

2016•2017
FACULTEIT BEDRIJFSECONOMISCHE WETENSCHAPPEN
*master in de toegepaste economische wetenschappen:
handelsingenieur in de beleidsinformatica*

Masterproef
Decision modeling : from text to model

Promotor :
Prof. dr. Koenraad VANHOOF

Valerie Kerkhofs
*Scriptie ingediend tot het behalen van de graad van master in de toegepaste economische
wetenschappen: handelsingenieur in de beleidsinformatica*

2016•2017

FACULTEIT BEDRIJFSECONOMISCHE
WETENSCHAPPEN

*master in de toegepaste economische wetenschappen:
handelsingenieur in de beleidsinformatica*

Masterproef

Decision modeling : from text to model

Promotor :
Prof. dr. Koenraad VANHOOF

Valerie Kerkhofs

*Scriptie ingediend tot het behalen van de graad van master in de toegepaste economische
wetenschappen: handelsingenieur in de beleidsinformatica*

Acknowledgments

This master thesis is the end of my Master of Business and Information Systems Engineering. The knowledge and skills I gained over the five years of study are used to complete this master thesis. During my study, I became more and more interested in the field of business process modelling and decision modelling. Choosing a topic that is related to these fields was an easy decision. The topic that I eventually have chosen is about automatically finding decisions in a decision model. During my study, I had projects where it was the assignment to construct a decision model. The information that was available was often long and contained lots of extra information. So from my limited experience with decision modelling I know that it can be useful to have a standard method for finding decisions.

Without the help of other people, I would not be able to complete this master thesis. Especially I would like to take the time to thank my supervisor prof. dr. Koen Vanhoof. He was always available for feedback or help when needed. Not only he supported me during the process of writing this master thesis he also brought me in contact with the concepts of business process modelling and decision modelling during one of his courses. His classes were the basis of my interest in these topics.

Finally, I would like to thank my family and friends. They were always ready to encourage me when I needed it. A special appreciation goes to my parents who not only supported me this year but also during my whole study period. They gave me the opportunity to develop myself to the person I am today. To end, I want to thank everyone who was involved with this master thesis, without them I was not able to deliver this result.

Valerie Kerkhofs

Mei, 2017

Summary

Within a company decisions are taken every day. For this company, it is important that these decisions can be taken in an efficient way. Due to an efficient way of working money can be saved. Modelling decisions ensures that decisions will be made more successfully. Decision rules are created and by using the input data the output of the decision can be found. When the input data does not change, the outcome of the decision also will remain the same. Executing decisions with a decision model will ensure more consistency than an execution without a decision model.

The decision rules that are necessary to create the decision model need to be derived from the data or available knowledge in the company. This information can be stored in a structured way. Hence there is also the possibility that this information is captured in long elaborated sometimes hard to understand texts. There can be concluded that gathering all the required rules is an intensive step for creating a decision model.

The available data can take several forms. The data can consist of only numbers, only text or a mixture of both. The data can also be structured or non-structured. All these differences provoke that information extraction cannot be done in a standardized way. A method to make the information extraction easier is needed.

This exploratory research checked if there is an easier method to get the needed information from the textual data. This method will be created with the help of natural language processing techniques. The techniques will be executed with the Natural Language Toolkit. Natural language processing is a method to analyse text. Through the different levels of a text different analysis can be done. These levels go from word level till the level that is applicable to the whole text.

Within natural language processing there is the possibility to search for several structures. These structures are presented by a grammar. When a specific grammar is defined for a decision then in theory these decisions can be separated from the rest of the text. During the research the grammar that is specific for a decision will need to be defined. When the grammar is defined then there needs to be checked if it is possible to use it to find decisions.

At the end of the research there needs to be examined if the created method will have any added value to the process of modelling decisions. The time that is needed to create a new model by using the method cannot be longer than without using the method.

Table of content

1	Introduction	1
1.1	Research questions	2
1.1.1	Main research question	2
1.1.2	Sub questions	2
1.2	Research methodology	3
2	Decision modelling	5
2.1	What is decision modelling?	5
2.2	Scope of decision modelling	6
2.3	Creating a decision model	7
2.4	Business rules	10
2.5	Current problems.....	11
3	Natural language processing.....	13
3.1	What is natural language processing?.....	13
3.2	How to perform natural language processing?	14
3.3	Current applications?	15
3.4	Conclusion	16
4	Argument mining	17
4.1	Argument detection	18
4.2	Current applications of argument mining	19
4.3	Challenges for argument mining	19
5	Natural Language Toolkit	21
5.1	About the tool	21
5.2	Information extraction.....	22
5.2.1	Information extraction architecture.....	22
5.2.2	Chunking.....	23
5.2.3	Chinking.....	26
5.2.4	Named entity recognition	26
5.3	Classifying text.....	26
5.3.1	Supervised classification.....	27
5.3.2	Decision trees	27
5.3.3	Conclusion.....	28

6	Stanford parser.....	29
6.1	About the parser.....	29
6.2	Dependencies.....	29
6.3	Executing the Stanford parser	30
6.4	GrammarScope	31
7	Conclusion	39
7.1	Further research.....	40
	References.....	41
	List of figures	43
	List of tables	45
	List of boxes	47
	Appendices	49
	Appendix 1	49
	Appendix 2	51
	Appendix 3	57
	Appendix 4	61
	Appendix 5	63
	Appendix 6	65

1 Introduction

Daily many decisions need to be taken in an organization. If the decision making is not happening efficiently this can be a time-consuming task. The organization will benefit from making decisions fast and in a consistent way. Decision modelling will help with making decisions more efficient. An overview of the business rules and business processes will be given with decision modelling. For each specific case a company can decide in an efficient way what to do. During the creation of a decision model the business rules need to be stated. These rules can be described in legal texts, policies or business rules stored in a text document. By searching in the text documents for key words, decisions can be found. With these decisions, the models can be created (Stöhr, 2016).

Besides the many decisions that organizations need to make, companies store a lot of data. This data can be structured or unstructured. Structured data is data that is fixed for a certain field in a file. For example, in an excel file the fields of one column can be a product name and the fields of the second column can be the amount purchased. With unstructured data, the data is not fixed for a specific field. An example is a comment; this comment can be as long as the person who inserts it wants and it can have all types of characters. Structured data is easy to analyse. With running an SQL query for example more insights about the captured data can be found out. Such an SQL query cannot be executed on unstructured data. (Chakrabarti, 2003)

Unstructured data is less easy to analyse. The stored data has most of the times a different format. Tan (1999) states that 80 percent of the data in a company is stored as a text document. The unstructured data also contains important information so without analysing this, information will be lost. (Tan, 1999)

From the data that companies store a considerably amount will be linguistic data. To analyse this data natural language processing can be used. Natural language processing techniques are currently used to extract information from documents. With these techniques, the structure of a text can be found. Several tools exist for natural language processing and they can use different terms for natural language processing. Different names are: text processing, knowledge discovery in text, intelligent text analysis and natural language processing (Tan, 1999). From this point on the term natural language processing will be used.

Because most of the stored data are text documents the commercial potential of natural language processing is higher than for data mining. For companies, it can be more valuable to get also an analysis from texts and not only from numbers because numbers are not saying everything. (Tan, 1999)

As mentioned before it is not easy to find decisions in a text that contains a lot of information. With this master thesis, an exploratory research will be done to check if it is possible to use natural language processing and argument mining on unstructured text documents to define the decisions. At the moment, there is no link between decision modelling and natural language processing. The possibility of linking these two subjects will be controlled. There will also be checked if there is a benefit of linking them. The natural language processing tool can be used either as a tool to find the decision or as a check to look if the right decisions are found.

Natural language processing can be used to find the decisions that are clearly defined. An example is "Does the customer has a driving licence?". The answer to this question is always either yes or no. There are also decisions where the answer is relying on a lot of conditions. For these decisions, it is not always clear what is the right outcome. To find these decisions argument mining can be used.

When decisions are extracted manually from a text a person can make his or her own assumptions when something is not exactly explained. If different people make a decision model from the same text most of the times the models will contain some differences. With the use of natural language processing and argument mining the possibility to make your own assumptions will be limited.

The text is constructed starting from the following research questions. Consequently, an answer to the following questions can be found while reading the text.

1.1 Research questions

1.1.1 Main research question

- Is it possible to find decisions in a text by using natural language processing?

The outcome of the master thesis will be a conclusion on the possibility of using natural language processing with decision modelling. The decision will be made with keeping in mind the efficiency of the task and the degree of difficulty.

1.1.2 Sub questions

The following sub questions will help to find an answer to the main question and will give additional insights.

- What is decision modelling?

Before making a decision on how natural language processing can find decisions the requirements for a decision model need to be known. A study on decision modelling will help with defining the requirements.

- What is natural language processing?

More insights on natural language processing are needed to be able to apply the concept of natural language processing on decision modelling. It is important to know what is necessary to perform natural language processing.

- What is argument mining?

More knowledge will be gathered about argument mining. There also needs to be reviewed if the concept of argument mining has an added value for natural language processing. Is the evaluation in natural language processing running different than for argument mining?

- Which are the existing natural language processing tools and which characters do they have?

Getting an overview of all the existing natural language processing tools and the specific features. Compare the tools in the way they are working. When looking at the different tools there can be concluded which tool is more appropriate to use to create decision models.

To answer these questions this master thesis is constructed as follows. In the first chapter, there will be a detailed discussion on decision modelling. After the chapter a clear view is created on what is decision modelling and what are the shortcomings. The following chapter will explain the concept of natural language processing to get a better idea why it is useful for decision modelling. In the shadow of natural language processing argument mining will be discussed. After all the theories are explained the concepts will be applied in practice. First the Natural Language Toolkit will be explained and afterwards GrammerScope.

1.2 Research methodology

To be able to make a conclusion for this master thesis an exploratory research will be performed. The methodology that is used to gain more insights is explained in this part.

A literature study will be completed to learn more about the used concepts. The concepts that will be discussed are decision modelling, natural language processing and argument mining. Scientific papers are used to find the needed information. The theories that are used in the literature study will help to answer the research questions.

After the knowledge about the concepts is gathered a practical study will be completed. Two tools are tested, NLTK and GammarScope. For every tool, some background is given about the way they work. The techniques of both tools will be tested on an example text that contains knowledge for making decisions. The example texts that are used come from an exercise book about business process modelling and decision modelling. After the techniques are applied to a text there will be a manual check if all the decision are contained in the outcome.

A combination of the literature study and the practical study will be used to make a conclusion. As this is an exploratory study the outcome will be a starting point for further research.

2 Decision modelling

The following part will explain the concept of decision modelling. There will be started with defining what decision modelling is and what the scope is. Afterwards the creation of a decision model will be explained and some problems that can arise are highlighted.

2.1 What is decision modelling?

For companies to be able to perform their daily activities they have processes. These processes are sequences of activities to become a certain result. To understand the processes and to be able to automate them these processes are documented in a business process model. A business process model is a visual representation of a specific process. The number of unique processes can be high so documentation is recommended. A business process model consists of different types of activities. One of these activities can be a decision. Decisions are captured in the process model but they are not explicitly modelled in it. In a business process model, they are presented by one activity. To model a detailed decision, a decision model can be used. With the decision model a clear view can be generated about the structure of the decision that needs to be taken (Taylor, Fish, Vanthienen, & Vincent, 2013).

The reason why a decision should be modelled is because companies benefit from it. Many decisions are taken daily. These decisions can be easy or more difficult to execute. In both cases when the decisions are modelled these can be executed in a fast and consistent way. After the execution of the decision the outcome will influence other activities so a company will only benefit from a good evaluation (*Decision model and notation*, 2014). Another reason is that with decision modelling the decision logic is clearly stated and not in possession of a specific person. Often it is the case that one person knows the best practices for evaluating a decision. With decision modelling the business rules can be executed by anyone and the execution of the decision is not dependent on that one person. The rules need to be stated one time and then can be executed without mistakes as many times as necessary (von Halle, & Goldberg, 2010).

The characters of decision modelling enforce the interest in decision modelling even more. The use of technologies is not necessary for organizing business logic. This makes decision modelling accessible and doable for almost everyone. The model purely represents the business logic in a simple structure with a declarative nature and optimal integrity. There are no biases from a process, data or technology. Decision modelling is independent of technology but can be easily implemented in different technologies (von Halle, & Goldberg, 2010).

Decision-making has three perspectives. In business process models decisions are captured in activities, this is the first perspective. Process models will provide an overview of the decision activity in combination with the other activities from the process. The second perspective is the decision requirements diagram. A decision requirements diagram is the link between business process models and decision logic models. All the required inputs and relations are presented in this diagram. The last perspective is the decision logic. In the decision logic, all the rules for the evaluation of a decision are specified. The second and third perspective are captured in a decision model (*Decision model and notation*, 2014).

2.2 Scope of decision modelling

Decision modelling can be used in three cases. First, decision modelling will help with modelling human decision-making. To model human decisions a decision requirements diagram can be used. The decisions that need to be modelled are most of the times described in natural language and not in decision logic. There can be different knowledge sources. Knowledge can come from people, regulatory bodies, documents or bodies of legislation. All these different sources will store their knowledge in a different way. Consequently, the knowledge extraction will not happen in a consistent way (*Decision model and notation*, 2014).

Decision modelling can also be used to model requirements for automated decision-making. This automation will create extra value for the company. The value that will be created can be measured with the decision yield. This decision yield will evaluate the performance of the decision-making using five dimensions. In figure 1 is illustrated how these five dimensions work together (Fish, & Taylor, 2012).

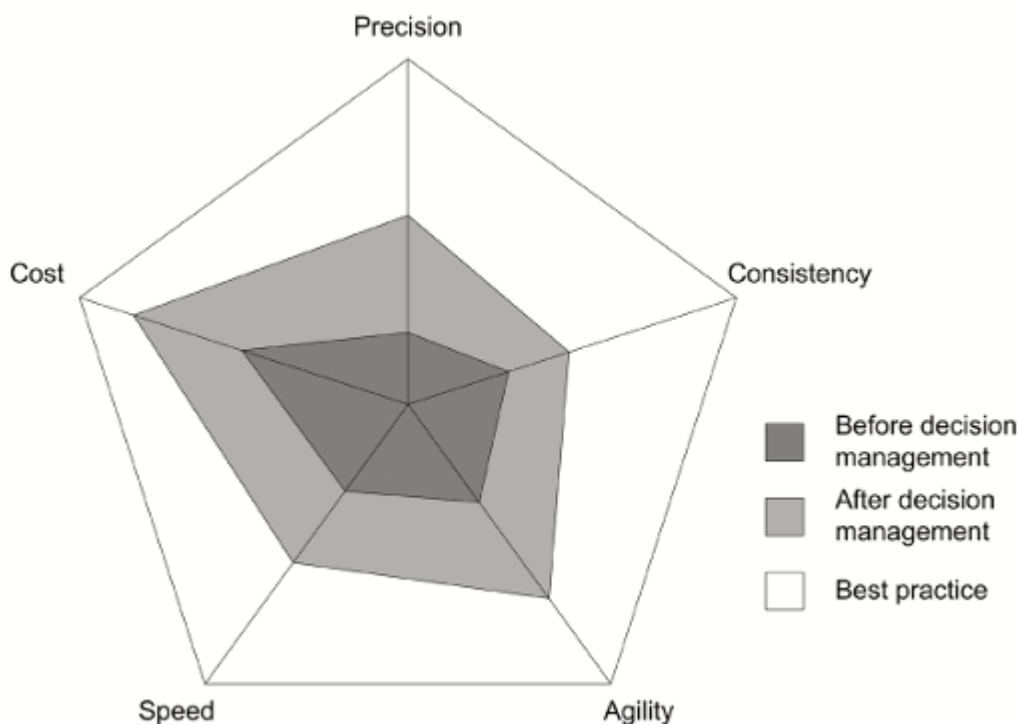


Figure 1: the dimensions of decision management

Precision states how effective the decisions are. To determine the precision of a decision the expected outcomes of for example revenue or losses will be compared with the real values. The integration and coordination of a decision in a company is measured by the consistency. Consistency is not measured for one specific time or product but can be measured over time, for different channels and different product lines. The agility to change decision rules is also an important dimension. This dimension states if the rules can easily be changed. For a company, it is important to implement new information about decision rules in a fast and consistent way. The

speed of executing decisions is the fourth dimension that is measured. Executing decisions includes some costs. This cost is the last dimension that will influence the decision yield (Fish, & Taylor, 2012).

On the graph above there is an illustration before decision management is done and after decision management is done. When the five dimensions cannot be improved anymore, the full figure will be coloured. The illustration shows that after decision management the five dimensions are improved. So, the dimensions are better than before decision management. This means an increase in precision, a decrease in costs, increased speed, increased agility and increased consistency (Fish, & Taylor, 2012).

The use of decision models will solve some issues. In rule systems, it can happen that there are duplications of rules in the system. By defining a decision table this can be avoided. Validating the decision table ensures that there will be checked if some rules are missing or if rules conflict with each other. The following problem is relating to the acquisition process. Communication to customers and service for the customer will benefit from the use of a decision model. Consistent decisions can be made in a fast way by evaluating the created decision model. This will avoid exceeding service times and inconsistent decision making (Taylor, Fish, Vanthienen, & Vincent, 2013).

2.3 Creating a decision model

A modelling standard is created for constructing decision models. This standard is the decision modelling notation (DMN) and is created by the Object Management Group (OMG) (*Decision model and notation*, 2014).

Decision modelling consists of four main steps. These are identifying decisions, describing decisions, specifying decision requirements and creating the model (Decision modelling with DMN, 2016). The four steps of decision modelling can be represented in two levels. A decision model consists of two levels that are used to construct a model. The first one is the decision requirements level and the second the decision logic. Level one is an abstract model of the decision and level two is a detailed model of the decision (Taylor, Fish, Vanthienen, & Vincent, 2013).

To get to know the decision requirements the business logic needs to be gathered. This can be done by asking domain experts about the subject. It is also possible that the knowledge is present in a text file. In this case the rules need to be defined by reading the file (Taylor, Fish, Vanthienen, & Vincent, 2013).

The decision requirements level will give an abstract overview of all the elements that are used to construct the decision model. Four types of elements can be used and these elements are linked to business concepts. The decision requirements level is presented by the decision requirements diagram (DRD) and the decision requirements graph (DRG). The decision requirements graph consists of a decision, business knowledge, input data and a knowledge source (Taylor, Fish, Vanthienen, & Vincent, 2013).

The input data element is linked to the data that will be used for executing the decision. All the input data combined will define the case of a decision. So, without the input data there is no possibility for executing the decision. The second element is a decision element. This element is linked to the actual decision that needs to be taken. The decision element will use the input data and determine the output. The transfer from input to output will be processed by the decision logic. The decision logic can be linked to one or more business knowledge models. The third element gives more information about the source of a decision. The source of a decision can be an authority, business knowledge model or a policy manual. All these sources can be represented by the business knowledge element. In table 1 the graphical representation of all these elements can be found (Taylor, Fish, Vanthienen, & Vincent, 2013).

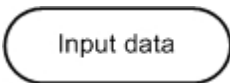
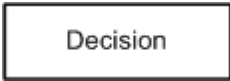
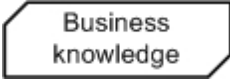
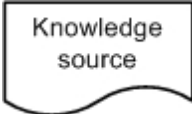
Input data	
Decision	
Business knowledge	
Knowledge source	

Table 1: the elements of a decision model

Business knowledge can be captured as business rules. Because the aim of this master thesis is to search for business rules in a text the next paragraph will have a more elaborate view on business rules.

The four elements of the requirements level are related with each other. The relations of these elements are shown in a decision requirements diagram. In figure 2 an example can be found of a decisions requirements diagram. A decision about the eligibility needs to be made. This decision has as input an application form and the outcome of the 'risk' decision. The 'risk' decision has as input information about the customer behaviour. The knowledge source for the 'eligibility' decision comes from the policy rules that are provided by the policy group. For the 'risk' decision the knowledge is gathered from a score model.

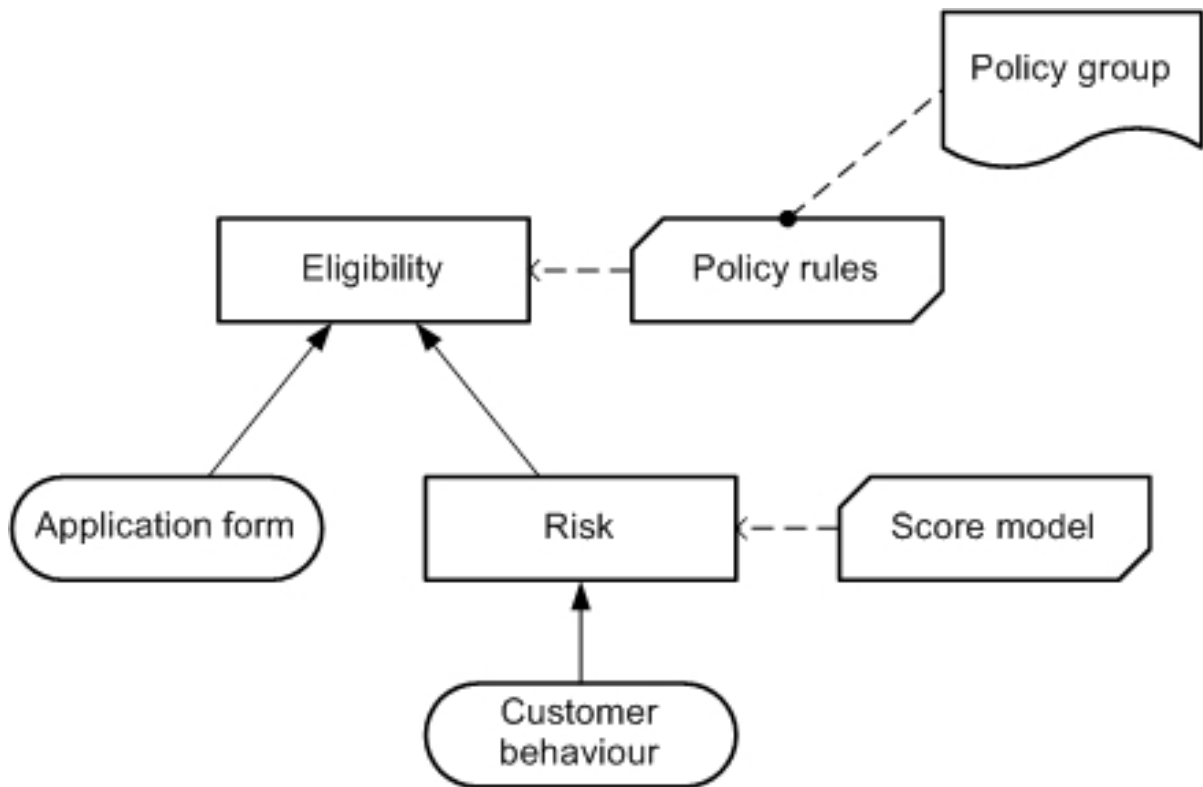


Figure 2: a decision model about eligibility

The second level is the decision logic. This level is related to the decision element. For this level, there is a specific language created for providing the logic. The Friendly Enough Expression Language (FEEL) which has specific syntax and semantics for stating the decision logic. The definition of this semantics and syntax is made using grammar rules for the composition of the expressions and semantic rules for the composition of the meaning of a complex expression. An expression defined in the FEEL language has the same interpretation in every type of implementation (Taylor, Fish, Vanthienen, & Vincent, 2013).

On the other hand, there are boxed expressions. These are used to present the decision logic in a standard way. The decision will be linked to elements from the decision requirements level. Due to the standardised way of structuring the decision logic it is possible to validate the logic automatically. With the same reasoning the execution of the decision making can also be automated. A decision requirements diagram and a boxed expression together are a complete decision model (Taylor, Fish, Vanthienen, & Vincent, 2013).

The decision logic that is present in the boxed expression is defined by rules. These rules which are described in the knowledge source, have a certain quality. This quality should be remained after the rules are converted in a decision table. The check for completeness, correctness and validation is a good way to keep the existing quality. Rules can change so also after maintenance the quality should be assured. A decision table will be a good support in assuring quality in both stages (Taylor, Fish, Vanthienen, & Vincent, 2013).

2.4 Business rules

Business rules are captured in the decision logic. The output of a decision is determined by the business rules. A business rule can be transformed to an action and has an advising goal. On the one hand, a business rule can obligate people to do something but it can also give guidelines about what are best practices. The form of a business rule can differ. It can be displayed in a tabular form or in a statement (Gailly, & Geerts, 2013).

There are eight ways to present a business rule. The different patterns can be used to perform the natural language processing. In the next part a brief overview of the patterns will be given (Gailly, & Geerts, 2013).

To explain the patterns some specific terminology is used. An economic event is the event that causes a change in the resources due to production, consumption or distribution. An economic resource is an element that is limited and can be used limited. The company controls the economic resources. A commitment is the arrangement that an economic event will be executed at a specific time stamp. An economic agent is the person who has control over the economic events and commitments (Gailly, & Geerts, 2013).

Pattern 1: "commitments should not be preceded by their fulfilment economic events"

An economic event does not need to be followed by a commitment in case the commitment and the economic event are linked to each other.

Pattern 2: "no future economic events are recorded"

In the rule, there is a time indicator and this will indicate when the economic event occurred. The economic event can only take place when the time indicator is completed. For example, 'The car can be sold when the seats are repaired.' The economic event is 'selling the car' and the time indicator is 'when the seats are repaired'.

Pattern 3: "prices have a positive numeric value"

When prices are used in the business rule these prices should have a positive value. To this business rule more business rule patterns can be applied.

Patterns 4: "only existing resources can participate in an outflow/ decrement economic event"

When a resource is linked to an economic event (inflow or outflow) then this resource needs to exist.

Pattern 5: "economic resources transferred match the economic resources committed to"

When a resource will be transferred, the type of this resource should match the resource type that is committed to the economic event.

Pattern 6: "the number of items transferred should not exceed the number of items committed to"

When there is a specific amount of commitments defined for an economic event then this number cannot be exceeded. For example, if a task can be performed by one resource then it is not possible to assign two resources to that task.

Pattern 7: "only resources that have been transferred can be returned"

The incoming resources of an economic event can only be resources that transferred earlier to another event and now return.

Pattern 8: "a resource can be returned (inflow) only by the agent to whom the resource was transferred (outflow)"

If an agent transferred a resource to an economic event this agent should also return it to the original event.

As stated before the information to find these business rules is available in the enterprise. There are several ways in which this information can be stored. To find the rules a person will need to look at all the available information and try to compose the rules out of the information (Gailly, & Geerts, 2013).

2.5 Current problems

One of the main issues is the extraction of the decision rules. The decision modelling notation does not provide a clear way how to complete the specific steps for constructing a decision model. Decision modelling notation will only describe what should be included in these steps. As an example, the extraction of the decision rules is a case of trial and error. DMN captures the rules how to model specific cases and how to include all the rules but the user needs to identify the different cases (Taylor, Fish, Vanthienen, & Vincent, 2013).

The decision logic used to construct a decision model must be extracted from knowledge that is inside the company. This knowledge can be stored in a document that sometimes is rather complex (Taylor, Fish, Vanthienen, & Vincent, 2013). As mentioned before more than 80% of the data stored in a company are text files (Tan, 1999). These can be long and rather complex so hard to analyse.

3 Natural language processing

As stated before decisions are mostly gathered in text files. To analyse these text files natural language processing can be used. The next part will describe what natural language processing is, how it works and where it can be used for.

3.1 What is natural language processing?

Natural language processing is extracting information out of unstructured text. This extraction will happen with computerized techniques. The kind of texts that can be used for natural language processing are oral and written texts. A goal of natural language processing is better detecting the need of a user to give them a better answer (Liddy, 2005).

The core of natural language processing is to understand the natural language. There are three problems related to the understanding of natural language. They relate to thought processes, representation and meaning of the linguistic input and the world knowledge. To solve these three problems seven levels are created in natural language processing (Chowdhury, 2005). These levels can be linked to the levels of language. Currently the number of capabilities from a natural language processing system is linked to the different levels that can be covered by the system (Liddy, 2005).

The first level is phonology. In this level attention is paid to the sound of words and the sound between words. A phonological analysis has three rules; phonetic rules, phonemic rules and prosodic rules. Sounds within a word are handled by phonetic rules. When words are spoken together the pronunciation can differentiate. This difference is analysed by phonemic rules. Prosodic rules will analyse the variation in intonation and stress of words in a sentence (Liddy, 2005).

Morphology is the second level and this level will pay attention to the nature of words. Every word is composed of morphemes. A morpheme is the smallest unit of meaning in a word. By breaking up a word in its different morphemes a meaning of a word can be found. A natural language processing system can perform this procedure to find a meaning of a word (Liddy, 2005).

The next level, lexical, will make an interpretation of an individual word. One word can have different meanings in different contexts. This analysis will give a first insight in which is the right meaning for the word in the specific context. The natural language processing system can use lexicons to find the right meaning of a word (Liddy, 2005).

Furthermore, words can be analysed in a sentence this is the syntactic level. The analysis will be done by having a look at the grammatical structure of the sentence. Dependencies between words in the sentence will be found during the analysis (Liddy, 2005).

The fifth level is semantic. Here will be looked at the whole sentence to discover the meaning. A part of the analysis are the interactions of word-level meanings. In this level words with different meanings will be analysed regarding the rest of the sentence. After this analysis one word can only have one meaning (Liddy, 2005).

When the analysis of a sentence is done, bigger parts of the text will be analysed in the discourse level. Connections between sentences will be made. There are two types of discourse analysis. The first is anaphora resolution. This will replace words and pronouns by the entity to which they refer when the word is semantically vacant. The second type is text structure recognition will analyse the function of a sentence in a text (Liddy, 2005).

The last level is the pragmatic level. In this level the analysis will try to reveal an extra meaning of the text without encoding them. Hidden meanings can be found in this level. Analytical knowledge is not sufficient at this level. Also, world knowledge, understanding of intentions, plans and goals are important. When natural language processing systems make an analysis at this level they will use knowledge bases to use some data that already was gathered. (Liddy, 2005).

3.2 How to perform natural language processing?

To perform natural language processing over the different levels there are four approaches. These four approaches are the symbolic approach, the statistical approach, the connectionist approach and the hybrid approach. All the approaches are different and will be reviewed and compared in the next section (Liddy, 2005).

The first approach is the symbolic approach which is focused on the linguistic aspect. Proven facts about language will be used in combination with algorithms. Rules developed by humans and lexicons will be used as the primary source of evidence. Rule-based systems are an example of the symbolic approach. Also, semantic networks are a good example of a symbolic approach. The network consists of nodes which represent objects or concepts and links which are the representation of relations between nodes. Semantic organizations can be derived from the structure of the network (Liddy, 2005).

The focus is more on mathematical techniques with the statistical approach. The data that is used for the analysis is only observed data. From high amounts of data generalized models will be derived. Only the knowledge present in the text will be used by this approach. Most of the times the statistical approach is used in speech recognition, lexical acquisition, part-of-speech tagging and others (Liddy, 2005).

Another approach is the connectionist approach. This approach is similar to the statistical approach because this approach also generates generalized models. The difference with the statistical approach is the combination of statistical learning with theories of representation. Within the connectionist approach it is harder to observe because the architectures can combine different theories. There is a difference made between two types of connectionist models. First there is the localist model which assumes that each unit is representing one concept. These localist models can be compared with semantic networks. The only difference is that the links are not labelled. The second type is the distributed model. For distributed models a concept is represented by a function of simultaneous activation of multiple units. Distributed models are mostly used with syntactic parsing, associative retrieval or limited domain translation tasks (Liddy, 2005).

Besides these different approaches there are also several frameworks of natural language processing defined. They all describe different phases during natural language processing. Two phases are used to describe natural language processing. The first phase is text refining. Within this phase some structure will be added to the text. The second one is knowledge distillation. During this phase patterns in the text will be searched (Tan, 1999).

Solka (2008) defines a framework for natural language processing that consists of three steps. The first one is extraction of features from the document. These features will capture the content of the document. The second is representing a way to measure the distance between the documents. The distance between two documents will indicate if they are similar. After these two steps, different analysis can be made. For example, a cluster analysis, visualisation of the document information or discriminant analysis (Solka, 2008).

Another natural language processing method is argument mining. An argument is consisting of a piece of evidence. The argument states if a certain claim or statement is true or false. It can also define which is the right answer to a question. With argumentation mining, arguments can be found automatically in a text (Palau and Moens, 2009). Argument mining will be discussed in detail in the next chapter.

3.3 Current applications?

There are several applications for natural language processing. The reason why there are so many applications is that natural language processing is usable for almost every application that uses text. In the following part, some examples of applications will be discussed but keep in mind that more applications are existing (Liddy, 2005).

To start, natural language processing can be used for information retrieval when there is a significant amount of text in the application. It is also possible to extract information from a text. With information extraction, the information that is gathered will be presented in a structured representation or by using key elements. Question-answering, visualization or data mining are examples of information extraction (Liddy, 2005).

Answering to a query of a user is also possible with natural language processing. It is defined by question-answering. The outcome of question-answering will be the part of the text where the answer can be found (Liddy, 2005).

With natural language processing, a text can also be summarized. By using natural language processing, the text will be shortened but no content will be left out. The summarization will provide a shortened version of the original text (Liddy, 2005).

As there is a considerable amount of applications possible with natural language processing there are also some issues that needs to be solved. When a pattern is created that can be used for an analysis this pattern is created for the specific domain or task. This pattern is customised for the specific domain or task so when applying it to another domain or task it will not be optimal

anymore. This problem causes a high cost for applying complex natural language processing techniques on large natural language processing applications (Chowdhury, 2005).

A second issue is linked to the difficulty of analysing human behaviour. When a natural language processing system is created a factor for checking if it performs correctly is reliability. Questions can be asked if they are given the right outcome and if they are efficient. Often a system works efficient but due to the complex behaviour of a human they are not that reliable. Abnormalities in human behaviour will cause different outcomes. Due to these different outcomes, the reliability decreases (Chowdhury, 2005).

3.4 Conclusion

Natural language processing has different possibilities for application. For finding decisions in a text some of these methods can be useful. All the levels of analysis can be used to find different kinds of information in a text. The level that will be interesting for finding decisions is the semantic level. This level will do specific analysis for the meaning of a sentence. When natural language processing will be applied there will be looked for functionalities that do an analysis on the semantic level.

4 Argument mining

Arguments are used daily by people to express their reasoning, to make conclusions or when they have a discussion. Argumentation mining is a method to detect all the arguments in a text, classify them and present arguments in a structured way in a text. All of this happens by looking at the interactions with other arguments. Arguments consist out of two parts. The first part are reasons or premises and the second part is a proposal or conclusion. Argumentation mining does not include a validity or correctness check. The structure of argumentation is presented by a tree and the arguments are the leaves of the tree (Mochales, & Moens, 2011).

Argument mining can be applied to decision modelling. When there is more than one outcome for a decision it is sometimes not clear which decision needs to be made in which case. With argument mining it will be possible to find the preferred results in a text. The semantic load of the used text will also be viewed. This semantic load will help to evaluate some decisions (Garcia Villalba and Saint-Dizier, 2012).

A complete argumentation analysis is constructed out of several parts. All these parts need to be completed to have a complete analysis. The following parts need to be considered; linguistic constraints, domain dependent, conceptual relations and discourse structure (Mochales, & Moens, 2011).

The main element in argumentation mining is the argument. This argument will always be a combination of at least two propositions. A proposition is a sentence that explains something to make a statement. Knowing this, the definition of an argument can be elaborated. An argument can be presented by the nature and relations that hold between the propositions of an argument and the relations follow a scheme. This scheme will define relations between propositions and reasoning (Mochales, & Moens, 2011).

All the elements from argumentation mining are related to other arguments. The relations are provided by coordination, subordination or forming a multiple argument. Table 2 illustrates how the different types of relations work (Mochales, & Moens, 2011).

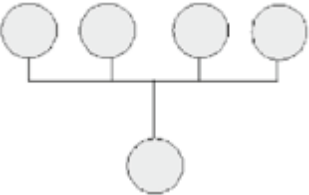
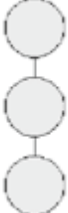
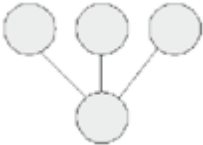
Relation by coordination	Relation by subordination	Multiple argumentation
		

Table 2: different relations in argument mining

With a relation by coordination one argument is linked to other arguments and all the other arguments are linked to each other. In the above example, one argument is linked to four other

arguments but the four arguments are not linked to each other. The second form is a relation by subordination where there is a linear relation between arguments. In the example above the first argument is linked with the second and the second argument is linked with the third argument. There is no direct link between the first and third argument. The last form is a multiple relation where one argument is linked to several arguments but the other arguments are not linked to each other. On this example one argument is linked to three arguments but the three arguments are not linked to each other (Mochales, & Moens, 2011).

4.1 Argument detection

At the beginning of the argument detection the classification problem will be analysed. For this analysis, statistical classifiers can be used. Each sentence will be presented as a vector of features before the information detection starts (Mochales, & Moens, 2011).

To detect arguments the maximum entropy model will be used. This model uses the principle of maximum entropy. This principle states that inferences from incomplete information should be drawn from the probability distribution that has the maximum entropy permitted by the information available (Mochales, & Moens, 2011).

At the beginning the classification problem needs to be solved. Sentences will be classified as argumentative or non-argumentative. Statistical classifiers can be used for this analysis. One of these classifiers is the maximum entropy model which is based on the maximum entropy principle. When information is incomplete and the inferences are based on this information then the interference should be derived from the probability distribution with the maximum entropy permitted with the information available. The second classifier is the naïve Bayes classifier. There will be a joint probability of the model $p(x,y)$ where x is an element and y is a label. Predictions will be made by using the Bayes rule. The following rule will be calculated $p(y|x)$ and the label y that most likely will be chosen (Palau, & Moens, 2009).

After the classification, the limits of the arguments need to be defined. To find these limits two approaches can be used. The first one is by using the existing structure of the document. This approach is limited because it assumes that an argument only is present in one section and is not discussed in multiple sections. A second way to find the limits is to understand the semantics of the different arguments. This last approach can be performed by checking the semantic distance between sentences in an argument (Palau, & Moens, 2009).

Another method to do the classification is by the argumentative propositions. Research states that if it is possible to detect the argumentative propositions then it is also possible to classify them. This detection can also be performed by statistical classifiers. First the clauses and sentences need to be classified. The clauses and sentences can be found by a parsing tool. To check if the clauses and sentences contain arguments the maximum entropy classifier can be used. After this a second classifier will be applied. A support vector machine will be used to classify the clauses as a premise or a conclusion (Palau, & Moens, 2009).

When all the arguments are detected the structure of the arguments needs to be defined. This structure will contain the relations between arguments. To find the structures, parsing can be applied to the arguments. To parse a text a grammar will be defined. This grammar states the rules for grouping the text. These rules are capturing grammatical features (Palau, & Moens, 2009).

4.2 Current applications of argument mining

As the research field of argumentation mining is rather new there are not that many applications. There is interest in argument mining so several studies have stated the applications that are possible with argument mining. Some examples can be found in this part.

Within the area of law there is a positive interest in argumentation mining. The decisions made in court can be complex. To fund these decisions a web of arguments is linked to it. The arguments need to be clear to be able to convince the other party that the right decision is made. Consequently, the arguments are highly important (Mochales, & Moens, 2011).

There is also the possibility to compare cases with each other. This comparison can be based on the amount of arguments that are used to construct the decision. Also, the argument that was most effective and complete can be found. For a judge, it can be relevant to know in which cases a specific type of argument is used. When all the different cases are found there can be analysed how this argument is used and when it triggered a positive outcome (Mochales, & Moens, 2011).

Another field that is open for the application of argument mining is science. A scientific text is based on arguments. An example can be found in the biomedical science. A text in this field often contains arguments why an experiment is more suited than a previous experiment. With argumentation mining a researcher can define the most favourable experiment related to the research (Peldszus, & Stede, 2013).

On social media, a lot of content is generated. To be able to analyse this content argument mining can be used. The opinion of the users can be derived. For a specific product, the evaluation can be made if the users appreciate it or are disappointed about it (Peldszus, & Stede, 2013).

The last field that is open for argument mining is politics. Discussions about political issues can be analysed. Because these discussions can contain complex arguments a tool to analyse them can be helpful (Peldszus, & Stede, 2013).

4.3 Challenges for argument mining

Argumentation mining is still a growing research field. In the future, there are still some challenges to tackle. At the moment, there is room for some strong tools to perform argumentation mining. With the presence of good tools the concept will be applied faster in several disciplines (Lippi, & Torroni, 2016).

Analysing user generated content with argument mining can be used for finding customer profiles. Information that can be generated from this analysis can also be used for a market analysis. From user generated content the social behaviour and social interactions can be derived. With argument mining there is the ability to create new insights. In the past there is already tried to use argument mining for this purpose. The reason why it was not completely successful is the lack of a good argument mining tool (Lippi, & Torroni, 2016).

The research field of artificial intelligence is interested in applying the theories of argument mining. Artificial intelligence is about building machines that can reason on the same level as a human. With argument mining it can be possible to create a machine that both uses statistical and logical frameworks. The machine can then generate new knowledge that is gained from unstructured text (Lippi, & Torroni, 2016).

5 Natural Language Toolkit

In the previous parts the theory of natural language processing is explained. The next part will discuss how this theory can be used in practice with the help of the Natural Language Toolkit. Some more information about the tool will be provided and the most important techniques will be highlighted.

5.1 About the tool

Python has functionalities to perform natural language processing tasks but these are not powerful enough to perform a complete natural language processing task. To cover this problem the Natural Language Toolkit (NLTK) is created. NLTK is an open source library that can be used in Python. The developers have chosen for a library in Python because it is a language that has a small learning curve. The coding language that is used in the NLTK library is also easy to understand. This makes the tool accessible for everyone who has knowledge about natural language processing but has no programming experience (Bird, Klein, & Loper, 2009).

For new users, information is available on the website of NLTK. Because of the availability of the information, learning to use the toolkit is not a hard step. As the field of natural language processing will continue to evolve NLTK will be able to implement the new evolutions in their tool (Bird, Klein, & Loper, 2009).

NLTK contains functionalities to explore linguistic data and manipulate the data. With the library, systems can be created to perform the different steps of language processing tasks. The last part of functionalities is checking the performance of the natural language processing techniques (Bird, Klein, & Loper, 2009).

Within the tool there are specific functionalities that can be linked to the levels of natural language processing that are discussed before. Relating to the word levels (phonology, morphology and lexical) the frequencies of words can be found. There are specific functionalities for the phonology level that use predefined dictionaries. To find the morphology of a word words can be tokenized. A token is the grammatical definition of a word. Semantics of a word can be found by using WordNet. WordNet is a semantical dictionary for the English language. The semantics of a word is captured in the lexical level. Also, lexical relations can be found by categorizing, tagging words or the classification of text. For the syntactical level, grammars can be used. A grammar defines the word class of a word in a specific sentence. The semantics of a text can be analysed by using parsing in a text. A discourse parser can be used for analysing the discourse level. In contrast to the other levels there are no specific functions defined for the pragmatic level (Bird, Klein, & Loper, 2009).

As not all functionalities of NLTK can be used to find decisions in a text only the important ones will be discussed. The next part will first describe how the process of information extraction is performed within NLTK. Also, the functionality of classifying text will be discussed in more depth.

5.2 Information extraction

As mentioned before natural language processing has several application areas. The application field that will be used in this part is information extraction. Information extraction from structured data is straight forward. Using one query on a list will give the answer to a question. When the data is unstructured, which is the case for textual data, the presentation in a list is not possible. There are two approaches to get some structure in a text. One approach is to build a general representation of meaning. This approach will be discussed later. In this part the approach is looking for specific information in a text (Bird, Klein, & Loper, 2009).

5.2.1 Information extraction architecture

Figure 3 shows how information extraction systems are build. The input of the system is raw unstructured text. Several steps are needed to have the complete system. The first step is separating the text into sentences. With the tokenizer, each sentence will be divided into words. In the step part-of-speech tagging a tag will be given to each sentence. These tags are used in the next step entity recognition. In entity recognition, each sentence will be analysed for interesting entities. The final step is using the entities created in the previous step and finding relations between them (Bird, Klein, & Loper, 2009).

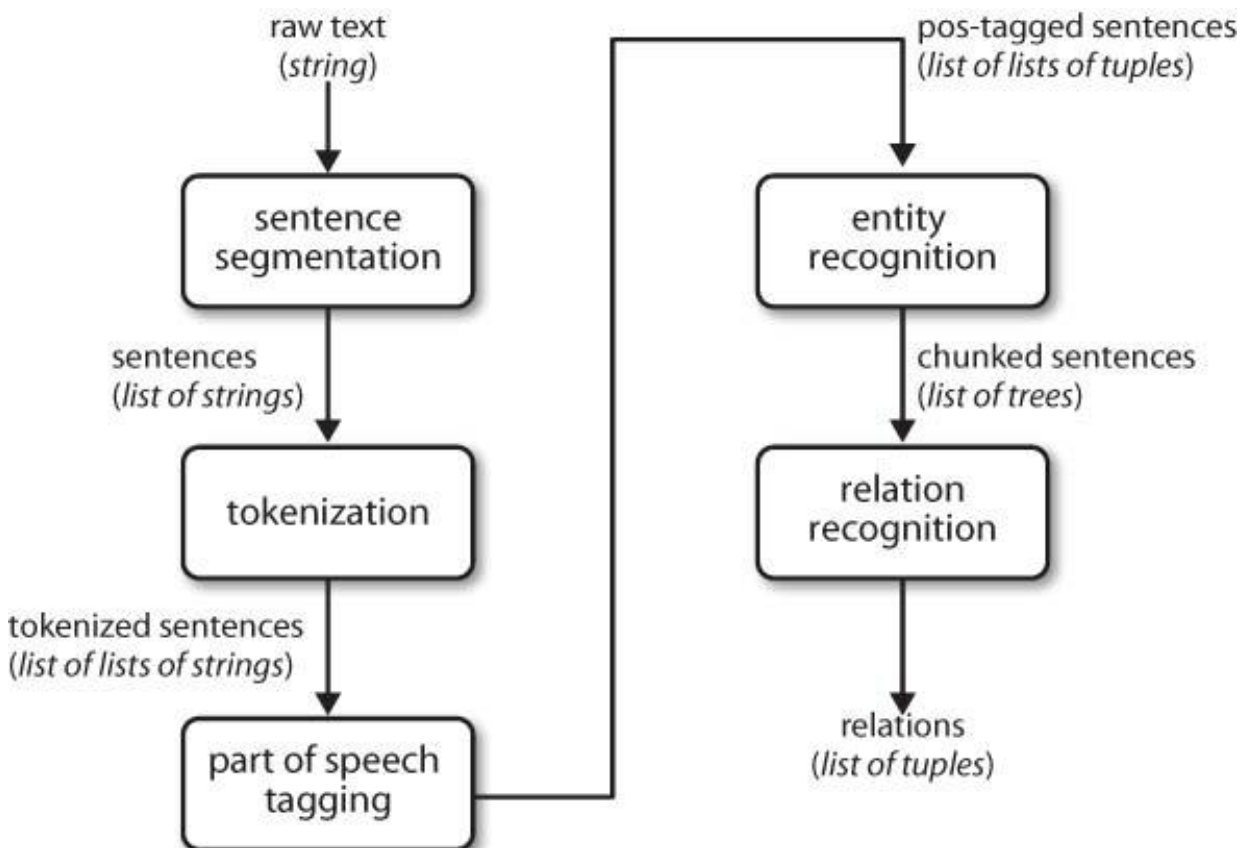


Figure 3: a framework for information extraction systems

The first three tasks can be performed by the code from box 1. The sentence segmentation is covered by the first line, the tokenizer by the second line and the last line covers the part-of-speech tagging (Bird, Klein, & Loper, 2009).

```
def ie_preprocess(document):
    sentences = nltk.sent_tokenize(document)
    sentences = [nltk.word_tokenize(sent) for sent in sentences]
    sentences = [nltk.pos_tag(sent) for sent in sentences]
```

Box 1: NLTK code for segmentation, the tokenizer and part-of-speech tagging

When these functions are executed the entities will be labelled and divided into segments. The last step is to analyse the entities and relations to find specific patterns between entities (Bird, Klein, & Loper, 2009).

5.2.2 Chunking

At this point the text is tokenised and the words are provided of a tag. The next step is to find the entities from the text. The entities of a text can be found by entity recognition. There is a specific technique to perform this step namely chunking. Chunking is divided in different types. The type that will be discussed is noun phrase chunking. Here a noun phrase is related to a chunk (Bird, Klein, & Loper, 2009).

Chunks can be found by performing part-of-speech tagging. Before starting to divide a text in chunks, a chunk grammar needs to be defined. This grammar will decide how sentences will be chunked. A rule will be used in a grammar to state when a part of the text is a chunk. To define the rules a tag pattern is used. The tag pattern will describe the sequence of words that can be grouped (Bird, Klein, & Loper, 2009).

The expectation is that the result of chunking a text with decisions will be the decisions when an appropriate grammar is defined. The grammar must contain a specific pattern for decisions.

There is no predefined tag pattern for a decision. To find the tag pattern of a decision part-of-speech tagging can be performed on a sentence that contains a decision. Because there is no fixed tag pattern for a decision this process will be repeated for five sentences. After the part-of-speech tagging is performed an analysis will be performed on the results to check if there is a returning pattern in the tags. In box 2 this process is illustrated (Bird, Klein, & Loper, 2009).

Sentences:

When a customer buys four items he gets a discount of ten percent.

→ [('When', 'WRB'), ('a', 'DT'), ('customer', 'NN'), ('buys', 'VBZ'), ('four', 'CD'), ('items', 'NNS'), ('he', 'PRP'), ('gets', 'VBZ'), ('a', 'DT'), ('discount', 'NN'), ('of', 'IN'), ('ten', 'JJ'), ('percent', 'NN'), ('.', '.')]]

→ 'WRB', 'DT', 'NN', 'VBZ', 'CD', 'NNS', 'PRP', 'VBZ', 'DT', 'NN', 'IN', 'JJ', 'NN'

If the animal on the farm has two legs it is a chicken.

→ [('If', 'IN'), ('the', 'DT'), ('animal', 'NN'), ('on', 'IN'), ('the', 'DT'), ('farm', 'NN'), ('has', 'VBZ'), ('two', 'CD'), ('legs', 'NNS'), ('it', 'PRP'), ('is', 'VBZ'), ('a', 'DT'), ('chicken', 'NN'), ('.', '.')]]

→ 'IN', 'DT', 'NN', 'IN', 'DT', 'NN', 'VBZ', 'CD', 'NNS', 'PRP', 'VBZ', 'DT', 'NN'

All the red boxes need to be stored in the basement.

→ [('All', 'PDT'), ('the', 'DT'), ('red', 'JJ'), ('boxes', 'NNS'), ('need', 'VBP'), ('to', 'TO'), ('be', 'VB'), ('stored', 'VBN'), ('in', 'IN'), ('the', 'DT'), ('basement', 'NN'), ('.', '.')]]

→ 'PDT', 'DT', 'JJ', 'NNS', 'VBP', 'TO', 'VB', 'VBN', 'IN', 'DT', 'NN'

The yellow pulls will be ordered every Friday.

→ [('The', 'DT'), ('yellow', 'JJ'), ('pulls', 'NNS'), ('will', 'MD'), ('be', 'VB'), ('ordered', 'VBN'), ('every', 'DT'), ('Friday', 'NNP'), ('.', '.')]]

→ 'DT', 'JJ', 'NNS', 'MD', 'VB', 'VBN', 'DT', 'NNP'

Painting a door takes two hours.

→ [('Painting', 'VBG'), ('a', 'DT'), ('door', 'NN'), ('takes', 'VBZ'), ('two', 'CD'), ('hours', 'NNS'), ('.', '.')]]

→ 'VBG', 'DT', 'NN', 'VBZ', 'CD', 'NNS'

Box 2: tags of the five decisions

Looking at the examples in box 2 a pattern can be derived. In the beginning, there is always a determiner (DT) which is followed by a noun (NN, NNS). After the noun a verb (VBZ,VBP,VB,VBN,VBG) will follow. Between these three word classes an undefined number of words can appear. This outcome leads to the following tag pattern: a determiner, a noun and a verb. This patterns can be transferred to a grammar that is used to parse a text. Between the three word classes an undefined number of words can appear. To mark these words a question mark is used in the grammar. The following grammar is created: grammar = "NP: {<DT>?<NN|NNS>?<VBZ|VBP|VB|VBN|VBG>?}". Now this grammar will be applied for chunking a text that contains decisions. This chunker can be tested on a text and the results can be printed. The code that is presenting these actions can be found in box 3 (Bird, Klein, & Loper, 2009).

The text that will be used is an exercise from an exercise book about business process modelling and decision modelling. The text that is chosen is about the decision to permit to rent a car to a person. The original text can be read in appendix 1.

example_text = "The decision logic of each individual decision can be declared as a decision table. The top-level decision has three input expressions: Sufficient funds, Requested amount and Valid Identification each depicted as a column in the table. The first refers to the determined result of the Sufficient funds sub-decision whereas the last refers to the determined result of the Valid Identification sub-decision. In combination with the contained cells, each column frames a condition. Each row in the table portrays a business rule, that applies if all conditions that are specified in the row's cells are met. The conclusions of a business rule is the cell value in the output column Withdrawal status to the right. The rules are: If the customer has sufficient funds and the requested amount is below 15000e the withdrawal is approved. If the customer has sufficient funds and the requested amount is 15000e or more the withdrawal is approved if the customer provides valid identification. If the customer has sufficient funds and the requested amount is 15000e or more but no valid identification can be provided the withdrawal is not approved. If the customer does not have sufficient funds the withdrawal is also not approved"

```
text = nltk.word_tokenize(example_text1)
text1 = nltk.pos_tag(text)
grammar = "NP: {<DT>?<NN|NNS>?<VBZ|VBP|VB|VBN|VBG>?}" #grammar
cp = nltk.RegexpParser(grammar) #chunk parser
result1 = cp.parse(text1) #apply chunk parser
print(result1) #print result
result1.draw() #display tree
```

Box 3: chunking the example tekst

The output from this code can be found in appendix 2. The output does not give a clear overview of the decisions that are present in the text. The chunks that are created are between brackets. The expectation was that the chunk would contain the decision but this is not the case. This means that the grammar is not elaborate enough or that the method is not appropriate for finding decisions.

Before concluding that the method is not suitable for finding decisions another grammar will be used. Looking again at the patterns found in the five decisions from box 2 a second pattern can be found in four of the five decisions. The first pattern appears in the beginning of the sentence and in the second part of the sentences contains a new pattern. This second pattern consists of a verb, a determiner and a noun. The new grammar will be a combination of the first grammar and the second grammar.

```
grammar = "NP: {<DT>?<NN|NNS|NNP>?<VBZ|VBP|VB|VBN|VBG>? <
VBZ|VBP|VB|VBN|VBG>?<DT>?<NN|NNS|NNP>?}"
```

Box 4: the adapted grammar

The code from box 3 will be repeated but the grammar will be replaced by the grammar from box 4. In appendix 3 the output after executing the new code can be found. Looking at the chunks it is still not clear what the available decisions are in the text.

After chunking the text with two different grammars the conclusion can be made that chunking is not providing the desired result. The next step is to check whether another functionality can find decisions in a text.

5.2.3 Chinking

Besides chunking there is also the possibility to chink a text. This will do the opposite of chunking. As mentioned before, chunking will search for the structure that is defined in a grammar and display this in a group. Chinking on the other side will search for the grammar but will display everything besides the grammar in a group. The tokens from the grammar will be removed from the chunk (Bird, Klein, & Loper, 2009).

5.2.4 Named entity recognition

Another method to extract information from a text is named entity recognition. Instead of searching for patterns in a text there will be searched for specific entities.

There are two steps for performing named entity recognition. These are identifying the boundaries of the named entity and declaring the type. An entity can be represented by a feature, for example an organization or a location.

One of the application areas of named entity recognition is question answering. By using named entity recognition, the specific parts of text that contain the answer will be given. If one of the available features would be a decision than there could be searched for decisions in a text. This is not the case so named entity recognition cannot be used to find decisions in a text (Bird, Klein, & Loper, 2009).

5.3 Classifying text

After the conclusion is made that information extraction does not have the right functionalities to find decisions there will be looked at a new functionality of NLTK. This functionality is classifying text. Specific word structures and word frequencies are present in a text and are related to a meaning. Classifying a text will help finding patterns in a text and give the patterns a corresponding class label. One part of classification is finding features that need to be identified to know how the text can be classified. A second part is the construction of models that will be used to automate the tasks for language processing. As a last part, with classifying text there is the ability to learn from the models that are created (Bird, Klein, & Loper, 2009).

Text classification can be used for different tasks. One of the tasks is finding the right features. The classifier will be used to check if a feature is relevant or not and how they can be represented.

In some cases, features are depending on the context of the word. To find these features the context can be exploited by using text classification (Bird, Klein, & Loper, 2009).

There is a possibility that the previous classification tasks are related. To find these relations a sequence classification can be performed. Several inputs will be used so a joint classifier will be created to define the correct label. For the classification task a sequence classifier will be used. This classifier will search the class label for the first input and this class label will be used to find the class label for the next input (Bird, Klein, & Loper, 2009).

With the task of finding decisions in a text there will be a focus on a machine learning technique that can be used with text classification. This technique is creating a decision tree. Instead of classifying a text a document can be given a label using classification (Bird, Klein, & Loper, 2009).

5.3.1 Supervised classification

All the tasks that are described in the previous part need to be performed with a supervised classification. The term supervised classification will be explained in this part.

During a classification task the appropriate class label will be chosen for a text. The input for a classification task is already labelled before the task is performed. When a classifier is supervised, the classification will be performed on a training text that contains the correct labels for every input (Bird, Klein, & Loper, 2009).

5.3.2 Decision trees

An application of classifying text is the use of decision trees. Decision trees are a machine learning method that will automatically create classification models. Within NLTK there exist three machine learning methods: decision trees, naïve Bayes classifiers and maximum entropy classifiers. The following part will discuss the decision trees. The other machine learning methods are not captured in this research (Bird, Klein, & Loper, 2009).

A decision tree is a tree structure that represents the labels for input values. At the decision nodes, a check for feature values will be performed. At the leaves of the tree, the labels will be assigned. The root node is the initial decision that needs to be taken in the decision tree. This root node contains a feature for an input value. Depending on the feature of the input value a branch will be followed till the next node. At every node, this process will be repeated till the leaf node is reached and the input label gets a label. In figure 4 an example of a decision tree can be found. This decision tree checks the gender of a name (Bird, Klein, & Loper, 2009).

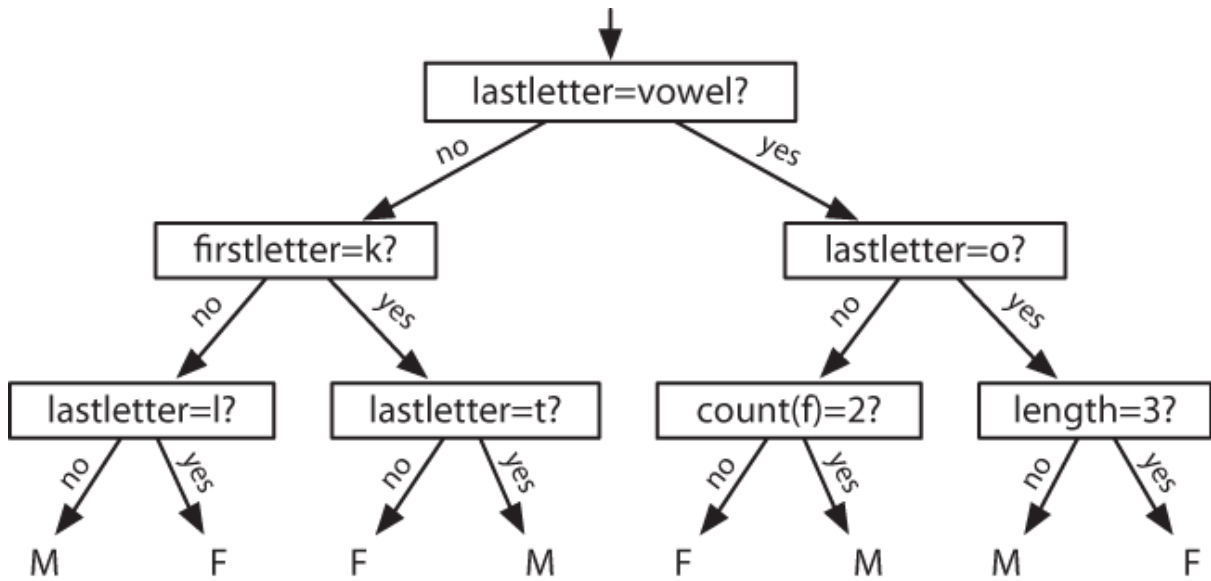


Figure 4: a decision tree to determine the gender of a name

To illustrate the example there will be checked if the name 'Valerie' is masculine or feminine. The last letter is an 'E', this is a vowel. The answer to the first decision is 'YES'. The answer to the second decision is 'No', as the last letter is no 'O'. For the last decision the answer is 'NO' because the length of the word is 7 letters. The result of the decision tree is that the name 'Valerie' is feminine.

With a decision tree the assignment of labels to input values can easily be executed. Another task is choosing a decision stump. When a decision tree only has one node that defines how inputs are classified by using one feature, this decision tree is a decision stump. If the algorithm for a decision stump is known this can be used to create a more elaborated decision tree (Bird, Klein, & Loper, 2009).

5.3.3 Conclusion

The functionalities for classifying text can be used to extract information out of a text. But the method that is used by classification will not be applicable to find decisions in a text. Classification selects words looking at their features. Decisions are not defined as features so this method cannot be applied to finding decisions (Bird, Klein, & Loper, 2009).

6 Stanford parser

As the right functionality to extract a decision is not found within NLTK a new method is used. The Stanford parser is a parsing method that is specialised on finding dependencies in a text.

6.1 About the parser

The Stanford parser is created by the Stanford natural language processing group. The parser will detect relations within the text based on grammatic structures. The relations that will be found are called dependencies. The relations are between a governor and a dependent. Because it is always between two components the relation is called binary. To be able to find the grammatical structure each word needs to be linked to a word class. To define this word class the Stanford parser uses part-of-speech tagging and phrasal labels. The grammatical relations that can be found with the parser have a specific name. In the next part, some examples of these grammatical relations will be given. Apart from these grammatical relations there are also other grammatical relation which are not described (de Marneffe, & Manning, 2008).

The appositional modifier (appos) links the first noun to the noun that defines or modifies that noun. For example, 'Jan is the brother of Jef'. With appos the outcome will be (Jan, brother).

The coordination (cc) represents the relations between an element of a conjunct and the coordinating conjunction word. For example, 'It is either left or right'. The coordination will give the following result (left, or).

Relating to the coordination there is also the conjunct (conj) relation. With a conjunct, the relation between two elements which are connected by a coordinating conjunction can be found. Using the same example as for a coordination this will give another output. 'It is either left or right' will give (left, right).

A dependency (dep) is the default value. If the system cannot find a better dependency it will be marked as dep.

The direct object (dobj) will give the relation between an object and the verb of this object. For example, 'She works for the government.' This example will have the following outcome, (works, government).

6.2 Dependencies

As given in the examples of grammatical relations a dependency will be assigned when no other relation can be found. For this dependency relation, there exist five different representation. There is a basic representation, a collapsed dependency, a collapsed dependency with propagation of conjunct dependencies, a collapsed dependency preserving a tree structure and a non-collapsed dependency (de Marneffe, & Manning, 2008).

The first form is a basic dependency. This is equal to the grammatical relation 'dependency' explained in the first example. In a sentence, every word will be linked to another word or to the root. A root can be a punctuation mark (de Marneffe, & Manning, 2008).

Besides the basic type there are also collapsed dependencies. This will give a combined representation of two dependencies. The dependency can be collapsed when a preposition, conjunction or information about the referent is included. For example, there is a preposition prep(bases, in) and an object of preposition pobj(in, LA). The collapsed representation will be prep_in(based, LA). A second example is a relation with a conjunction. The two relations cc(makes, and) and conj(makes, distributes) becomes conj_and(makes, distributes). Here the 'and' that links the 'makes' and 'distributes' will be included in the relation (de Marneffe, & Manning, 2008).

Collapsed dependencies with propagation of conjunct dependencies are the third variant. This form excludes the dependencies that do not support the tree structure. Relations that do not support this structure are relations between parts of a relative clause and the antecedent linked to this clause. Also, the relation between the object of preposition and the controlling subject relation will be excluded (de Marneffe, & Manning, 2008).

The last variant is non-collapsed dependencies. This variant will display all the dependencies. They will not be collapsed if it is possible. Even the dependencies that do not support the tree structure are included (de Marneffe, & Manning, 2008).

6.3 Executing the Stanford parser

Two different classes can be used to extract the dependencies out of a text by using the Stanford parser. The first class is 'edu.Stanford.naturallanguage.processing.parser.lexparser.LexicalizedParser'. Within this class the specific variants of dependencies can be defined. The specific code can be found in box 5 (de Marneffe, & Manning, 2008).

<p>"basicDependencies" Basic dependencies.</p> <p>"collapsedDependencies" Collapsed dependencies (not necessarily a tree structure).</p> <p>"CCPropagatedDependencies" Collapsed dependencies with propagation of conjunct dependencies (not necessarily a tree structure). [This representation is the default, if no option is specified.]</p> <p>"treeDependencies" Collapsed dependencies that preserve a tree structure.</p> <p>"nonCollapsedDependencies" Non-collapsed dependencies: basic dependencies as well as the extra ones which do not preserve a tree structure.</p> <p>"nonCollapsedDependenciesSeparated" Non-collapsed dependencies where the basic dependencies are separated from the extra ones (by "====").</p>
--

Box 5: variants of dependencies in class 'edu.Stanford.naturallanguage.processing.parser.lexparser.LexicalizedParser'

The second class that can be used is `'edu.Stanford.natural language processing.trees.EnglishGrammaticalStructure'`. This class can be used when the Penn treebank tree structure already is used. The Penn treebank structure is a specific way of using part-of-speech tagging. In this class the dependencies are presented as shown in box 6 (de Marneffe, & Manning, 2008).

```
-basic  
-collapsed  
-CCprocessed  
-collapsedTree  
-nonCollapsed  
-conllx  
-originalDependencies
```

Box 6: variants of dependencies in class 'edu.Stanford.natural language processing.trees.EnglishGrammaticalStructure'

The tool that will be used to execute the Stanford parser is GrammarScope. A detailed overview of this tool will follow in the next part. GrammarScope uses the second class to execute the Stanford parser.

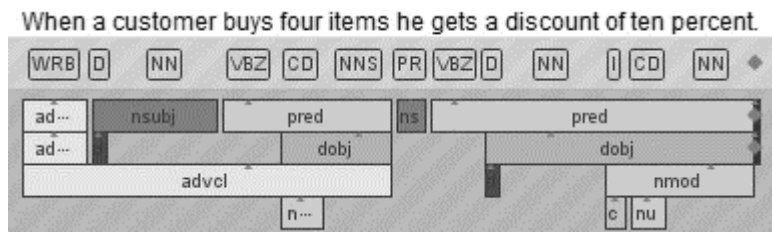
6.4 GrammarScope

The Stanford parser can be used in several ways. The package can be implemented in NLTK or the code can be executed by JavaScript. Another way to work with the Stanford parser is using the tool GrammarScope. This tool uses the Stanford parser to find dependencies in a text (de Marneffe, & Manning, 2008).

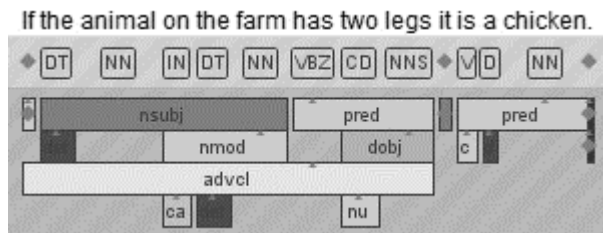
From the previous dependency types, there will be searched for the basic dependencies. The similar way of working as with the chunker of NLTK is applied, the parser will be applied to the five decisions of box 2. In box 7 the results of the analysis can be found.

Sentences:

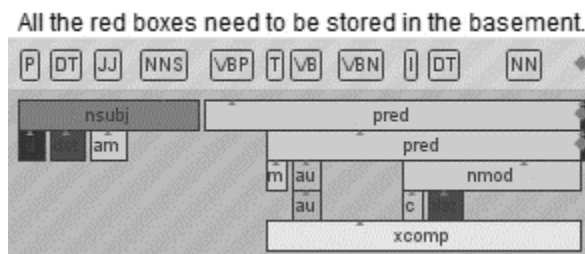
When a customer buys four items he gets a discount of ten percent.



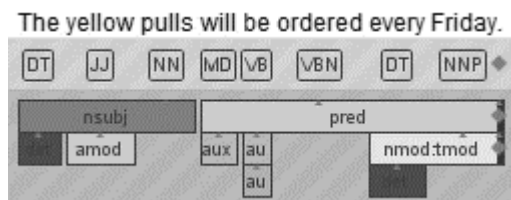
If the animal on the farm has two legs it is a chicken.



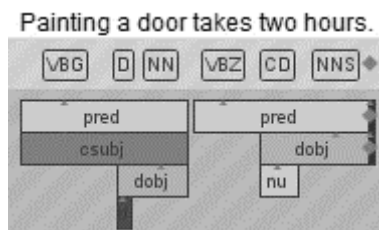
All the red boxes need to be stored in the basement.



The yellow pulls will be ordered every Friday.



Painting a door takes two hours.



Box 7: GrammarScope applied to the five decisions

Looking at the outcomes of the parser a first pattern can be found in the sentences. In four of the five sentences, there first is a nominal subject (nsbj) relation followed by a predicate (pred) relation. In the last sentence there also appears a subject relation followed by a predicate relation but it is a clausal subject relation instead of a nominal subject relation. The definitions as they are stated in the GrammarScope tool are illustrated below (de Marneffe, & Manning, 2008).

The predicate grammatical relation. The predicate of a clause is the main VP of that clause; the predicate of a subject is the predicate of the clause to which the subject belongs.

The nominal subject grammatical relation. A nominal subject is a subject which is a noun phrase.

The clausal subject grammatical relation. A clausal subject is a subject which is a clause (subject is "what she said" in both examples). (de Marneffe, & Manning, 2008, p5)

After finding this pattern the parser will be applied to the text in appendix 3. The text that will be used is about the decision to rent a car or not. This text contains sentences with decisions and without decisions. Looking at the result the conclusion can be made that the pattern which was found above is applicable to most of the sentences. The distinction between a sentence with a decision and without a decision cannot be made.

As the previous pattern is not sufficient to find decisions in a text there will be searched for a new pattern. At this point, there will also be checked if some relations appear in each of the five decisions. Looking at the first two examples they both have an adverbial clause modifier (advcl) relation. The definition can be found below (de Marneffe, & Manning, 2008).

The adverbial clause modifier grammatical relation. An adverbial clause modifier is a clause which modifies a verb or other predicate (adjective, etc.), as a modifier not as a core complement. This includes things such as a temporal clause, consequence, conditional clause, purpose clause, etc. The dependent must be clausal (or else it is an `advmod`) and the dependent is the main predicate of the clause. (de Marneffe, & Manning, 2008, p4)

A second relation that is available in the decisions is the open clausal complement (xcomp) relation. This relation is only found in the third example but looking at the outcome when the previous pattern is applied the open clausal complement appears only when the sentence includes a decision. The definition of an open clausal relation can be found in the next part (de Marneffe, & Manning, 2008).

An open clausal complement (*xcomp*) of a VP or an ADJP is a clausal complement without its own subject, whose reference is determined by an external subject. These complements are always non-finite. The name *xcomp* is borrowed from Lexical-Functional Grammar. (Mainly "TO-clause" are recognized, but also some VBG like "stop eating") (de Marneffe, & Manning, 2008, p10)

After selecting these two relations the complete text will be analysed again. When the sentences that have an advcl or xcomp relation are filtered from the text the following result is found. A disadvantage of the tool is the filtering option that is missing. The sentences with the wanted relations need to be extracted manually.

In appendix 4 the sentences of the text with an advcl or xcomp relation can be found. Looking at these sentences the conclusion can be made that the filtered sentences only contain decisions. The original text counted 583 words and the filtered text 451 so the filtering made the text shorter.

If it is possible to find decisions by searching for these relations a check is needed to confirm that all the decisions are found. So, next a decision model will be made from the generated decisions. This decision model will then be compared with a decision model that is generated from the original text. When the two models will be compared, some factors will be evaluated.

In figure 5 the decision model and the linked decision tables made from the original text can be found in table 3, 4 and 5.

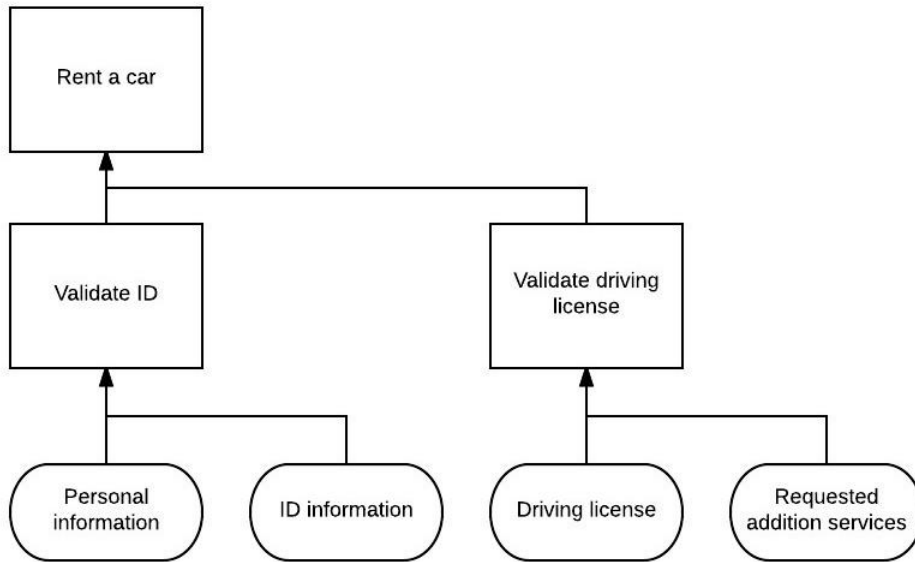


Figure 5: the decision model to rent a car

Rent a car (uniform hit policy)		
Input: Valid ID	Input: Driving license	Output: Document status
Valid	Valid	Valid
Valid	Not valid	Not valid
Not valid	Not valid	Not Valid
Not valid	Valid	Not valid

Table 3: the decision logic of the main decision rent a car

Validate ID (uniform hit policy)		
Input: Personal information	Input: ID information	Output: Valid ID
Valid	Valid	Valid
Valid	Not valid	Not valid
Not valid	Not valid	Not Valid
Not valid	Valid	Not valid

Table 4: the decision logic of the sub decision validate ID

Validate driving license (first hit policy)			
Input: Driving license information	Input: Request for chauffeur	Input: Availability chauffeur	Output: Driving licence
Valid	Yes	Free	Valid
Valid	Yes	Occupied	Valid
Valid	No	/	Valid
Not valid	Yes	Free	Valid
Not valid	Yes	Occupied	Not valid
Not valid	No	/	Not valid

Table 5: the decision logic of the sub decision validate driving license

The next step is to create the decision model and decision logic with the rules that are extracted by using GrammarScope (appendix 4). The decision model that is created is the same as the one that is created from the original text. As the same result was the same this decision model and decision tables are not shown. For this example, the decision model is first created by reading the text and searching for the decisions. Afterwards the tool GrammarScope is used. Because when using GrammarScope there was already knowledge about the decision there could be some bias.

For the next example, there first will be started with using the tool GrammerScope. In this case, there is no prior knowledge about the decisions and the process. The decision model from figure 6 is constructed together with the decision tables from table 6 and 7. The complete text can be found in appendix 5 and the outcome of GrammarScope in appendix 6.

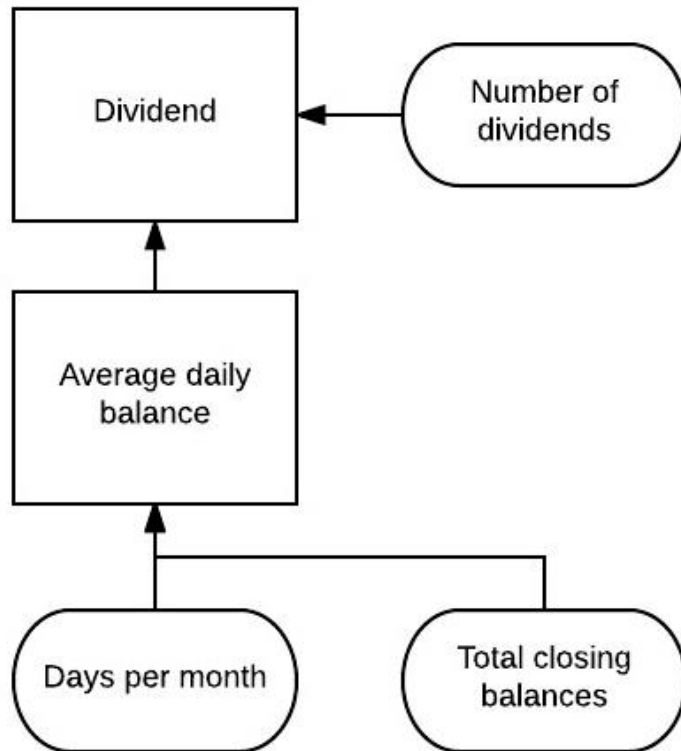


Figure 6: the decision model to determine a dividend created with GrammarScope

Dividend (first hit policy)		
Input: Average daily balance	Input: Number of dividends	Output: Dividend
<25	/	0
>= 25	<=500	6%
>= 25	{501,2000}	6,5%
>= 25	>= 2001	7%

Table 6: the decision logic for the main decision dividend created with GrammarScope

Average daily balance (first hit policy)		
Input: Days per month	Input: Total closing balances	Output: Average daily balance
/	/	= Total closing balances / Days per month

Table 7: the decision logic for the sub decision average daily balance created with GrammarScope

From the decision model can be derived that the process is about determining the amount of dividend. The sub decision that needs to be made is about the average daily balances. This average will be calculated in the sub decision. The right solution is not available so at this point no conclusion can be made about the correctness of the model. So, the next step is to check whether all the decisions are captured in this decision model. A new decision model will be created by reading the original text. The results can be found in figure 7 and tables 8, 9, 10 and 11.

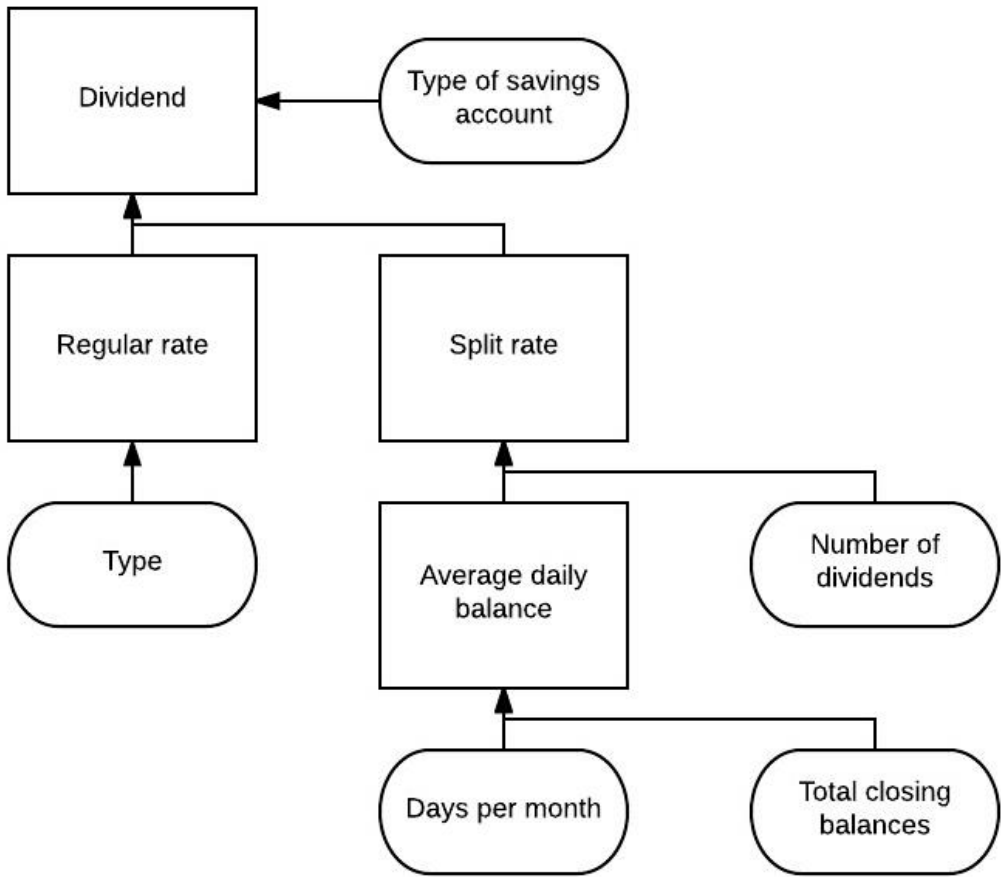


Figure 7: the decision model to determine a dividend created without GrammarScope

Dividend (first hit policy)			
Input: Type of savings account	Input: Regular rate	Input: Split rate	Output: Dividend
Regular rate	/	/	= Regular rate
Split rate	/	/	= Split rate

Table 8: the decision logic for the main decision dividend created without GrammarScope

Regular rate (uniform hit policy)	
Input: Type	Output: Split rate
Insured	5,75%
Uninsured	6%

Table 9: the decision logic for the sub decision regular rate created without GrammarScope

Split rate (first hit policy)		
Input: Average daily balance	Input: Number of dividends	Output: Dividend
<25	/	0
>= 25	<=500	6%
>= 25	{501,2000}	6,5%
>= 25	>= 2001	7%

Table 10: the decision logic for the sub decision split rate created without GrammarScope

Average daily balance (first hit policy)		
Input: Days per month	Input: Total closing balances	Output: Average daily balance
/	/	= Total closing balances / Days per month

Table 11: the decision logic for the sub decision average daily balance created without GrammarScope

The model that is created after reading the original file is more elaborate than the result when using the tool GrammarScope. The information about the regular rate was not included in the outcome of GrammarScope. So, in contrast to the first example the use of GrammarScope does not ensure that all the decisions are covered.

7 Conclusion

Decision modelling can add value to the daily processes of a company. It is important that a task which adds value happens in the most efficient way. When it happens efficiently it will add the most value. The goal of the thesis is to find a way that decisions can be found automatically in a text. This automation will make the task of finding decisions more efficient and, related to that task the complete decision modelling.

A structured way to make decisions is decision modelling. To construct a decision model several steps need to be completed. First the decision knowledge needs to be gathered and afterwards the decision model can be constructed with the decision logic. Only the step of finding the decision knowledge will be supported by the new method. All the other steps will remain manual steps.

The decision knowledge is often captured in text files. To analyse a text natural language processing can be used. With the analysing methods of natural language processing information can be extracted from a text.

One of the fields where natural language processing is used is argument mining. Argument mining will search for arguments and detections. Argument mining is linked to the semantic level and the discourse level of natural language processing.

NLTK makes the practical application possible. The theories of natural language processing are captured as different functionalities of the tool. Decisions can be compared to relations in a text. To find these relations or dependencies the text can be chunked or classified. Both functionalities are not elaborate enough to find decisions. With chunking, it is possible to find specific structures in a text. The grammar that is used to find structures in a text was not specific enough to find decisions. Classifying a text is more focused on finding predefined features in a text. In NLTK a decision is not defined as a feature. So with classifying text, decisions cannot be extracted.

The Stanford parser is designed to find dependencies in a text. With this parser, grammatical relations could be found in a text. Some specific relations can extract decisions but in some cases not all the decisions are captured.

Working with both NLTK and GrammarScope is not that time consuming. For NLTK the parameters that need to be implemented are the text that needs to be analysed and the grammar. As the grammar will stay fixed only the text needs to be changed for different tasks. GrammarScope is a more commercialised tool and straight forward to use. After implementing the text and choosing the parsing type, relations can be found directly. NLTK works with coding so this needs some more background knowledge.

The exercises that were made during this research were never longer than one page. Comparing the time that is needed to read the text and the time that is used to find the rules the conclusion can be made that performing the analysis is faster with GrammarScope than without GrammarScope. When reading the whole text, more attention will be paid to details that are not necessary. GrammarScope will only give the text that is needed. Because during this research there is only worked with simple texts that were not longer than a page the time difference is

rather small. When GrammarScope will be used on longer pages the expectation is that the time difference will be bigger.

7.1 Further research

This research explored if natural language processing can be used to find decisions. The conclusions that are made are based on the results found during the research. As there were some limitations and some assumptions are made there is still room for further research.

During the research it turned out that there is no clear definition for a decision. The structures that were used for finding decisions were derived from self-defined examples. A collaboration with a language expert can give more clarity for the term.

When a uniform definition can be created for a decision then it probably will become easier to search for a decision. A more specific grammar can be defined or there can be searched for a specific relation. For classification a feature can be created with the help of the definition.

The compliance can be compared of decision models that are created with the use of natural language processing. Will decision models created from the same text but by different persons be the same? Are the decision requirements better captured?

References

- Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python* (1st ed.). Beijing: O'Reilly.
- Chowdhury, G. (2005). Natural language processing. *Annual Review Of Information Science And Technology*, 37(1), 51-89. <http://dx.doi.org/10.1002/aris.1440370103>
- Dagan, I., & Feldman, R. (1995). Knowledge discovery in textual databases (KDT). *KDD*, 95, 112-117.
- de Marneffe, M., & Manning, C. (2008). *Stanford typed dependencies manual* (1st ed.). Retrieved from https://nlp.stanford.edu/software/dependencies_manual.pdf
- Decision management solutions. (2016).
- Decision model and notation*. (2014) (1st ed.). Needham, USA. Retrieved from <http://www.omg.org/spec/DMN/1.0/Beta1/PDF/>
- Fish, A., & Taylor, J. (2012). *Knowledge Automation: How to Implement Decision Management in Business Proc* (1st ed.). John Wiley & Sons.
- Gailly, F., & Geerts, G. (2013). Ontology-Driven Business Rule Specification. *Journal Of Information Systems*, 27(1), 79-104. <http://dx.doi.org/10.2308/isys-50428>
- Homet, L., & Chakrabarti, S. (2003). *Mining the web: discovering knowledge from hypertext data*. United States of America: Morgan Kaufmann.
- Leonard-Barton, D. (1993). Core capabilities and core rigidities: A paradox in managing new product development. *Long Range Planning*, 26(1), 154. [http://dx.doi.org/10.1016/0024-6301\(93\)90313-5](http://dx.doi.org/10.1016/0024-6301(93)90313-5)
- Liddy, E. (2005). Enhanced Text Retrieval Using Natural Language Processing. *Bulletin Of The American Society For Information Science And Technology*, 24(4), 14-16. <http://dx.doi.org/10.1002/bult.91>
- Lippi, M., & Torroni, P. (2016). Argumentation Mining. *ACM Transactions On Internet Technology*, 16(2), 1-25. <http://dx.doi.org/10.1145/2850417>
- Mochales, R., & Moens, M. (2011). Argumentation mining. *Artificial Intelligence And Law*, 19(1), 1-22. <http://dx.doi.org/10.1007/s10506-010-9104-x>

- Palau, R., & Moens, M. (2009). Argumentation mining. *Proceedings Of The 12Th International Conference On Artificial Intelligence And Law - ICAIL '09*.
<http://dx.doi.org/10.1145/1568234.1568246>
- Peldszus, A., & Stede, M. (2013). From Argument Diagrams to Argumentation Mining in Texts:.
International Journal Of Cognitive Informatics And Natural Intelligence, 7(1), 1-31.
<http://dx.doi.org/10.4018/jcini.2013010101>
- Saint-Dizier, P., & Garcia Villalba, M. (2012). In *Some facets of argument mining for opinion analysis* (pp. 23-35). Toulouse France: IOS Press.
- Solka, J. (2008). Text data mining: Theory and methods. *Statistics Surveys*, 2(0), 94-112.
<http://dx.doi.org/10.1214/07-ss016>
- Stöhr, J. (2016). *From identification o executable logic of operational decision methodological approach with DMN*. Signavio GmbH. Retrieved from
- Tan, A. (1999). Natural language processing: the state of the art and the challenges. *Knowledge Discovery From Advanced Databases*, 8, 65-70. Retrieved from
http://www3.ntu.edu.sg/home/asahtan/papers/tm_pakdd99.pdf
- Taylor, J., Fish, A., Vanthienen, J., & Vincent, P. (2013). Emerging standards in decision modeling. *BPM And Workflow Handbook Series*, 133-146.
- Taylor, J., Fish, A., Vanthienen, J., & Vincent, P. (2013). Emerging standards in decision modeling. In N. Palmer & K. Swenson, *Intelligent BPM systems (BPM and workflow series)* (1st ed., pp. 133-146). IBPMS.
- Torrioni, P., & Lippi, M. (2015). Argument Mining: a Machine Learning Perspective. *Theory And Application Of Formal Argumentation*, 163-176.
- von Halle, B., & Goldberg, L. (2010). *The decision model: a business logic framework linking business and technology* (1st ed.). Boca Raton: Taylor & Francis Group.

List of figures

Figure 1: the dimensions of decision management..... 6
Figure 2: a decision model about eligibility..... 9
Figure 3: a framework for information extraction systems22
Figure 4: a decision tree to determine the gender of a name.....28
Figure 5: the decision model to rent a car.....34
Figure 6: the decision model to determine a dividend created with GrammarScope.....36
Figure 7: the decision model to determine a dividend created without GrammarScope37

List of tables

Table 1: the elements of a decision model 8

Table 2: different relations in argument mining 17

Table 3: the decision logic of the main decision rent a car 34

Table 4: the decision logic of the sub decision validate ID 34

Table 5: the decision logic of the sub decision validate driving license 35

Table 6: the decision logic for the main decision dividend created with GrammarScope 36

Table 7: the decision logic for the sub decision average daily balance created with GrammarScope
..... 36

Table 8: the decision logic for the main decision dividend created without GrammarScope 37

Table 9: the decision logic for the sub decision regular rate created without GrammarScope 38

Table 10: the decision logic for the sub decision split rate created without GrammarScope 38

Table 11: the decision logic for the sub decision average daily balance created without
GrammarScope 38

List of boxes

Box 1: NLTK code for segmentation, the tokenizer and part-of-speech tagging.....	23
Box 2: tags of the five decisions	24
Box 3: chunking the example tekst	25
Box 4: the adapted grammar	25
Box 5: variants of dependencies in class 'edu.Stanford.natural language processing.parser.lexparser.LexicalizedParser'	30
Box 6: variants of dependencies in class 'edu.Stanford.natural language processing.trees.EnglishGrammaticalStructure'	31
Box 7: GrammarSope applied to the five decisions.....	32

Appendices

Appendix 1

A customer who wants to rent a car is obligated to have valid *ID* and *Driving license* (in case the services of a personal chauffeur are not additionally requested and available) in order to use the services of the company. This decision is based on two company policies:

- Age requirements policy (Cars are not provided to people under the age allowed in the country)
- Driving license policy (Customers have to have valid driving license and car driving category)

The inputs for this decision are ID status and Driving License status. The output is the Documents status. If the status of both ID and the Driving license (as considerations) is Valid, then only is the Documents status (as a conclusion) also considered Valid. If the status of any of the two inputs is Not Valid, then the Documents status is also listed as Not Valid.

The validation is executed every time when a customer requests to rent a car. It is the Branch Manager who has to make sure that the decision is made. If a regulation related to the validation of documents is changed (e.g. minimum age required for an individual to rent a car), then the decision must be updated.

KPIs for this decision are the *validation cycle time* and the *core system speed of 'connection to external sources'* (as the information for the documents is provided by external institutional sources).

Thus, the final decision is based on two sub decisions:

Validate ID

The personal information of each customer who requests a car should be validated (name, PIN, date of birth, address) as well as the validity of his ID card (ID card number, expiry date). There are no exceptions from this rule. A policy related to this sub decision is the Age requirements policy stated above.

If the input ID information is Valid (i.e. the information from the official external information source used for validation – *Civil Registry* – corresponds to the documents of the customer), then the ID status (output for the decision) is also Valid. If the information stated in the ID of the client is different than the one returned by the Official Institution database, then the ID status is set to Not Valid.

There are KPI defined to check the quality of this decision: *cycle time for ID validation* and *Core system speed of 'connection to external sources'*.

Validate Driving license

When a customer requests a car, his input driving license information (Name, PIN, Driving License number, Driving category, Expiry date) must be validated. This data comes from an Official Institution external database which varies through the different countries (Federal Mobility and Transport Administration / Traffic Police Register / Government transport Department). If this information corresponds to the one in the document of the customer, then the output Driving license status is Valid. If not, the status is Not Valid.

There is an exception of the rule in case the customer has requested additional services of a chauffeur by the company. Then two considerations are checked: the Driving license information and Chauffeur request and availability.

If the information in the license is Valid, then it does not matter if the customer has requested a chauffeur. The Driving license status is Valid.

If the Driving license information is Not Valid but a Chauffeur is requested and available, then the Driving license status is valid again.

Finally, when a license data is not validated and chauffeur is not requested, the status of the license is set to Not Valid.

Appendix 2

Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 07:18:10) [MSC v.1900 32 bit (Intel)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

RESTART: C:\Users\Valerie\Google Drive\Handelingenieur\masterproef\NLTK\test 1.py

(S

The/DT

decision/NN

logic/NN

of/IN

each/DT

individual/JJ

decision/NN

can/MD

(NP be/VB)

(NP declared/VBN)

as/IN

a/DT

decision/NN

table/NN

./.

The/DT

top-level/JJ

(NP decision/NN has/VBZ)

three/CD

input/NN

expressions/NNS

./:

Sufficient/JJ

funds/NNS

./,

Requested/NNP

amount/NN

and/CC

Valid/NNP

Identification/NNP

each/DT

depicted/VBD

as/IN

a/DT

column/NN

in/IN
the/DT
table/NN
./.
The/DT
first/JJ
refers/NNS
to/TO
the/DT
determined/JJ
result/NN
of/IN
the/DT
Sufficient/JJ
funds/NNS
sub-decision/JJ
whereas/IN
the/DT
last/JJ
refers/NNS
to/TO
the/DT
determined/JJ
result/NN
of/IN
the/DT
Valid/NNP
Identification/NNP
sub-decision/NN
./.
In/IN
combination/NN
with/IN
the/DT
contained/JJ
cells/NNS
./,
(NP each/DT column/NN frames/VBZ)
a/DT
condition/NN
./.
Each/DT
row/NN

in/IN
(NP the/DT table/NN portrays/VBZ)
a/DT
business/NN
rule/NN
./,
that/IN
applies/NNS
if/IN
(NP all/DT conditions that/NNS are/VBP)
(NP specified/VBN)
in/IN
the/DT
row's/NN
(NP cells/NNS are/VBP)
(NP met/VBN)
./.
The/DT
conclusions/NNS
of/IN
a/DT
business/NN
(NP rule/NN is/VBZ)
the/DT
cell/NN
value/NN
in/IN
the/DT
output/NN
column/NN
Withdrawal/NNP
status/NN
to/TO
the/DT
right/NN
./.
(NP The/DT rules/NNS are/VBP)
:/:
If/IN
(NP the/DT customer/NN has/VBZ)
sufficient/JJ
funds/NNS
and/CC

the/DT
requested/JJ
(NP amount/NN is/VBZ)
below/IN
15000e/CD
(NP the/DT withdrawal/NN is/VBZ)
approved.If/IN
(NP the/DT customer/NN has/VBZ)
sufficient/JJ
funds/NNS
and/CC
the/DT
requested/JJ
(NP amount/NN is/VBZ)
15000e/CD
or/CC
more/JJR
(NP the/DT withdrawal/NN is/VBZ)
(NP approved/VBN)
if/IN
(NP the/DT customer/NN provides/VBZ)
valid/JJ
identification.If/NN
(NP the/DT customer/NN has/VBZ)
sufficient/JJ
funds/NNS
and/CC
the/DT
requested/JJ
(NP amount/NN is/VBZ)
15000e/CD
or/CC
more/JJR
but/CC
no/DT
valid/JJ
identification/NN
can/MD
(NP be/VB)
(NP provided/VBN)
(NP the/DT withdrawal/NN is/VBZ)
not/RB
(NP approved.If/VB)

(NP the/DT customer/NN does/VBZ)
not/RB
(NP have/VB)
sufficient/JJ
funds/NNS
(NP the/DT withdrawal/NN is/VBZ)
also/RB
not/RB
(NP approved/VBN))

Appendix 3

RESTART: C:/Users/Valerie/Google Drive/Handelingenieur/masterproef/NLTK/grammar 2.py

(S

(NP The/DT decision/NN logic/NN)

of/IN

(NP each/DT)

individual/JJ

(NP decision/NN)

can/MD

(NP be/VB declared/VBN)

as/IN

(NP a/DT decision/NN table/NN)

./.

(NP The/DT)

top-level/JJ

(NP decision/NN has/VBZ)

three/CD

(NP input/NN expressions/NNS)

:/:

Sufficient/JJ

(NP funds/NNS)

./,

(NP Requested/NNP amount/NN)

and/CC

(NP Valid/NNP Identification/NNP)

(NP each/DT)

depicted/VBD

as/IN

(NP a/DT column/NN)

in/IN

(NP the/DT table/NN)

./.

(NP The/DT)

first/JJ

(NP refers/NNS)

to/TO

(NP the/DT)

determined/JJ

(NP result/NN)

of/IN

(NP the/DT)

Sufficient/JJ

(NP funds/NNS)
sub-decision/JJ
whereas/IN
(NP the/DT)
last/JJ
(NP refers/NNS)
to/TO
(NP the/DT)
determined/JJ
(NP result/NN)
of/IN
(NP the/DT Valid/NNP Identification/NNP)
(NP sub-decision/NN)
./.
In/IN
(NP combination/NN)
with/IN
(NP the/DT)
contained/JJ
(NP cells/NNS)
./,
(NP each/DT column/NN frames/VBZ a/DT condition/NN)
./.
(NP Each/DT row/NN)
in/IN
(NP the/DT table/NN portrays/VBZ a/DT business/NN)
(NP rule/NN)
./,
that/IN
(NP applies/NNS)
if/IN
(NP all/DT conditions that/NNS are/VBP specified/VBN)
in/IN
(NP the/DT row's/NN cells/NNS)
(NP are/VBP met/VBN)
./.
(NP The/DT conclusions/NNS)
of/IN
(NP a/DT business/NN rule/NN)
(NP is/VBZ the/DT cell/NN)
(NP value/NN)
in/IN
(NP the/DT output/NN column/NN)

(NP Withdrawal/NNP status/NN)
to/TO
(NP the/DT right/NN)
./.
(NP The/DT rules/NNS are/VBP)
:/:
If/IN
(NP the/DT customer/NN has/VBZ)
sufficient/JJ
(NP funds/NNS)
and/CC
(NP the/DT)
requested/JJ
(NP amount/NN is/VBZ)
below/IN
15000e/CD
(NP the/DT withdrawal/NN is/VBZ)
approved.If/IN
(NP the/DT customer/NN has/VBZ)
sufficient/JJ
(NP funds/NNS)
and/CC
(NP the/DT)
requested/JJ
(NP amount/NN is/VBZ)
15000e/CD
or/CC
more/JJR
(NP the/DT withdrawal/NN is/VBZ approved/VBN)
if/IN
(NP the/DT customer/NN provides/VBZ)
valid/JJ
(NP identification.If/NN the/DT customer/NN)
(NP has/VBZ)
sufficient/JJ
(NP funds/NNS)
and/CC
(NP the/DT)
requested/JJ
(NP amount/NN is/VBZ)
15000e/CD
or/CC
more/JJR

but/CC
(NP no/DT)
valid/JJ
(NP identification/NN)
can/MD
(NP be/VB provided/VBN the/DT withdrawal/NN)
(NP is/VBZ)
not/RB
(NP approved.If/VB the/DT customer/NN)
(NP does/VBZ)
not/RB
(NP have/VB)
sufficient/JJ
(NP funds/NNS the/DT withdrawal/NN)
(NP is/VBZ)
also/RB
not/RB
(NP approved/VBN))
>>>

Appendix 4

- A customer who wants to rent a car is obligated to have valid ID and Driving license (in case the services of a personal chauffeur are not additionally requested and available) in order to use the services of the company
- Driving license policy (Customers have to have valid driving license and car driving category)
- If the status of both ID and the Driving license (as considerations) is Valid, then only is the Documents status (as a conclusion) also considered Valid.
- If the status of any of the two inputs is Not Valid, then the Documents status is also listed as Not Valid.
- The validation is executed every time when a customer requests to rent a car.
- It is the Branch Manager who has to make sure that the decision is made.
- If a regulation related to the validation of documents is changed (e.g. minimum age required for an individual to rent a car), then the decision must be updated.
- Thus, the final decision is based on two sub decisions:
 - The personal information of each customer who requests a car should be validated (name, PIN, date of birth, address) as well as the validity of his ID card (ID card number, expiry date).
 - If the input ID information is Valid (i.e. the information from the official external information source used for validation – Civil Registry – corresponds to the documents of the customer), then the ID status (output for the decision) is also valid.
 - If the information stated in the ID of the client is different than the one returned by the Official Institution database, then the ID status is set to Not Valid.
- There are KPI defined to check the quality of this decision: cycle time for ID validation and Core system speed of 'connection to external sources'.
- When a customer requests a car, his input driving license information (Name, PIN, Driving License number, Driving category, Expiry date) must be validated
 - If this information corresponds to the one in the document of the customer, then the output Driving license status is Valid
 - If not, the status is Not Valid.
- Then two considerations are checked: the Driving license information and Chauffeur request and availability.
 - If the information in the license is Valid, then it does not matter if the customer has requested a chauffeur.
- The Driving license status is Valid.

- If the Driving license information is Not Valid but a Chauffeur is requested and available, then the Driving license status is valid again. Finally, when a license data is not validated and chauffeur is not requested, the status of the license is set to Not Valid.

Appendix 5

A credit union offers two types of savings accounts: regular rate and split rate. The regular rate account pays dividends on the account balance at the end of each quarter – funds withdrawn during the quarter earn no dividends. There is no minimum balance on the regular rate account. Regular rate accounts may be insured. Insured accounts pay 5.75 percent annual interest. Uninsured regular rate accounts pay 6.00 percent annual interest.

For split rate accounts, dividends are paid monthly on the average daily balance for that month. Daily balances go up and down according to deposits and withdrawals. The average daily balance is determined by adding each day's closing balance and dividing this sum by the number of days in the month. If the average daily balance is less than 25, then no dividend is paid. Otherwise, if the average daily balance is 25 or more, 6 percent per annum is paid on the first 500, 6.5 percent on the next 1,500 and 7 percent on funds over 2000. There is no insurance on split rate accounts.

Appendix 6

For split rate accounts, dividends are paid monthly on the average daily balance or that month. Daily balances go up and down according to deposits and withdrawals. The average daily balance is determined by adding each day's closing balance and dividing this sum by the number of days in the month. If the average daily balance is less than 25, then no dividend is paid. Otherwise, if the average daily balance is 25 or more, 6 percent per annum is paid on the first 500, 6.5 percent on the next 1,500 and 7 percent on funds over 2000.

Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:

Decision modeling : from text to model

Richting: **master in de toegepaste economische wetenschappen:
handelsingenieur in de beleidsinformatica**

Jaar: **2017**

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Voor akkoord,

Kerkhofs, Valerie

Datum: **1/06/2017**