, and not the neighbours before it, is not immediately evident from the feature values themselves.

The apparent uselessness of the `eventRELATION` features can most likely be explained by their universally scarce assignment in the corpus. Representing the presence of the events as numerical values seemed not to be useful at all, since usually only temporal relations usually of an event to other events in its immediate surroundings are annotated. Having a sense of how many events are in between one event and another is no substitute for grammatical insight.

All in all, temporal relations are represented better by nominal-valued features than by numerical features. For some contexts it is best to know simply what event is closest, and in others which root verb is closest. In fact, it is often the case that they are both important. Given their frequent co-occurrence as relevant features, it might be beneficial to the algorithm to include a larger variety of more informed measures, like the closest noun event, or the relation to an event verb's own arguments.

6.5.3 Future Work

While the obtained results are decent overall, and much better than the ones for the zero classifier, there is still a lot of room for improvement. All of the classifiers suffer from overfitting for at least one of the classes, and many of the features might be less useful than they could be due to very low rates of assignment. This seems to be a direct result of the small size of the TimeBank corpus, and the lack of resources that entails. While some statistical NLG systems such as the NitroGen statistical realizer (Langkilde & Knight, 1998) are widely used to good effect, NitroGen is based on the Penn Treebank corpus, which is an order of magnitude bigger than TimeBank. More annotated resources would increase the rate of assignment of both the rare feature values and the rare classes, which are the clearest cause of problems in the classifier.

This leaves the question how to obtain more resources. There is one more English TimeML-annotated corpus of newspaper articles, the AQUAINT corpus[1]. This corpus contains 73 documents, and could be merged with the TimeBank

[1] http://www.timeml.org/site/timebank/timebank.html

corpus for additional data, as has been previously done successfully by Mani et al. (2006). Besides this, while the most obvious answer would be annotating more data by hand, this is expensive. There are several other, more feasible ways to obtain more data.

Mani et al. (2006) observed a major improvement of their results in automatically annotating temporal relations based on the TIMEBANK corpus by adding in a closure component, SPUTLINK (Verhagen, 2004). While the task in this research is different, the inclusion of transitive closure, obtained by inferring the additional temporal relations in the event semantics through Allen's interval algebra, might be of benefit for the large number of features which is based on the temporal relations in TIMEBANK. No recent standalone version of SPUTLINK is available, but adding a closure component to the algorithm which extracts the features from the corpus would not be very difficult, and is a feasible strategy to gain more information from the available resources.

It should also be noted that the availability of TIMEML-annotated resources is directly related to the continued research in automatic annotation. While the current automatic annotation tools do not function well enough to use the results for training of a generator, continued progress is being made, and training on automatically annotated files might be an option in the future. Because most of the information which is being handled is similar, the possibilities of tense and aspect generation therefore seem likely to evolve along with those for automated temporal processing.

Aside from the limited size of the TIMEBANK corpus, it is also quite likely that the corpus simply does not provide all of the useful information for tense and aspect generation by itself. The fact that some of the most useful features for tense and aspect generation contained syntactic information on the different sentences, and semantic information about the verb at hand, is indicative of the value of such information for generation. This is an issue for aspect particularly in the light of the traditional models which it is subject to, following Vendler (1957) and Dowty (1979). Aspectual class was not accounted for at all in the feature set, while it is likely to be a useful feature for aspect. More syntactic and semantic features, which could be obtained through parsers and semantic resources such as WORDNET, would likely result in a better performance.

If combined generation is used to perform NLG of tense and aspect, an issue is that the very rarely occurring tense-aspect combinations will likely never be generated by a statistical system, since there probably always will be a more frequent combination which can be used under the same circumstances. A complete NLG system will have to account for this somehow. One option to make this possible is to have the decision be redirected to an separate rule-based system based on Reichenbach's theory of tense under certain circumstances. It is not unthinkable to use two different decision mechanisms in different contexts, and this could result in a system which combines the speed and adaptability of machine-learned methods with the rigidity of applied linguistic rules when needed. Whether this is the best solution to the problem of scarcely occurring tense-aspect combinations or not, some adaptation will have to be made to the system to ensure they can be classified if it is to perform realistic generation of newspaper-like texts. The merits of statistical generation of tense and aspect as compared to rule-based generation and vice versa, and under which types of circumstances these merits are beneficial, should be investigated further.

All in all, there are several possible sources of additional information through

which the performance of the algorithm could be improved. Exploring these options and determining which has a positive effect on the results by which progress can be made with respect to the current results. The most straightforward ways of taking this research forward currently seem to be merging the TIMEBANK corpus with the AQUAINT corpus to obtain more data, and to use Allen's interval algebra to obtain closure in the temporal relations, which would improve the frequency with which features based on temporal relations are assigned. There are many additional options through which the results could possibly be enhanced, among which the inclusion of more syntactic and semantic information. The generation framework presented here is the first of its kind, and provides a basis from which more advanced systems can be developed, but much work remains to be done to make full generation of narratives possible.

Chapter 7

Concluding Remarks

This research describes a framework which generates the tense and aspect of an event based on information about its context, semantic content, and temporal relations to other events and times. This information can be extracted from the TimeBank corpus for training, but similar information could be obtained within an NLG system, in which case the resulting algorithm would function as an additional module to the microplanner. The results reported are for the English TimeBank corpus, but several similarly annotated corpora are available in different languages, so the framework can be cross-linguistically applied.

I used J48, J48graft, and PART classifiers in the WEKA data mining environment to create decision tree algorithms which can be used to classify tense and aspect based on a number of features with values extracted from the TimeBank corpus. Tense and aspect were classified both in isolation and in combination, and for each strategy, feature selection was performed through ablation. The optimal results were obtained by combined tense and aspect generation through J48, with an accuracy of 60.38%, and an F-measure of 0.557. This is comparable to the results obtained for prediction of temporal relations between events in the TempEval challenge, through NLU.

At the beginning of this dissertation, I asked the following question: given information about the non-linguistic temporal context in which an event occurs, as well as about various semantic and grammatical parameters related to the way that event needs to be expressed linguistically, is it possible to accurately predict its tense and aspect properties? Strictly speaking, the answer to this question remains inconclusive, since the results which were obtained using the framework described in this thesis were of moderate quality. However, the framework opens up a myriad of directions to explore, and there is good reason to believe many of those can be used to improve its performance. Accurate corpus-based tense and aspect generation is not available as of yet, but this research represents an initial step toward it.

References

Afantenos, S., Karkaletsis, V., & Stamatopoulos, P. (2005). Summarization from medical documents: a survey. *Artificial Intelligence in Medicine*, *33*, 157-177.

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Artificial Intelligence and Language Processing*, *26* (11), 832–843.

Appelt, D. (1985). Planning english referring expressions. *Artificial Intelligence*, *26-1*, 1-33.

Bache, C. (1982). Aspect and aktionsart: Towards a semantic distinction. *Journal of Linguistics*, *18 (1)*, 57-72.

Bartalesi Lenzi, V., Moretti, G., & Sprugnoli, R. (2012). Cat: the celct annotation tool. In N. Calzolari et al. (Eds.), *Proceedings of the eight international conference on language resources and evaluation* (p. 333-338). Istanbul, Turkey: ELRA.

Barzilay, R. (2003). *Information fusion for multi-document summarization: Paraphrasing and generation.* Unpublished doctoral dissertation, Columbia University.

Barzilay, R., Elhadad, N., & McKeown, K. (2002). Inferring strategies for sentence ordering in multidocument news summarization. *Journal of Artificial Intelligence Research*, *17*, 35-55.

Barzilay, R., & Lapata, M. (2005). Collective content selection for concept-to-text generation. In *Proceedings of human language technology conference and conference on empirical methods in natural language processing* (p. 331-338).

Barzilay, R., & Lapata, M. (2008). Modeling local coherence: An entity-based approach. *Computational Linguistics*, *34 (1)*, 1-34.

Bateman, J. (1997). Enabling technology for multilingual natural language generation: the kpml development environment. *Natural Language Engineering*, *3 (1)*, 15-55.

Belz, A. (2008). Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, *1*, 1-26.

Belz, A., Kow, E., Viethen, J., & Gatt, A. (2010). Empirical methods in natural language generation. In E. Krahmer & M. Theune (Eds.), (p. 294-327). Springer.

Bethard, S., & Martin, J. H. (2007). Temporal relation classiĭnĄcation using syntactic and seman tic features. In *Proceedings of the 4th international workshop on semantic evaluations.*

Binsted, K., Cawsey, A., & Jones, R. (1995). Generating personalised patient

information using the medical record. In *Lecture notes in computer science* (p. 934.). Springer Verlag.

Binsted, K., Pail, H., & Ritchie, G. (1997). Children's evaluation of computer-generated punning riddles. *Pragmatics & Cognition, 5:2*, 305-354.

Bittar, A. (2011). *Building a timebank for french: A reference corpus annotated according to the iso-timeml standard.* Unpublished doctoral dissertation, UniversitÃĺ Paris Diderot.

Bluedorn, A. (2002). *The human organization of time: Temporal realities and experience.* Stanford University Press.

Boguraev, B., & Ando, R. K. (2005). Effective use of TimeBank for TimeML analysis. In *Proceedings of the 2005 international conference on annotating, extracting and reasoning about time and events* (p. 41-58).

Bouttaz, T., Pignotti, E., Mellish, C., & Edwards, P. (2011). A policy-based approach to context dependent natural language generation. In *Proceedings of the 13th european workshop on natural language generation* (p. 151-157).

Cahill, A., & Genabith, J. van. (2006). Robust PCFG-based generation using automatically acquired lfg approximations. In *Proceedings of the 21st international conference on computational linguistics and 44th annual meeting of the acl* (p. 1033-1040).

Cahill, L., Doran, C., Evans, R., Mellish, C., Paiva, D., Reape, M., et al. (1999). In search of a reference architecture for nlg systems. In *Proceedings of the 7th european workshop on natural language generation* (p. 77-85).

Callaway, C., & Lester, J. (2002). Narrative prose generation. *Artificial Intelligence, 139*, 213-252.

Carroll, J., & Oepen, S. (2005). High efficiency realization for a wide-coverage unification grammar. In *Proceedings of the second international joint conference on natural language processing.*

Caselli, T. (2010). *It-timeml: Timeml annotation scheme for italian - version 1.3.1* (Tech. Rep.). Pisa, Italy: ILC CNR.

Caselli, V., T. Bartalesi Lenzi, Sprugnoli, R., Pianta, E., & Prodanof, I. (2011). Annotating events, temporal expressions and relations in italian: the it-timeml experience for the ita-timebank. In *Proceedings of the fifth law workshop* (p. 143-151). Portland, Oregon.

Cheng, H., Poesio, M., Henschel, R., & Mellish, C. (2001). Corpus-based np modifier generation. In *Proceedings of the second meeting of the north american chapter of the association for computational linguistics on language technologies* (p. 1-8).

Coch, J. (1996). Evaluating and comparing three text-production techniques. In *Proceedings of the 16th conference on computational linguistics* (Vol. 1, p. 249-254).

Coch, J. (1998). Interactive generation and knowledge administration in multimeteo. In *Proceedings of the ninth international workshop on natural-language generation* (p. 300-303).

Cohen, W. W. (1995). Fast effective rule induction. In *Machine learning: proceedings of the twelfth international conference* (p. 115-123).

Comrie, B. (1976). *Aspect.* Cambridge University Press.

Costa, F., & Branco, A. (2012). Timebankpt: A timeml annotated corpus of portuguese. In N. Calzolari et al. (Eds.), *Proceedings of the eight interna-*

tional conference on language resources and evaluation. Istanbul, Turkey: ELRA.

Dale, R., Green, S., Milosavljevic, M., Paris, C., Verspoor, C., & Williams, S. (1998). The realities of generating natural language from databases. In *Proceedings of the 11th australian joint conference on artificial intelligence.*

Dale, R., & Reiter, E. (1995). Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science*, *19 (8)*, 233-263.

Dale, R., & Viethen, J. (2009). Referring expression generation through attribute-based heuristics. In *Proceedings of the 12th european workshop on natural language generation* (p. 58âĂŞ65).

Daumé III, H., Knight, K., Langkilde-Geary, I., Marcu, D., & Yamada, K. (2002). The importance of lexicalized syntax models for natural language generation tasks. In *Proceedings of the second international natural language generation conference.*

Davidson, D. (1967). The logical form of decision and action. In N. Rescher (Ed.), (p. 81-95). University of Pittsburgh Press.

Day, D., Ferro, L., Gaizauskas, R., Hanks, P., Lazo, M., Pustejovsky, J., et al. (2003). The TIMEBANK corpus. In *Proceedings of corpus linguistics* (p. 647-656).

Deemter, K. van, & Odijk, J. (1997). Context modeling and the generation of spoken discourse. *Speech Communication*, *21*, 101-121.

Di Fabbrizio, G., Stent, A., & Bangalore, S. (2008). Trainable speaker-based referring expression generation. In *Twelfth conference on computational natural language learning* (p. 151-158).

Dorr, B. J., & Gaasterland, T. (1995). Selecting tense, aspect and connecting words in language generation. In *Proceedings of the 14th international joint conference on artificial intelligence.*

Dowty, D. (1979). *Word meaning and montague grammar.* Dordrecht, the Netherlands: D. Reidel.

Dowty, D. (1986). The effects of aspectual class on the temporal structure of discourse: Semantics or pragmatics? *Linguistics and Philosophy*, *9 (1)*, 37-61.

Eldahad, M., McKeown, K., & Robin, J. (1997). Floating constraints in lexical choice. *Computational Linguistics*, *23*, 195-239.

Eldahad, M., & Robin, J. (1996). A reusable comprehensive syntactic realization component. In *Proceedings of the eighth international workshop on natural-language generation (inlg-1996).*

Elomaa, T. (1999). The biases of decision tree pruning strategies. In *Proceedings of the third international symposium on advances in intelligent data analysis* (p. 63-74).

Elson, D. (2012). *Modeling narrative discourse.* Unpublished doctoral dissertation, Columbia University, New York City.

Evans, R., Piwek, P., & Cahill, L. (2002). What is NLG? In *Proceedings of the sixth international conference on natural language generation.*

Evans, R., Weir, D., Carroll, J., Paiva, D., & Belz, A. (2007). Modelling control in generation. In *Proceedings of the eleventh european workshop on natural language generation.*

Ferres, L., Parush, A., Roberts, S., & Lindgaard, G. (2006). Helping people with

visual impairments gain access to graphical information through natural language: The igraph system. In *Proceedings of the 10th international conference on computers helping people with special needs.*

Ferro, L., Gerber, L., Mani, I., Sundheim, B., & Wilson, G. (2005). *TIDES 2005 standard for the annotation of temporal expressions* (Tech. Rep.). MITRE.

Ferro, L., Mani, I., Sundheim, B., & Wilson, G. (2001). *TIDES temporal annotation guidelines, version 1.0.2* (Tech. Rep.). MITRE.

Forăscu, C., Ion, R., & Tufiş, D. (2007). Semi-automatic annotation of the romanian timebank 1.2. In S. K. Constantin Orăsan (Ed.), *Proceedings of the ranlp 2007 workshop on computer-aided language processing.* Borovets, Bulgaria.

Forăscu, C., & Tufiş, D. (2012). Romanian timebank: An annotated parallel corpus for temporal information. In N. Calzolari et al. (Eds.), *Proceedings of the eight international conference on language resources and evaluation.* Istanbul, Turkey: ELRA.

Frank, E., & Witten, I. H. (1998). Generating accurate rule sets without global optimization. In *Proceedings of the fifteenth international conference on machine learning* (p. 144-151).

Fürnkranz, J. (1999). Separate-and-conquer rule learning. *Artificial Intelligence Review*, *13(1)*, 3-54.

Gatt, A., Belz, A., & Kow, E. (2009). The tuna-reg challenge 2009: Overview and evaluation results. In *Proceedings of the 12th european workshop on natural language generation* (p. 174-182).

Gatt, A., Goudbeek, M., & Krahmer, E. (2011). Attribute preference and priming in reference production: Experimental evidence and computational modeling. In K. van Deemter, A. Gatt, R. van Gompel, & E. Krahmer (Eds.), *Proceedings of the workshop on the production of referring expressions.*

Gatt, A., Portet, F., Reiter, E., Hunter, J., Mahamood, S., Moncur, W., et al. (2009). From data to text in the neonatal intensive care unit: Using nlg technology for decision support and information management. *AI Communications*, *22*, 153-186.

Gatt, A., Sluis, I. van der, & Deemter, K. van. (2007). Evaluating algorithms for the generation of referring expressions using a balanced corpus. In *Proceedings of the eleventh european workshop on natural language generation* (p. 49-56).

Goldberg, E., Driedger, N., & Kittredge, R. (1994). Using natural-language processing to produce weather forecasts. *IEEE Expert*, *9 (2)*, 45-53.

Grosz, B., & Sidner, C. L. (1986). Attention, intention and the structure of discourse. *Computational Linguistics*, *12 (3)*, 175-204.

Guerrero Nieto, M., & Saurí, R. (2012). *Modes timebank 1.0.* Philadelphia. Available from `http://www.ldc.upenn.edu/Catalog/docs/LDC2012T01`

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2008). The weka data mining software: An update. *SIGKDD Explorations*, *11(1).*

Harris, M. (2008). Building a large-scale commercial nlg system for an EMR. In *Proceedings of the fifth international natural language generation conference.*

Hervás, R., & Gervás, P. (2009). Evolutionary and case-based approaches to reg: Nil-ucm-evotap, nil-ucm-valuescbr and nil-ucm-evocbr. In *Proceedings of the 12th european workshop on natural language generation* (p. 187-188).

Hitzeman, J., Moens, M., & Grover, C. (1995). Algorithms for analyzing the temporal structure of discourse. In *Eacl '95 proceedings of the seventh conference on european chapter of the association for computational linguistics* (p. 253-260).

Hobbs, J. (2002). Toward an ontology for time for the semantic web. In *Proceedings of the lrec 2002 workshop annotation standards for temporal information in natural language* (p. 28-35). Las Palmas, Spain.

Hobbs, J., & Pustejovsky, J. (2003). Annotating and reasoning about time and events. In *Proceedings of the aaai spring symposium on logical formalizations of commonsense reasoning.* California.

Hockett, C. (1960). The origin of speech. *Scientific American*, *203 (3)*, 88-96.

Hong, S. (1997). Use of contextual information for feature ranking and discretization. *IEEE Transactions on Knowledge and Data Engineering*, *9(5)*, 718-730.

Hong, S., Hosking, J., & Winograd, S. (1995). *Use of randomization to normalize feature merits* (Tech. Rep. No. RC-20072). IBM.

Hunter, J., Freer, Y., Gatt, A., Reiter, E., Sripada, S., Sykes, C., et al. (2011). BT-Nurse: Computer generation of natural language shift summaries from complex heterogeneous medical data. *Journal of the American Medical Informatics Association*, *18*, 621-624.

Hüske-Kraus, D. (2003). Text generation in clinical medicine – a review. *Methods of Information in Medicine*, *42*, 51-60.

Hüske-Kraus, D. (2003). Suregen 2: A shell system for the generation of clinical documents. In *Proceedings of the 10th conference of the european chapter of the association for computational linguistics* (p. 215-218).

Hwang, C., & Schubert, L. (1992). Tense trees as the fine structure of discourse. In *Acl '92 proceedings of the 30th annual meeting on association for computational linguistics.*

Im, S., You, H., Jang, H., Nam, S., & Shin, H. (2009). Ktimeml: specification of temporal and event expressions in korean text. In *Proceedings of the 7th workshop on asian language resources* (p. 115-122).

Kahn, M., Fagan, L., & Sheiner, L. (1991). Combining physiologic models and symbolic methods to interpret time-varying patient data. *Methods of Information in Medicine*, *30*, 167-178.

Kamp, H., & Reyle, U. (1993). *From discourse to logic.* Kluwer.

Karamanis, N., Poesio, M., Mellish, C., & Oberlander, J. (2004). Evaluating centering-based metrics of coherence for text structuring using a reliably annotated corpus. In *Proceedings of the 42nd annual meeting on association for computational linguistics* (p. 391-398).

Kay, M. (1996). Chart generation. In *Acl '96 proceedings of the 34th annual meeting on association for computational linguistics* (p. 200-204).

Klein, D., & Manning, C. D. (2003). Fast exact inference with a factored model for natural language parsing. In *Advances in neural information processing systems* (Vol. 15, p. 3-10). Cambridge, MA: MIT Press.

Knight, K., & Marcu, D. (2000). Statistics-based summarization — step one: Sentence compression. In *The 17th national conference on artificial intelligence.*

Kotsiantis, S. B., Zaharakis, I. D., & Pintelas, P. E. (2006). Machine learning: a review of classification and combining techniques. *Artificial Intelligence Review*, *26(3)*, 159-190.

Krahmer, E., & Theune, M. (2002). Efficient context-sensitive generation of referring expressions. In K. van Deemter & R. Kibble (Eds.), *Information sharing: Reference and presupposition in language generation and interpretation.* CSLI Publications.

Kukich, K. (1983). Design of a knowledge-based report generator. In *Proceedings of the 21st annual meeting of the association for computational linguistics* (p. 145-150).

Labov, W. (2011). Oral narratives of personal experience. In P. Hogan (Ed.), *The cambridge encyclopedia of the language sciences.* Cambridge University Press.

Langkilde, I. (2000). Forest-based statistical sentence generation. In *Proceedings of the 1st north american chapter of the association for computational linguistics conference* (p. 170-177).

Langkilde, I., & Knight, K. (1998). Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 17th international conference on computational linguistics* (Vol. 1, p. 704-710).

Langkilde-Geary, I., & Knight, K. (2002). HALogen statistical sentence generator. In *Proceedings of the acl-02 demonstrations session* (p. 102-103). Philadelphia.

Language resource management – Semantic annotation framework – Part 1: Time and events (Norm No. ISO 24617-1). (2012). ISO, Geneva, Switzerland.

Lapata, M. (2003). Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the annual meeting of the association for computational linguistics.*

Lavoie, B., & Rambow, O. (1997). A fast and portable realizer for text generation systems. In *Proceedings of the fifth conference on applied natural language processing (anlp97)* (p. 265-268).

Lester, J., & Porter, B. (1997). Developing and empirically evaluating robust explanation generators: The knight experiments. *Computational Linguistics*, *23 (1)*, 65-101.

Lim, T.-S., Loh, W.-Y., & Shih, Y.-S. (2000). A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, *30*, 203-229.

Llorens, H., Saquete, E., & Navarro-Colorado, B. (2010). Timeml events recognition and classification: learning crf models with semantic roles. In *Proceedings of the 23rd international conference on computational linguistics* (p. 725-733). Beijing.

Mairesse, F., & Walker, M. (2011). Controlling user perceptions of linguistic style: Trainable generation of personality traits. *Computational Linguistics*, *37(3)*, 455-488.

Mani, I. (2001). *Automatic summarization.* John Benjamins Publishing Agency.

Mani, I., Wellner, B., Verhagen, M., Lee, C., & Pustejovsky, J. (2006). Machine learning of temporal relations. In *Proceedings of the 44th annual meeting of the association for computational linguistics.* ACL.

Mani, I., & Wilson, G. (2000). Robust temporal processing of news. In *Pro-*

ceedings of the 38th annual meeting on association for computational linguistics (p. 69-76).

Mann, W., & Thompson, S. (1988). Rhetorical structure theory: Toward a functional theory of text organization. *Text*, *8(3)*, 243-281.

Manurung, H., Ritchie, G., & Thompson, H. (2000). A flexible integrated architecture for generating poetic texts. In *Proceedings of the fourth symposium on natural language processing.*

Manurung, R., Ritchie, G., Pain, H., Waller, A., OâĂŹMara, D., & Black, R. (2008). The construction of a pun generator for language skill development. *Applied Artificial Intelligence*, *22*, 841-869.

Marcu, D. (1997a). From local to global coherence: A bottom-up approach to text planning. In *Proceedings of the 14th national conference on artificial intelligence* (p. 629-635).

Marcu, D. (1997b). The rhetorical parsing of natural language texts. In *Eacl '97 proceedings of the eighth conference on european chapter of the association for computational linguistics* (p. 96-103).

Mazur, P., & Dale, R. (2009). The dante temporal expression tagger. In *Proceedings of the 3rd language and technology conference* (p. 245-257).

McCoy, K. F., Pennington, C. A., & Suri, L. Z. (1996). Considering the effect of second language learning on generation. In *Proceedings of the eighth international workshop on natural-language generation (inlg-1996).*

McDonald, D. (1993). Issues in the choice of a source for natural language generation. *Computational Linguistics*, *19(1)*, 191-197.

McKeown, K. (1985a). Discourse strategies for generating natural-language text. *Artificial Intelligence*, *27 (1)*, 1-41.

McKeown, K. (1985b). *Text generation: Using discourse strategies and focus constraints to generate natural language text.* Cambridge University Press.

Mellish, C., Evans, R., Cahill, L., Doran, C., Paiva, D., Reape, M., et al. (2000). A representation for complex and evolving data dependencies in generation. In *In proceedings of the applied natural language processing (anlpnaacl2000) conference.*

Mellish, C., Knott, A., Oberlander, J., & O'Donnell, M. (1998). Experiments using stochastic search for text planning. In *Proceedings of the 9th international workshop on natural language generation* (p. 98-107).

Mellish, C., Scott, D., Cahill, L., Paiva, D., Evans, R., & Reape, M. (2006). A reference architecture for natural language generation systems. *Natural Language Engineering*, *12(1)*, 1-34.

Miller, G. (1995). Wordnet: A lexical database for english. *Communications of the ACM*, *38(11)*, 39-41.

Minnen, G., Carroll, J., & Pearce, D. (2001). Applied morphological processing of english. *Natural Language Engineering*, *7(3)*, 207-223.

Moens, M., & Steedman, M. (1988). Temporal ontology and temporal reference. *Computational Linguistics*, *14*(2), 15-28.

Murthy, S. K. (1998). Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, *2(4)*, 345-389.

Nakanishi, H., Miyao, Y., & Tsujii, J. (2005). Probabilistic models for disambiguation of an hpsg-based chart generator. In *Proceedings of the ninth international workshop on parsing technologies (iwpt)* (p. 93-102).

Nakhimovsky, A. (1988). Aspect, aspectual class, and the temporal structure of narrative. *Computational Linguistics*, *14 (2)*, 29-43.

Nenkova, A., & McKeown, K. (2003). *Improving the coherence of multi-document summaries: a corpus study for modeling the syntactic realization of entities* (Tech. Rep.). Columbia University, CS Department.

Oberlander, J., & Lascarides, A. (1992). Preventing false temporal implicatures: Interactive defaults for text generation. In *Proceedings of the 15th international conference on computational linguistics* (Vol. 1, pp. 721–727).

Oberlander, J., O'Donnell, M., Knott, A., & Mellish, C. (1998). Conversation in the museum: experiments in dynamic hypermedia with the intelligent labelling explorer. *New Review of Hypermedia and Multimedia*, *4*, 11-32.

O'Donnell, M., Mellish, C., Oberlander, J., & Knott, A. (2001). ILEX: an architecture for a dynamic hypertext generation system. *Natural Language Engineering*, *7(3)*, 225-250.

Paiva, D., & Evans, E. (2005). Empirically-based control of natural language generation. In *Proceedings of the 43rd annual meeting of the association for computational linguistics* (p. 58-65).

Paris, C., Linden, K. V., Fischer, M., Hartley, A., Pemberton, L., Power, R., et al. (1995). A support tool for writing multilingual instructions. In *Proceedings of the fourteenth international joint conference on artificial intelligence.*

Parsons, T. (1990). *Events in the semantics of english: A study in subatomic semantics* (Vol. 19). Cambridge, MA: MIT Press.

Partee, B. H. (1973). Some structural analogies between tenses and pronouns in english. *The Journal of Philosophy*, *70 (18)*, 601-609.

Passonneau, R. (1988). A computational model of the semantics of tense and aspect. *Computational Linguistics*, *14*(2), 44-60.

Perner, P. (2001). Improving the accuracy of decision tree induction by feature pre-selection. *Applied Artificial Intelligence*, *15(8)*, 747-760.

Pnueli, A. (1977). The temporal logic of programs. In *Proceedings of the 18th ieee symposium on foundations of computer science.* The Temporal Logic of Programs.

Portet, F., Reiter, E., Gatt, A., Hunter, J., Sripada, S., Freer, Y., et al. (2009). Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence*, *173 (7–8)*, 789–816.

Prior, A. (1967). *Past, present and future.* Oxford University Press.

Puçcaşu, G. (2007). Wvali: Temporal relation identification by syntactico-semantic analysis. In *Proceedings of the fourth international workshop on semantic evaluations* (p. 484-487).

Pustejovsky, J., Castaño, J., Ingria, R., SaurÃŋ, R., Gaizauskas, R., Setzer, A., et al. (2003). TimeML: Robust specification of event and temporal expressions in text. In *Proceedings of the fifth international workshop on computational semantics.*

Pustejovsky, J., Hanks, P., Sauri, R., See, A., Day, D., Ferro, L., et al. (2003). The TimeBank corpus. In *Proceedings of corpus linguistics* (p. 647-656).

Pustejovsky, J., Lee, K., Bunt, H., & Romary, L. (2010). ISO-TimeML: An international standard for semantic annotation. In N. Calzolari et al. (Eds.), *Proceedings of the seventh international conference on language resources and evaluation* (p. 394-397). Valletta, Malta: European Language Resources Association.

Pustejovsky, J., Littman, J., Saurí, R., & Verhagen, M. (2006). *TimeBank 1.2. documentation.*

Quinlan, R. (1993). *C4.5: Programs for machine learning.* San Mateo, CA: Morgan Kaufmann Publishers.

Reichenbach, H. (1947). *Elements of symbolic logic.* Macmillan & Co.

Reiter, E. (1994). Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? In *Proceedings of the seventh annual workshop on natural language generation.*

Reiter, E. (2010). Natural language generation. In A. Clark, C. Fox, & S. Lappin (Eds.), *Handbook of computational linguistics and natural language processing* (pp. 574–598). John Wiley & Sons.

Reiter, E., & Dale, R. (1992). A fast algorithm for the generation of referring expressions. In *Proceedings of the 14th conference on computational linguistics* (p. 232-238).

Reiter, E., & Dale, R. (1997). Building applied natural language generation systems. *Natural Language Engineering*, *3(1)*, 57-87.

Reiter, E., & Dale, R. (2000). *Building natural language generation systems.* Cambridge University Press.

Reiter, E., Gatt, A., Portet, F., & Meulen, M. van der. (2008). The importance of narrative and other lessons from an evaluation of an nlg system that summarises clinical data. In *Proceedings of the fifth international natural language generation conference* (p. 147-156).

Reiter, E., Robertson, R., & Osman, L. (2003). Lessons from a failure: Generating tailored smoking cessation letters. *Artificial Intelligence*, *144*, 41-58.

Reiter, E., Sripada, S., Hunter, J., Yu, J., & Davy, I. (2005). Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, *167*, 137-169.

Saquete, E., Martinez-Barco, P., Muñoz, R., Negri, M., Speranza, M., & Sprugnoli, R. (2006). Multilingual extension of a temporal expression normalizer using annotated corpora. In *Proceedings of the international workshop on cross-language knowledge induction* (p. 1-8).

Sarbin, T. R. (1986). Narrative psychology: The storied nature of human conduct. In T. R. Sarbin (Ed.), (p. 3-21). New York: Praeger.

Saurí, R., & Badia. (2010). *Spanish timebank 1.0.* Available from `http://www.ldc.upenn.edu/Catalog/docs/LDC2012T12/`

Saurí, R., & Badia, T. (2012). *Catalan timebank 1.0.* Philadelphia. Available from `http://www.ldc.upenn.edu/Catalog/docs/LDC2012T10/`

Saurí, R., Knippen, R., Verhagen, M., & Pustejovsky, J. (2005). Evita: a robust event recognizer for qa systems. In *Proceedings of the conference on human language technology and empirical methods in natural language processing* (p. 700-707). Stroudsburg.

Saurí, R., Littman, J., Knippen, B., Gaizauskas, R., Setzer, A., & Pustejovsky, J. (2006). *TimeML annotation guidelines version 1.2.1.* Available from `http://www.timeml.org/`

Setzer, A. (2001). *Temporal information in newswire articles: an annotation scheme and corpus study.* Unpublished doctoral dissertation, University of Sheffield.

Smith, C. (2003). *Models of discourse: The local structure of texts.* Cambridge University Press.

Spärck Jones, K. (2007). Automatic summarising: The state of the art. *Information Processing and Management*, *43*, 1449-1481.

Spreyer, K., & Frank, A. (2008). Projection-based acquisition of a temporal labeller. In *Proceedings of the third international joint conference on natural language processing* (p. 489-496). Hyderabad, India.

Sripada, Y., Reiter, E., Hunter, J., & Yu, J. (2002). *Sumtime-meteo: Parallel corpus of naturally occurring forecast texts and weather data* (Tech. Rep.). Computing Science Department, University of Aberdeen.

Stede, M. (1996). *Lexical options in multilingual generation from a knowledge base* (G. Adorni & M. Zock, Eds.). Berlin/Heidelberg: Springer.

Steedman, M., & Baldridge, J. (2011). Non-transformational syntax: Formal and explicit models of grammar. In R. Borsley & K. BÃűrjars (Eds.), (p. 181-224). John Wiley & Sons.

Strötgen, J., & Gertz, M. (2012). Multilingual and cross-domain temporal tagging. *Language Resources and Evaluation*.

Thompson, H. (1977). Strategy and tactics: a model for language production. In *Papers from the 13th regional meeting of the chicago linguistic society.*

Turner, R., Somayajulu, S., Reiter, E., & Davy, I. (2008). Using spatial reference frames to generate grounded textual summaries of georeferenced data. In *Proceedings of the fifth international conference on natural language generation* (p. 16-24).

Vendler, Z. (1957). Verbs and times. *The Philosophical Review*, *66* (2), 143–160.

Verhagen, M. (2004). *Times between the lines.* Unpublished doctoral dissertation, Department of Computer Science, Brandeis University.

Verhagen, M. (2010). The brandeis annotation tool. In N. Calzolari et al. (Eds.), *Proceedings of the seventh international conference on language resources and evaluation* (p. 3638-3643). Valletta, Malta: ELRA.

Verhagen, M., Gaizauskas, R. J., Schilder, F., Hepple, M., Moszkowicz, J., & Pustejovsky, J. (2009). The TempEval challenge: identifying temporal relations in text. *Language Resources and Evaluation*, *43(2)*, 161-179.

Verhagen, M., Mani, I., Saurí, R., Knippen, R., Seok Bae Jang, Littman, J., et al. (2005). Automating temporal annotation with TARSQI. In *Proceedings of the acl 2005 on interactive poster and demonstration sessions* (p. 81-84).

Verhagen, M., Saurí, R., Caselli, T., & Pustejovsky, J. (2010). Semeval-2010 task 13: Tempeval-2. In *Proceedings of the 5th international workshop on semantic evaluation* (p. 57-62). Uppsala, Sweden.

Viethen, J. (2011). *The generation of natural descriptions: Corpus-based investigations of referring expressions in visual domains.* Unpublished doctoral dissertation, Macquarie University, Sydney, Australia.

Walker, M., Stent, A., Mairesse, F., & Prasad, R. (2007). Individual and domain adaptation in sentence planning for dialogue. *Journal of Artificial Intelligence Research*, *30*, 413-456.

Wanner, L., & Hovy, E. (1996). The HealthDoc sentence planner. In *Proceedings of the eighth international workshop on natural-language generation (inlg-1996)* (p. 1-10).

Webb, G. (1999). Decision tree grafting from the all tests but one partition. In *Proceedings of the sixteenth international joint conference on artificial intelligence* (p. 702-707).

Webber, B. (1988). Tense as discourse anaphor. *Computational Linguistics*, *14*(2), 61-71.

White, M. (2006). EfĭňĄcient realization of coordinate structures in combinatory categorial grammar. *Research on Language and Computation*, *4(1)*, 39-75.

Williams, S., & Reiter, E. (2008). Generating basic skills reports for low-skilled readers. *Natural Language Engineering*, *14(4)*, 495-525.

Wonsever, D., Rosé, A., Malcuori, M., Moncecchi, G., & Descoins, A. (2012). Event annotation schemes and event recognition in spanish texts. In *Proceedings of the 13th international conference on computational linguistics and intelligent text processing* (p. 206-218).

Yoshikawa, K., Riedel, S., Asahara, M., & Matsumoto, Y. (2009). Jointly identifying temporal relations with markov logic. In *Proceedings of the 47th annual meeting of the acl and the 4th ijcnlp of the afnlp* (p. 405-413).

Yu, J., Reiter, E., Hunter, J., & Mellish, C. (2007). Choosing the content of textual summaries of large time-series data sets. *Natural Language Engineering*, *13 (1)*, 25-50.

Zwaan, R., Madden, C., & Stanfield, R. (2001). Time in narrative comprehension: A cognitive perspective. In D. Schram & G. Steen (Eds.), *Psychology and sociology of literature* (pp. 71–86). John Benjamins Publishing Company.