Technische Universität Berlin

Institute of Software Engineering
and Theoretical Computer Science

Part-of-Speech Tagging with Neural Networks for a Conversational Agent

Master Thesis

Master of Science (M.Sc.)

Author	Andreas Müller
Major	Computer Engineering
Matriculation No.	333471
Date	18th May 2018
1st supervisor	Prof. Dr.-Ing. Sebastian Möller
2nd supervisor	Dr. Axel Küpper

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Ausführungen, die anderen veröffentlichten oder nicht veröffentlichten Schriften wörtlich oder sinngemäß entnommen wurden, habe ich kenntlich gemacht.

Die Arbeit hat in gleicher oder ähnlicher Fassung noch keiner anderen Prüfungsbehörde vorgelegen.

Berlin, den May 4, 2018

Unterschrift

Abstract

A part-of-speech tagger is a system which automatically assigns the part of speech to words using contextual information. Potential applications for part-of-speech taggers exist in many areas of computational linguistics including speech recognition, speech synthesis, machine translation or information retrieval in general.

The part-of-speech tagging task of natural language processing is also used in the advisory artificial conversational agent called ALEX. ALEX was developed to answer questions about modules and courses at the Technische Universität Berlin. The system takes the written natural language requests from the user and tries to transform them into SQL-queries. To understand the natural language queries, the system uses a Hidden Markov Model (HMM) to assign tags to each word of the query (part-of-speech tagging). This HMM tagger is trained with manually created training templates that are filled with the data in the database to be queried. The manually created sentence-templates and the slot-filling resulted in many training data sentences with the same structure. This often led to wrong tagging results when the HMM tagger was presented with an input sentence, having a structure that doesn't occur in the training templates.

This thesis shows two different neural network approaches for the language modeling of the input sentences and evaluates and compares both neural network based tagger as well as the HMM based tagger.

Zusammenfassung

Ein Part-of-speech Tagger ist ein System, welches Wortarten anhand von Kontextinformationen automatisch den gegebenen Wörtern zuordnet. Potentielle Anwendungen solcher Tagger gibt es in vielen Bereichen der Computerlinguistik wie Spracherkennung, Sprachsynthese, maschinelle Übersetzung oder Information Retrieval im Allgemeinen.

Part-of-speech Tagging wird auch in ALEX verwendet, einem Artificial Conversational Agent. ALEX wurde entwickelt, um Fragen zu Modulen und Lehrveranstaltungen an der Technischen Universität Berlin zu beantworten. Das System nimmt die in natürlicher Sprache geschriebenen Anfragen des Benutzers und versucht diese in SQL-Abfragen umzuwandeln. Um die natürliche Sprache zu verstehen, verwendet das System ein Hidden-Markov-Model (HMM), um jedem Wort der Eingabe Wortarten zuzuweisen (Part-of-speech Tagging). Dieser HMM-Tagger wird mit manuell erstellten Trainingsvorlagen trainiert, die mit den Daten der abzufragenden Datenbank gefüllt werden. Die manuell erstellten Satzvorlagen führten zu vielen Trainingsdatensätzen mit gleicher Struktur und damit oft zu falschen Tagging-Ergebnissen, wenn der HMM-Tagger einen Eingabesatz mit einer Struktur verarbeiten sollte, die in den Trainingsvorlagen nicht vorkommt.

Diese Arbeit zeigt zwei verschiedene Ansätze für die Sprachmodellierung der Eingabesätze basierend auf neuronalen Netzwerken und bewertet und vergleicht sowohl die Neuronalen Netzwerk-basierten Tagger als auch den HMM-basierten Tagger.

Contents

List of Figures

List of Tables

Abbreviations

ACA	*Artificial Conversational Agent*
ANN	*Artificial Neural Network*
FNN	*Feed-forward Neural Network*
HMM	*Hidden Markov Model*
NLP	*Natural Language Processing*
NLTK	*Natural Language Toolkit*
RNN	*Recurrent Neural Network*

1 Introduction

Learning is one of the most essential parts of human life. From the beginning to the end, human beings acquire knowledge and skills. Learning means progress, additional value, failing and repeating. It enables growth and improvement.

In biology, learning is based on a specific strengthening of the connection of certain nerve cells in the central nervous system by facilitating signal transmission at the synapses through appropriate modifications. Being a huge amendable network of connected neurons the nervous system served as a role model for a research field called *Machine Learning*. This term was coined by A. Samuel[1] [14] in 1959, who distinguished two general approaches to the problem of machine learning: a general-purpose randomly connected Neural Network approach and a special-purpose highly organized network. Following a publication of W. McCulloch about the comparison of a computer with the nervous system of a flatworm in 1949 [8], he stated:

"A comparison between the size of the switching nets
that can be reasonably constructed or simulated at the present time
and the size of the neural nets used by animals,
suggests that we have a long way to go before we obtain practical devices."

– Arthur Lee Samuel (1959)

Less than 60 years later, today we have a lot of practical devices using machine learning and artificial intelligence technologies in our everyday life. Especially the processing and understanding of spoken or written natural language has a wide range of applications. One of those applications are advisory artificial conversational agents (ACA), chat-bots in short. They are designed to give natural language answers to natural language questions, making it as easy as possible for users to interact with a special system. ALEX is an example of an ACA that is able answer questions about

1 Arthur Lee Samuel was an early researcher in machine learning and artificial intelligence. He developed the first successful self-learning program: the Samuel-Checkers game [14].

courses and modules of the TU Berlin. This thesis aims to improve the understanding and learning of natural language of ALEX with artificial neural networks.

1.1 Scope of this Thesis

The scope of this thesis is the development of a neural network based part-of-speech tagger for the advisory Artificial Conversational Agent ALEX, the training of different language models and their evaluation with corresponding test sets.

In order to accomplish the new language models, two different neural network architectures are implemented: A feed-forward neural network and a recurrent neural network. For the training of both neural network implementations, a corpus of tagged language data is generated with the help various input templates, which are created on the basis of logged user input data.

To evaluate the language models, a data set of known data[2] and unknown data[3] is created. On the basis of this evaluation, both neural network models and the HMM are compared to each other.

In accordance to the evaluation results, the former HMM based part-of-speech tagger is then replaced by this new tagger. To guarantee a seamless integration, the new tagger is implemented as a separate module with the same program interface the old tagger already utilizes. This way no other components of the conversational agent have to be changed and the effort of the replacement is kept minimal.

2 Data, that was already used for the training of the model
3 Data, that includes words and sentence structures, that didn't occur in the training data sets

1.2 Related Work

This thesis is build upon the work of T. Michael [10], who describes the design and implementation of ALEX in detail. The conversational agent was implemented for the purpose of helping students of the TU Berlin to organize their studies by providing a simple way to gain information about modules and courses. It utilizes two separate already existing baseline systems by merging their data into one relational database. This database is used as the central access point for the information that users want to retrieve.

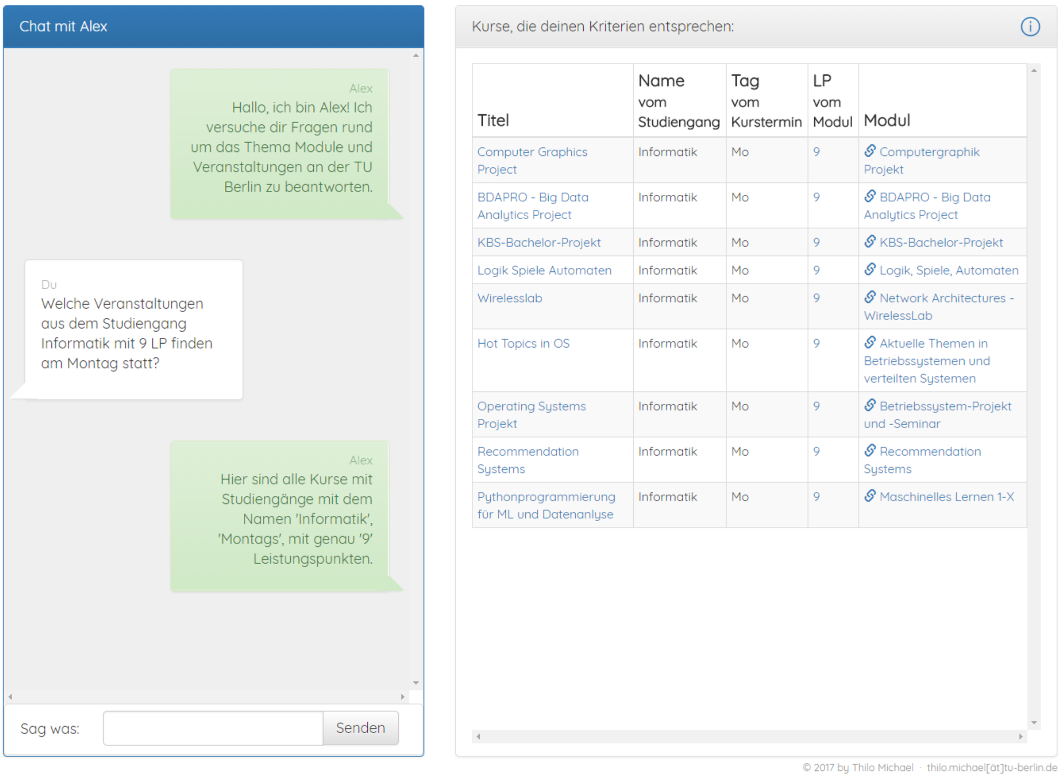

Figure 1.1: The user interface of ALEX. The left section contains the conversation with the Agent and a field where the user can type. The right section shows the result of the generated database query in tabular form.
In this example, the user asked for all courses of the subject *computer science* that provide 9 ECTS and are scheduled on a Monday. The agent answered accordingly and provided a list of 9 courses that fulfill the conditions.
This image was captured at the 21st April 2018.

ALEX consists of several processing modules:

- The **tagging module** uses a Hidden Markov Model to calculate the parts of speech for the user input, later described in chapter 2.3
- The **query generation module** composes actual SQL queries from the tagging output data by recognizing the requested model and the return type
- The **filter extraction module** provides refinement and constraint handling for the query generator
- The **response generation module** formulates answers for the user input in natural language by processing the generated query, the recognized model and the conversation state.

Moreover ALEX provides a user interface which utilizes web technologies and can be accessed in a web browser. Figure 1.1 shows the user interface where the user asked one question and the agent returned the result in tabular form and answered accordingly.

The focus of this thesis lies on the tagging module, as the main objective is to replace the Hidden Markov Model by Artificial Neural Networks.

1.2.1 The Hidden Markov Model

The Hidden Markov Model (HMM) is a probabilistic finite state machine that solves classification problems in general. It uses the observable output data of a system to derive hidden information from it. Among other applications, HMMs are used especially for speech recognition tasks.

The preliminary work for HMMs was done by R. L. Stratonovich. He first described the conditional Markov processes in 1960 [17] that were used in the following years to describe simple Markov Models and later Hidden Markov Models (see Baum et. al. [3][2]). The latter became popular for solving the task of automatic recognition of continuous speech [1] along with other applications like pattern recognition in general, the analysis of biological sequences (e.g. DNA) [4] and part-of-speech tagging [6].

1.2.2 The Artificial Neural Network Model

Artificial Neural Networks are networks that process information inspired by biological nervous system. They consist of connected computational units typically arranged in different layers. Such a unit (also called *artificial neuron*) can make calculations based on its inputs and pass the result to the next connected units. These connections are weighted, so that the weight can be adjusted depending on the activity of the unit. Thus a model of the features of the input data can be created.

After preceding research by W. McCulloch, W. Pitts [9] and D. Hebb [16] about arithmetical learning methods inspired by the connections of neurons in the 1940s, M. Minsky built the first neural network learning machine called SNARC (*Stochastic Neural Analog Reinforcement Computer)*[5] in 1951.

In the late 1950s, F. Rosenblatt developed the *Mark I Perceptron* computer and published a theorem of convergence of the perceptron[13] in 1958. He coined the term *perceptron* for an algorithm that was able to learn the assignment of input data to different classes. The perceptron represents a simple artificial neural network containing one single neuron at first[4]. F. Rosenblatt stated, that every function that is representable by the model can be learned with the proposed learning method. In 1960, B. Widrow presented the ADALINE[5] model of a neural network, where the input weights could already be adjusted by the learning algorithm [18].

A publication of M. Minsky and S. Papert [11] in 1969 analyzed and exposed some significant limitations of the basic perceptron. They pointed out, that it is not possible to learn functions without linear separability (e.g. the

4 Chapters 3.1 and 3.2 explain the architecture of different neural network structures in detail
5 ADALINE is an acronym for Adaptive Linear Neuron

exclusive-or problem). Due to these limitations and the fact, that the processing power of computers at that time was not sufficient for larger neural networks, the research interest in artificial neural networks decreased in the following years.

In 1982, J. Hopfield presented a previously described Neural Network with feedback (known as *Hopfield network*), that was able to solve optimization problems like the *Traveling Salesman Problem*[6]. Neural Network approaches got more attention again, also because the first processors based on transistor technology (microprocessors) came onto the market in the early 1970s and replaced the previously used tube technology in the following years, which made computers smaller and cheaper and increased their processing capacity.

For the task of POS tagging, Neural Network models were now able to outperform HMM based tagger. H. Schmid created and trained a multilayer Feed-forward Neural Network in 1994 and was able to show, that it performed better than an HMM tagger [15] at that time. In 2000, Ma et. al. run a series of comparative experiments that proved, that the results of a neural network tagger were superior to those of statistical models like the HMM [7].

1.3 Structure of this Thesis

As introduction, this first chapter gave a short overview about the subject of natural language processing and part-of-speech tagging in general.

The second chapter describes structure and functionality of the already existing ACA ALEX with the main focus on its language model and tagging interface.

Chapter 3 explains the implementation of a part-of-speech tagging system with two different neural network approaches.

6 The problem of the traveling salesman or round trip problem: The order of places to be visited once should be chosen in such a way that the distance covered is minimal, whereby the last place is again the starting point (round trip).

The training of the language models including the retrieval of the training data and tuning of the training parameter is described in Chapter 4.

Chapter 5 shows the evaluation of each language model with a generated test set and their comparison.

In conclusion the final Chapter 6 discusses and summarizes the evaluation results and gives an outlook on future work.

2 ALEX: Artificial Conversational Agent

Design and Implementation of an Advisory Artificial Conversational Agent by T. Michael [10] provides a detailed and comprehensive description of ALEX as a compilation of different modules. This chapter focuses on the modules that are relevant for the language processing and therefore adapted during this thesis: The retrieval and processing of training data (section 2.2), the Hidden Markov Model tagger (section 2.3) and the tagging interface (section 2.4).

2.1 System Overview

The modular structure of ALEX allows the separation of different functions and therefore easier replaceability of certain functionalities. Besides a module crawler for current data retrieval of web content for the database and a frontend interface module, ALEX offers a tagging module. This module provides the training of a language model as well as the assignment of tags to the words of a given input sentence.

Figure 2.1 shows the original architecture of ALEX. Modules that are adapted or replaced by another approach are emphasized with an orange border.

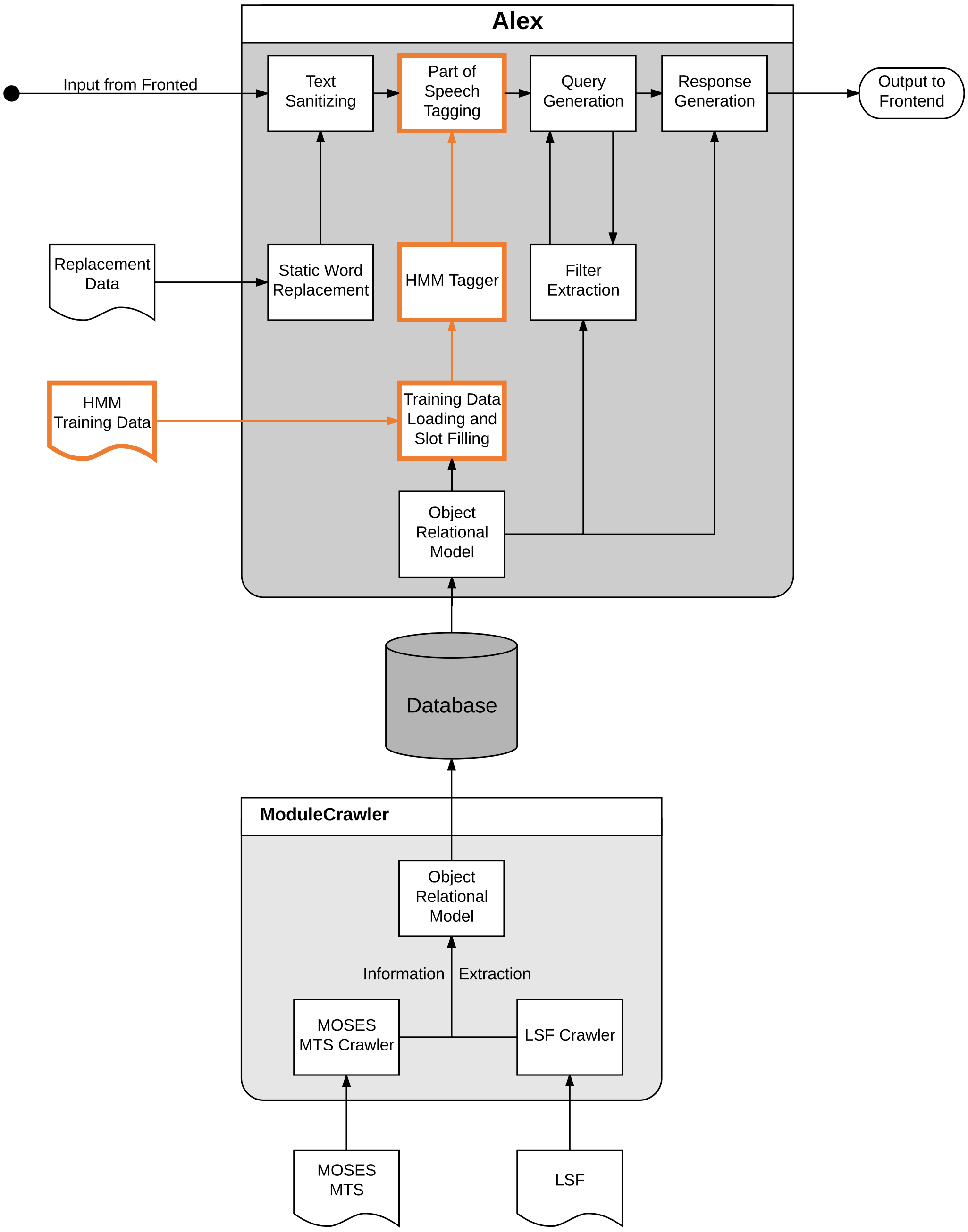

Figure 2.1: Overview of all components of ALEX. The orange bordered parts are components that lie within the scope of this thesis and are adapted or replaced. Original figure by T. Michael [10]

2.2 Training Data

In order to teach ALEX to correctly assign tags to words depending on their context, appropriate training data is required. The training data for ALEX proposed by T. Michael consists of 556,111 tagged sentences generated with 72 manually created sentence templates.

A sentence template is a sentence that provides the structure of a possible tagged training sentence with a proper syntax for slot filling. It consists of either special placeholders for specific data from the database (e.g. module titles), inline choices (e.g. the same sentence with every day of the week) or a marker to simply duplicate the sentence. The different slot filling forms can be combined or used multiple times in one sentence[1].

For the training of the Neural Network Models in this thesis, this slot filling mechanism is adopted and improved for the training with Artificial Neural Networks (see chapter 4).

2.3 The Hidden Markov Model Tagger

As described in section 1.2.1 of the introduction, the Hidden Markov Model (HMM) is a statistical tool that uses observable output data of a system to derive hidden information from it. Applications are image processing, gesture recognition and natural language processing tasks like speech recognition and part-of-speech tagging in particular.

In case of POS tagging, the observable states of the HMM represent the given sequence of words whereas the hidden states represent the corresponding parts of speech. The HMM calculates the joint probability of the whole sequence of hidden states with the help of transmission and output probabilities. Subsequently it finds the maximum probability of all possible state sequences and decides as a result, which parts of speech are most likely applied to the words of the input sequence.

1 T. Michael describes the training data structure in detail in chapter 3.4.2 of *Design and Implementation of an Advisory Artificial Conversational Agent* [10].

Figure 2.2 illustrates an example of a state sequence with three hidden states (part of speech tags) and the observed word sequence in an HMM. The calculation of the joint probability P of the word sequence in this case is shown in equation 2.1 as the product of transmission and output probabilities.

$$P = p_{start} \cdot p_{out,1} \cdot p_{trans,1} \cdot p_{out,2} \cdot p_{trans,2} \cdot p_{out,3} \tag{2.1}$$

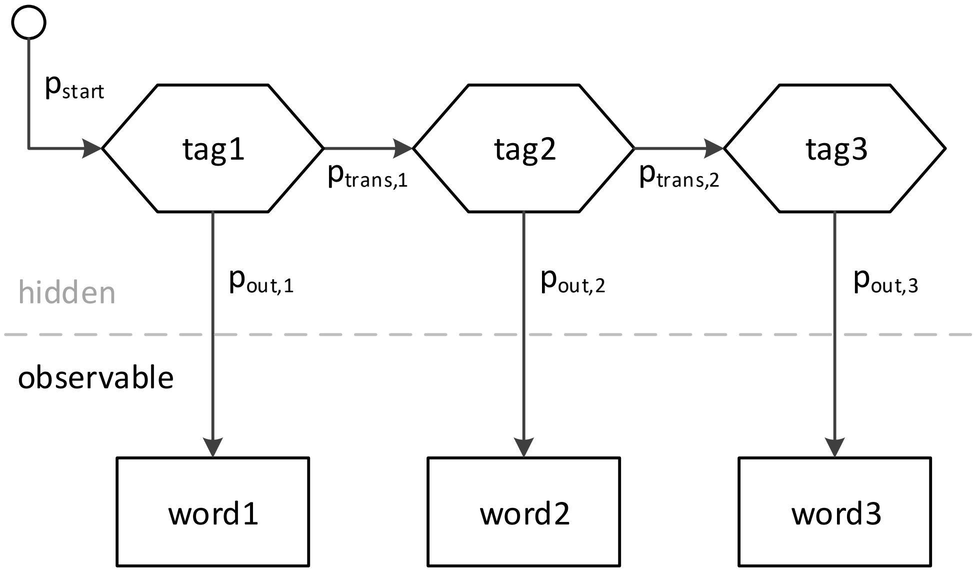

Figure 2.2: An example of a state sequence of three hidden states (tag1 – tag3) and an observed sequence of three words (word1 – word3) in an Hidden Markov Model. p_{start} denotes the start probability, p_{trans} the transmission probabilities between hidden states and p_{out} the output probabilities between a hidden state and an output.

For this purpose, an HMM is included in ALEX. According to T. Michael [10], a tagging scheme was developed to extract exactly the information from a user input that is needed to successfully create a database query and return the information the user asked for. This tagging scheme was intentionally built domain independent, giving the opportunity to make ALEX an ACA for any topic providing a corresponding database and training data.

To maintain this universal applicability as well as the compatibility to other modules of ALEX, the Neural Network approaches presented in this thesis use the same tagging scheme ALEX already utilizes. For a better understanding of the evaluation results in Chapter 5, table 2.1 gives an overview of the 6 different classes of tags that are used by ALEX.

	FORMATS	DESCRIPTION	EXAMPLE

`R`	*Return-tags*, describing data that is expected to be returned	`R_LIST`
		`R_SINGLE`
		`R_COUNT`
			*"**Which** modules ..."*
			*"**Which** module ..."*
			*"**How many** modules ..."*

`M`	*Model-tags*, describing the database model, e.g. `M_MTSModule` or `M_Course`	`M_[MODEL]`	*"Which **modules** ..."* *"Which **courses** ..."*
`C`	*Constraint-tags*, filtering the result set, given a database model and corresponding field, e.g. `C_MTSModule:ects`	`C_[MODEL]:[FIELD]`	*"Modules with **6** ects ..."*
`P`	*Property-tags*, indicating to include fields in the result set, e.g. `P_MTSModule:ects`	`P_[MODEL]:[FIELD]`	*"Modules with 6 **ects** ..."*

`Q`	*Comparison-tags*, describing an equal, greater than or less than constraint	`Q_EQ`
		`Q_LT`
		`Q_GT`
			*"... with **exactly** 6 ects ..."*
			*"... **less than** 6 ects ..."*
			*"... **more than** 6 ects ..."*

`X`	*Extra-tags*, describing words that are not relevant for the database query[2]	`X`
		`X_[WORD]`
		`X_[MODEL]:[FIELD]`
			*"**and**", "**of**", "**is**"*
			*"I need **help**"*
			*"**Professor** John Doe"*

Table 2.1: Overview of the tagging scheme used in ALEX, consisting of 6 different classes of tags with a total of 12 different formats. The examples contain **emphasized** words that belong to the corresponding tag format. Detailed explanation of the tagging classes and its formats is given by T. Michael [10].

2.4 Tagging Interface

As described in the previous chapter, the implementation of the tagging module of ALEX utilizes a Hidden Markov Model for the part-of-speech tagging. ALEX uses an already existing implementation of the HMM Tagger from the Natural Language Toolkit (NLTK)[3], called `HiddenMarkovModelTagger`.

To replace the existing tagger, a new tagger has to provide a class with two methods: `train` and `tag`. These methods are used to create the language model and apply it to unknown data.

The `train` method creates a new instance of the tagger class, trains this class with the given training data and returns it. The training data itself must be a list of sentences, where a sentence is a list of tuples, containing each word of this sentence and its corresponding tag. The following exemplifies the structure of the training input data containing two sentences where each word is tagged with *TAG*:

```
[
  [ ('the', TAG), ('dog', TAG), ('is', TAG), ('running', TAG) ],
  [ ('the', TAG), ('cat', TAG), ('sleeps', TAG), ('all', TAG), ('day', TAG) ]
]
```

The `tag` method attaches a tag to each word of an input sentence, according to the previously trained language model. The input has to be an unknown sentence as a simple list of words:

```
[ 'an', 'unknown', 'test', 'sentence' ]
```

The output is a corresponding list of tuples containing a word and its assigned tag:

```
[ ('an', TAG), ('unknown', TAG), ('test', TAG), ('sentence', TAG) ]
```

2 This can be either words with no special meaning at all (tagged with `X`), or words that have no meaning for the database query but for the system itself (e.g. the tag `X_HELP` for the word *"help"*) or words that lead to a particular constraint (like the tag `X_Person:fullname` for the word *"Professor"*, that leads to a name).

3 The Natural Language Toolkit is a collection of *Python* programming libraries for natural language processing, see `http://nltk.org`

3 Part-of-Speech Tagging with Neural Networks

... Ma et. al. have shown before, that the neural network approach outperforms part-of-speech tagger that are based on statistical models (like the HMM) [7].

3.1 Feed-forward Neural Network Model

...

3.1.1 Architecture

...

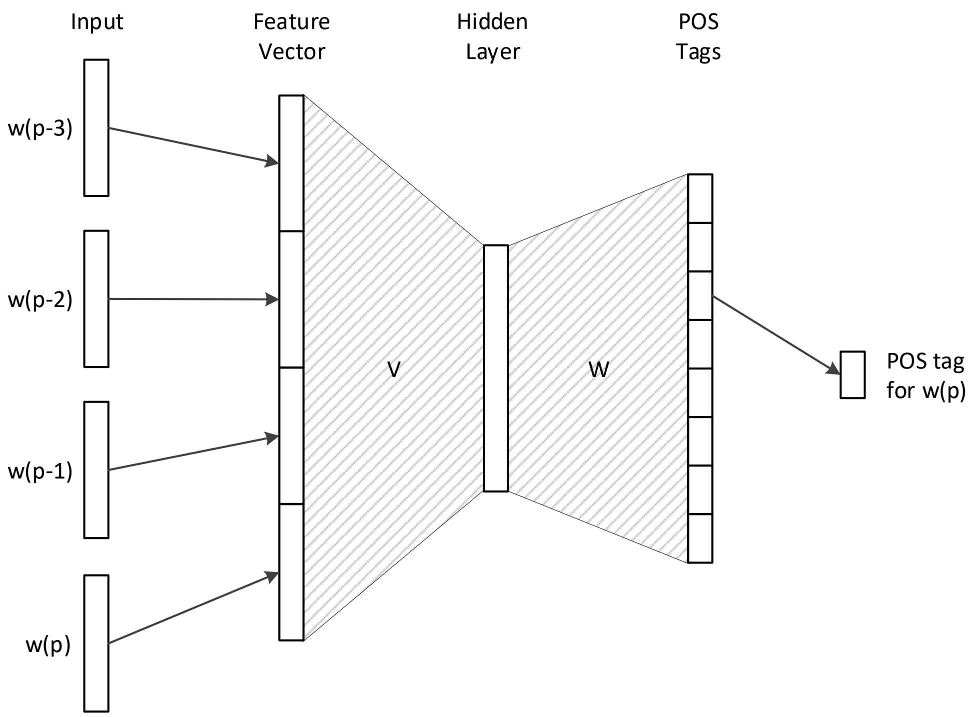

Figure 3.1: The structure of a feed-forward neural network. The feature vector is built by the initial vectors of the corresponding input word on position p and by its 3 preceding words (here as an example, the number of predecessors can of course vary).

3.1.2 Implementation

...

3.2 Recurrent Neural Network Model

...

3.2.1 Architecture

...

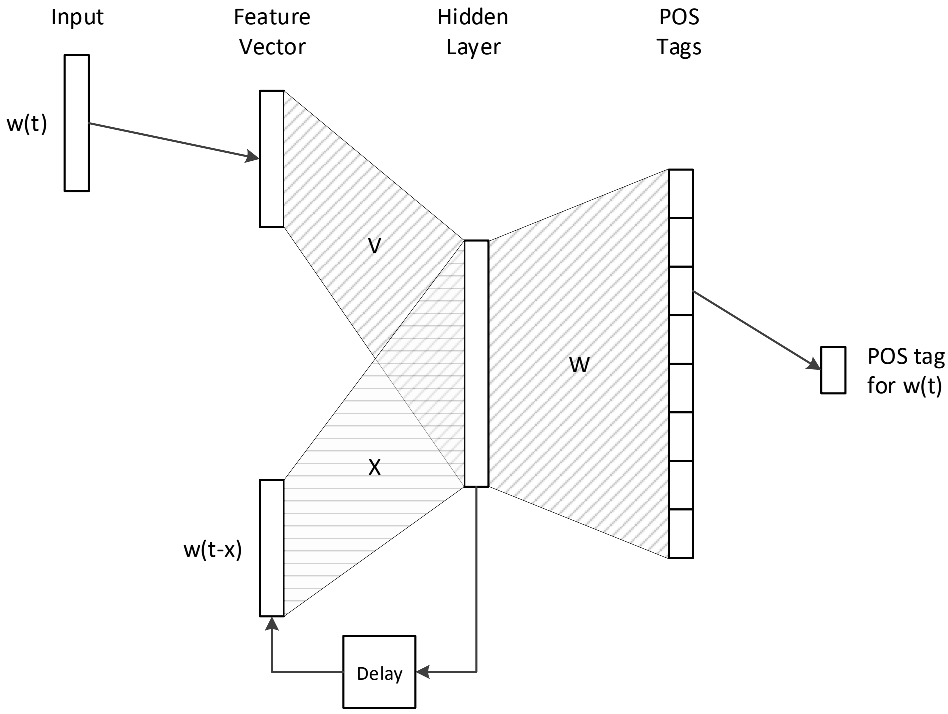

Figure 3.2: The structure of a recurrent neural network. The feature vector is the initial vector of the corresponding input word at time `t`. The output of the hidden layer from previously trained words (here as an example at time `x`) is fed back into the same hidden layer for the current word.

3.2.2 Implementation

...

4 Training of Language Models

To obtain an optimal language model with the Neural Network approach proposed in this thesis, as much annotated training data as possible is required. The following sections describe the generation of correctly tagged training data and the training of different language models due to parameter variation based on this generated training corpus.

4.1 Training Data Corpus

To create a training corpus with tagged sentences, the already existing sentence template set from ALEX is used as the basis for an improved and extended template set. In addition, a log of user input data[1] provides useful information about possible input sentences, that are not yet considered in the template set.

As described in chapter 2.2, the HMM tagger of ALEX uses 72 sentence templates to generate annotated training data. However the distribution of the generated sentences is highly unbalanced. Due to the combination of placeholders for different database fields and models, that are semantically meaningless, a huge number of training sentences exist, that are only partially suitable for training.

An analysis of the generated training corpus that was used to train the HMM tagger showed, that one single template out of the 72 sentence templates created more than 84% of the whole training corpus. This sentence template is the following[2]:

```
Welche   Module  werden von   Prof   {Person:firstname} {Person:lastname} angeboten
R_LIST M_MTSModule  X    X  X_Person C_Person:fullname  C_Person:fullname     X
```

1 The log data that was available for this thesis started on 30th of August, 2017 until 27th of April, 2018 and contained 1293 entries of user input.

2 The corresponding English translation of this sentence template is: *Which modules are held by Professor <firstname> <lastname>*

It combines all first names and all last names of each existing person in the database, leading to a huge number of name combinations, that do not exist.

Another example is the combination of study program degrees and program names. This combination exists in 5 of the 72 sentence templates, whereby all degrees are assigned to all study programs 5 times, although not every combination exists in reality. The following is an example of one of this sentence templates[3]:

```
Welche   Module    kann ich  im  {Program:degree}  {Program:name} belegen
R_LIST M_MTSModule  X    X   X   C_Program:degree  C_Program:name    X
```

To address this issue, the following adjustments for the new template set were made:

- Because of the very low occurrence in the log database, the slot `{Person:firstname}` was removed completely from that sentence template, which limits the user input to last names only[4]
- the slot `{Program:degree}` was partly replaced by the inline choice `(bachelor|master|diplom)`, which were the main terms asked for in the logs.

Also the wording occurring in the logs referring to linking words and actions was improved with the help of inline choices. The following table shows those improvements as a comparison of the former and the new sentence template:

3 The corresponding English translation of this sentence template is: *Which modules can I attend to in <program-degree> <program-name>*

4 This might seem like a degradation but is justified by the fact, that only 3 of 1293 log entries even contained a first name, always followed by the last name. In this specific use case, where all Persons are Professors and Lecturers, only using the last name appears legit.

FORMER TEMPLATE	NEW TEMPLATE	ENGLISH EQUIVALENT
Alle Module **vom** {Program:degree} {Program:name}	Alle Module **(vom | von | in | im)** {Program:degree} {Program:name}	*All modules* ***(of | in)*** *{Program:degree} {Program:name}*
Welche Module werden von Prof {Person:firstname} {Person:lastname} **angeboten**	Welche Module werden von Professor {Person:lastname} **(unterrichtet | angeboten | gehalten)**	*Which modules are* ***(taught | offered | held)*** *by Professor {Person:lastname}*
Wieviele LP **hat** das Modul {MTSModule:title}	**(Wieviele | Wieviel)** LP **(hat | bringt)** das Modul {MTSModule:title}	***How many*** *ects does the module {MTSModule:title}* ***(have | bring)***
Informationen zu Modul MTSModule:title	**(Informationen | Details | Mehr) (zu | zum)** Modul {MTSModule:title}	***(Information | details | more) of*** *the module {MTSModule:title}*

Table 4.1: An excerpt of the extension and improvement of the sentence templates by using inline choices. The last column provides the corresponding English translation for the new sentence template.

At last `28` new sentence templates were added, that provide a sentence structure that was found in the logs but not in the former template set. The new set contains `100` sentence or single word templates and can be found in the appendix A.1.

4.2 Parameter Tuning

...

5 Evaluation and Comparison

...

5.1 Test Design

Two evaluate the trained language models, two test sets were designed: A test set containing a selection of tagged sentences as they exist in the training corpus (here called the *known test*) and a test set, where the structure of the sentences from the known test was modified in a way, that it still remains semantically meaningful but does not occur in the training corpus (called the *unknown test*). The unknown test deliberately contains words, that do not exist in the vocabulary of the training corpus to evaluate, how the model handles completely unknown data.

The sentences are divided into different topics, to cover a wide range of possible input data. Attention was paid to a balanced number of sentences in the different areas, to ensure, that the evaluation result doesn't depend on only one or two topics.

The following tables shows the different topics, their respective number of evaluation sentences and one sentence as an illustration for each topic.

Topic	Count	Example
ECTS	0	*test*
Time	0	*test*
Faculty	0	*test*
Participants	0	*test*
Persons	0	*test*
Program	0	*test*
Modules	0	*test*
Chair	0	*test*
Exam	0	*test*
Courses	0	*test*
Locations	0	*test*

Table 5.1: The evaluation topics, the number of tagged test sentences and an example sentence for each topic.

ects time faculty participants persons program modules chair exam courses locations

6 Discussion and Conclusion

...

6.1 Summary

...

6.2 Discussion

...

6.3 Future work

... - additional module to check if generated trainings sentence returns a query result - evaluation based on semantical topics (like locations, persons, module titles, etc...)

[12]

Bibliography

[1] J. Baker. The DRAGON system–An overview. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):24–29, February 1975.

[2] Leonard E. Baum. An Equality with Applications to Statistical Prediction for Functions of Markov Process and to a Model to Ecology. *Bulletin of the American Mathematical Society*, 73:360–363, 1967.

[3] Leonard E. Baum and Ted Petrie. Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563, 1966.

[4] M.J. Bishop and E.A. Thompson. Maximum likelihood alignment of DNA sequences. *Journal of Molecular Biology*, 190(2):159 – 165, 1986.

[5] Daniel Crevier. AI: The Tumultuous Search for Artificial Intelligence. 1993.

[6] Julian Kupiec. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech & Language*, 6(3):225 – 242, 1992.

[7] Qing Ma, Kiyotaka Uchimoto, Masaki Murata, and Hitoshi Isahara. Elastic Neural Networks for Part of Speech Tagging. 5, January 2000.

[8] W. S. Mcculloch. The brain computing machine. *Electrical Engineering*, 68(6):492–497, June 1949.

[9] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, December 1943.

[10] Thilo Michael. Design and Implementation of an Advisory Artificial Conversational Agent, October 2016.

[11] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, 1969.

[12] Andreas Müller. Analyse von Wort-Vektoren deutscher Textkorpora, June 2015.

[13] F. Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. *Psychological Review*, pages 65–386, 1958.

[14] A. L. Samuel. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research

and Development, 3(3):210–229, July 1959.

[15] Helmut Schmid. Part-of-Speech Tagging with Neural Networks. October 1994.

[16] G. L. Shaw. Donald Hebb: The Organization of Behavior. In Günther Palm and Ad Aertsen, editors, *Brain Theory*, pages 231–233, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

[17] R. L. Stratonovich. Conditional Markov Processes. *Theory of Probability & Its Applications*, 5(2):156–178, 1960.

[18] B. Widrow. An adaptive "ADALINE" neuron using chemical "Memistors". *Stanford Electronics Laboratories Technical Report*, 1553–2(2), 1960.

A Appendix

A.1 Set of sentence templates

```
Welche Module kann (ich|man) (im|in) (bachelor|master|diplom) {Program:name} belegen
R_LIST M_MTSModule X X X C_Program:degree C_Program:name X

Welche Module kann (ich|man) (im|in) {Program:name} (bachelor|master|diplom) belegen
R_LIST M_MTSModule X X X C_Program:name C_Program:degree X

Alle Module (im|in) (bachelor|master|diplom) {Program:name}
R_LIST M_MTSModule X C_Program:degree C_Program:name

Alle Module (im|in) {Program:name} (bachelor|master|diplom)
R_LIST M_MTSModule X C_Program:name C_Program:degree

Alle Module (vom|von|in) {Program:name}
R_LIST M_MTSModule X C_Program:name

Alle Module (vom|von|in|im) {Program:degree} {Program:name}
R_LIST M_MTSModule X C_Program:degree C_Program:name

Welche Module gehören zum (bachelor|master|diplom) {Program:name}
R_LIST M_MTSModule X X C_Program:degree C_Program:name

Welche Module gehören zum {Program:name} (bachelor|master|diplom)
R_LIST M_MTSModule X X C_Program:name C_Program:degree

Zeige mir alle Module im Wahlpflichtbereich {Program:name} (bachelor|master|diplom)
R_LIST X X M_MTSModule X C_CourseRegulation:group C_Program:name C_Program:degree

Zeige mir alle Module im Wahlpflichtbereich (bachelor|master|diplom) {Program:name}
R_LIST X X M_MTSModule X C_CourseRegulation:group C_Program:name C_Program:degree

Welche Module gibt es (im|in) {Program:name}
R_LIST M_MTSModule X X X C_Program:name

Welche Module gibt es im Studiengang {Program:name}
R_LIST M_MTSModule X X X X_Program:name C_Program:name

Welche Module mit dem Abschluss {Program:degree} gibt es
R_LIST M_MTSModule X X X_Program:degree C_Program:degree X X

Ich suche alle Module (vom|von) {Program:name}
X X R_LIST M_MTSModule X C_Program:name

Ich suche alle Module (vom|von|im) {Program:degree} {Program:name}
X X R_LIST M_MTSModule X C_Program:degree C_Program:name

Module mit dem Namen {MTSModule:title}
M_MTSModule X X X_MTSModule:title C_MTSModule:title

Welche Module kann (ich|man) im {Program:name} {CourseRegulation:group} belegen
R_LIST M_MTSModule X X X C_Program:name C_CourseRegulation:group X

Welche Veranstaltungen im {Program:degree} {CourseRegulation:group} gibt es
```

A Appendix

```
R_LIST M_Course X C_Program:degree C_CourseRegulation:group X X

Welche Module haben (0|1|2|3|4|5|6|7|8|9|10|11|12) ects
R_LIST M_MTSModule X C_MTSModule:ects C_MTSModule:ects

Welche Module haben mehr als (0|1|2|3|4|5|6|7|8|9|10|11|12) ects
R_LIST M_MTSModule X Q_GT X C_MTSModule:ects C_MTSModule:ects

Welche Module haben weniger als (2|3|4|5|6|7|8|9|10|11|12) ects
R_LIST M_MTSModule X Q_LT X C_MTSModule:ects C_MTSModule:ects

Welche Module haben genau (0|1|2|3|4|5|6|7|8|9|10|11|12) ects
R_LIST M_MTSModule X X C_MTSModule:ects C_MTSModule:ects

Welche Veranstaltungen finden (Mo|Di|Mi|Do|Fr|Sa) statt
R_LIST M_Course X C_CourseDate:day X

Welche Veranstaltungen finden am (Mo|Di|Mi|Do|Fr|Sa) statt
R_LIST M_Course X X C_CourseDate:day X

Welche Veranstaltungen finden am (Mo|Di|Mi|Do|Fr|Sa) nach (7|8|9|10|11|12|13|14|15|16|17|18|19|20) Uhr statt
R_LIST M_Course X X C_CourseDate:day Q_GT C_CourseDate:startTime C_CourseDate:startTime X

Welche Veranstaltungen finden am (Mo|Di|Mi|Do|Fr|Sa) vor (8|9|10|11|12|13|14|15|16|17|18|19) Uhr statt
R_LIST M_Course X X C_CourseDate:day Q_LT C_CourseDate:startTime C_CourseDate:startTime X

Welche Veranstaltungen finden am (Mo|Di|Mi|Do|Fr|Sa) um (8|9|10|11|12|13|14|15|16|17|18|19) Uhr statt
R_LIST M_Course X X C_CourseDate:day X_CourseDate:startTime C_CourseDate:startTime C_CourseDate:startTime X

Welche Veranstaltungen sind (Mo|Di|Mi|Do|Fr|Sa)
R_LIST M_Course X C_CourseDate:day

Welche Veranstaltungen sind am (Mo|Di|Mi|Do|Fr|Sa)
R_LIST M_Course X X C_CourseDate:day

Welche Veranstaltungen sind am (Mo|Di|Mi|Do|Fr|Sa) nach (7|8|9|10|11|12|13|14|15|16|17|18|19|20) Uhr
R_LIST M_Course X X C_CourseDate:day Q_GT C_CourseDate:startTime C_CourseDate:startTime

Welche Veranstaltungen sind am (Mo|Di|Mi|Do|Fr|Sa) vor (8|9|10|11|12|13|14|15|16|17|18|19) Uhr
R_LIST M_Course X X C_CourseDate:day Q_LT C_CourseDate:startTime C_CourseDate:startTime

Welche Veranstaltungen sind am (Mo|Di|Mi|Do|Fr|Sa) um (8|9|10|11|12|13|14|15|16|17|18|19) Uhr
R_LIST M_Course X X C_CourseDate:day X_CourseDate:startTime C_CourseDate:startTime C_CourseDate:startTime

Welche Veranstaltung hält {Person:lastname} am (Mo|Di|Mi|Do|Fr|Sa)
R_LIST M_Course X C_Person:fullname X_CurseDate:day C_CourseDate:day

Welche Module mit mehr als (0|1|2|3|4|5|6|7|8|9|10|11|12) ects kann (ich|man) im {CourseRegulation:group} belegen
R_LIST M_MTSModule X Q_GT X C_MTSModule:ects C_MTSModule:ects X X X C_CourseRegulation:group X

Welche Module werden von Professor {Person:lastname} (unterrichtet|angeboten|gehalten)
R_LIST M_MTSModule X X X_Person C_Person:fullname X
```

A Appendix

```
Wer ist der Modulverantwortliche des Moduls {MTSModule:title}
M_Person X X X_Person X X_MTSModule:title C_MTSModule:title

Wer ist verantwortlich für das Moduls {MTSModule:title}
M_Person X X X X X_MTSModule:title C_MTSModule:title

Bei wem findet das Moduls {MTSModule:title} statt
X M_Person X X X_MTSModule:title C_MTSModule:title X

Welche Kurse werden von Professor {Person:lastname} (unterrichtet|angeboten|gehalten)
R_LIST M_Course X X X_Person C_Person:fullname X

Welche Kurse (unterrichtet|bietet|hält) Professor {Person:lastname}
R_LIST M_Course X X_Person C_Person:fullname

Wie viele ects (hat|bringt) das Modul {MTSModule:title}
R_SINGLE X_count R_MTSModule:ects X X C_MTSModule:title C_MTSModule:title

(Wieviele|Wieviel) ects (hat|bringt) das Modul {MTSModule:title}
X_count R_MTSModule:ects X X C_MTSModule:title C_MTSModule:title

(Informationen|Details|Mehr) (zu|zum) Modul {MTSModule:title}
R_SINGLE X X_MTSModule:title C_MTSModule:title

Welche Modulkataloge gibt es (im|in) {Program:degree} {Program:name}
R_LIST M_CourseRegulation X X X C_Program:degree C_Program:name

Zeige den Modulkatalog (im|in) {Program:degree} {Program:name}
R_LIST X M_CourseRegulation X C_Program:degree C_Program:name

Welche Module werden vom Fachgebiet {Chair:name} angeboten
R_LIST M_MTSModule X X X_Chair:name C_Chair:name X

Module vom Fachgebiet {Chair:name}
M_MTSModule X X_Chair:name C_Chair:name

Alle Module vom Fachgebiet {Chair:name}
R_LIST M_MTSModule X X_Chair:name C_Chair:name

Zeige mir alle Module vom Fachgebiet {Chair:name}
R_LIST X X M_MTSModule X X_Chair:name C_Chair:name

Welche Studiengänge von der Fakultät (1|2|3|4|5|6|7) gibt es
R_LIST M_Program X X X_Institute:faculty C_Institute:faculty X X

Module von der Fakultät (1|2|3|4|5|6|7)
M_MTSModule X X X_Institute:faculty C_Institute:faculty

Alle Module von der Fakultät (1|2|3|4|5|6|7)
R_LIST M_MTSModule X X X_Institute:faculty C_Institute:faculty

Zeige mir alle Module von der Fakultät (1|2|3|4|5|6|7)
R_LIST X X M_MTSModule X X X_Institute:faculty C_Institute:faculty

Welche Studiengänge von Fakultät (1|2|3|4|5|6|7) gibt es
R_LIST M_Program X X_Institute:faculty C_Institute:faculty X X

Module von Fakultät (1|2|3|4|5|6|7)
M_MTSModule X X_Institute:faculty C_Institute:faculty

Alle Module von Fakultät (1|2|3|4|5|6|7)
R_LIST M_MTSModule X X_Institute:faculty C_Institute:faculty
```

```
Zeige mir alle Module von Fakultät (1|2|3|4|5|6|7)
R_LIST X X M_MTSModule X X_Institute:faculty C_Institute:faculty

Veranstaltungen mit {ExamElement:description} als Prüfung
M_Course X C_ExamElement:description X X_ExamElement

Welche Veranstaltungen haben die Prüfung {ExamElement:description}
R_LIST M_Course X X X_ExamElement C_ExamElement:description

Kurse die am (Mo|Di|Mi|Do|Fr|Sa) angeboten werden
M_Course X X C_CourseDate:day X X

Welche Veranstaltungen werden an einem (Mo|Di|Mi|Do|Fr|Sa) angeboten
R_LIST M_Course X X X C_CourseDate:day X

Wann ist das erste Treffen von {Course:title}
M_CourseDate X X R_FIRST X X C_Course:title

Wann ist die erste Veranstaltung von {Course:title}
M_CourseDate X X R_FIRST X X C_Course:title

Wann findet {Course:title} statt
M_CourseDate X C_Course:title X

In welchen Studiengängen gibt es das Modul {MTSModule:title}
X R_LIST M_Program X X X X_MTSModule:title C_MTSModule:title

Welche Veranstaltungen des Fachgebiets {Chair:name} gibt es
R_LIST M_Course X X_Chair:name C_Chair:name X X

Module mit dem Titel {MTSModule:title}
M_MTSModule X X X_MTSModule:title C_MTSModule:title

Welche Veranstaltungen finden (im|in) Raum {CourseDate:room} statt
R_LIST M_CourseDate X X X_Room C_CourseDate:room X

Alle Veranstaltungen (im|in) Raum {CourseDate:room}
R_LIST M_CourseDate X X_Room C_CourseDate:room

Zeige mir alle Veranstaltungen (im|in) Raum {CourseDate:room}
X X R_LIST M_CourseDate X X_Room C_CourseDate:room

Welche Veranstaltungen werden {CourseDate:cycle} angeboten
R_LIST M_Course X C_CourseDate:cycle X

Welche Module haben eine Platzbeschränkung von mehr als (1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|
   16|17|18|19|20|21|22|23|24|25|26|27|28|29|30|31|32|33|34|35|40|45|50|55|60|65|70|75|80|85|
   90|95|100|200|300) Teilnehmern
R_LIST M_MTSModule X X X_ParticipantLimitation X Q_GT X C_MTSModule:participantLimitation
   X_ParticipantLimitation

Welche Module haben eine Platzbeschränkung von weniger als (1|2|3|4|5|6|7|8|9|10|11|12|13|14|
   15|16|17|18|19|20|21|22|23|24|25|26|27|28|29|30|31|32|33|34|35|40|45|50|55|60|65|70|75|80|
   85|90|95|100|200|300) Teilnehmern
R_LIST M_MTSModule X X X_ParticipantLimitation X Q_LT X C_MTSModule:participantLimitation
   X_ParticipantLimitation

Welche Module haben eine Platzbeschränkung von genau (1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|16|
   17|18|19|20|21|22|23|24|25|26|27|28|29|30|31|32|33|34|35|40|45|50|55|60|65|70|75|80|85|90|
   95|100|200|300) Teilnehmern
```

```
R_LIST M_MTSModule X X X_ParticipantLimitation X Q_EQ C_MTSModule:participantLimitation ↩
   X_ParticipantLimitation

Welche Module sind (beschränkt|begrenzt) auf (1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|16|17|18|19↩
   |20|21|22|23|24|25|26|27|28|29|30|31|32|33|34|35|40|45|50|55|60|65|70|75|80|85|90|95|100|↩
   200|300) (Teilnehmer|Studenten|Personen)
R_LIST M_MTSModule X X_ParticipantLimitation X C_MTSModule:participantLimitation ↩
   X_ParticipantLimitation

Welche Personen bieten Module mit mehr als (1|2|3|4|5|6|7|8|9|10|11|12|15|30) ects an
R_LIST M_Person X R_MTSModule:title X Q_GT X C_MTSModule:ects C_MTSModule:ects X

Bitte nur welche die (ich|man) im Studiengang {Program:name} studieren kann
X_PLEASE X_ONLY X X X X X_Program:name C_Program:name X X

Welche Kurse des Moduls {MTSModule:title} kann (ich|man) (Mo|Di|Mi|Do|Fr|Sa) belegen
R_LIST M_Course X X_MTSModule:title C_MTSModule:title X X C_CourseDate:day X

################################################################################
#                              Wörter ohne Kontext                             #
################################################################################

{MTSModule:title}
C_MTSModule:title

im master [100]
X C_Program:degree

im bachelor [100]
X C_Program:degree

nur [30]
X_ONLY

bitte [30]
X_PLEASE

hilfe [30]
X_HELP

hallo [30]
X_GREETING

Guten (Tag|Morgen|Abend|Nacht|Nachmittag) [5]
X_GOOD X_GREETING

Ich brauche hilfe [10]
X X X_HELP

scheiße [30]
X_CURSE

ja [30]
X_YES

nein [30]
X_NO

zurück [30]
X_BACK

Leben
```

```
X_HITCH_LIFE

Universum
X_HITCH_UNIVERSE

Rest
X_HITCH_EVERYTHING

################################################################################
#                              Personal stuff                                  #
################################################################################

(Was|Wie) ist dein Alter
X X X_PERSONAL X_AGE

(Was|Wie) ist dein Name
X X X_PERSONAL X_NAME

Wie heißt du
X X_NAME X_PERSONAL

Was bist du von Beruf
X X X_PERSONAL X X_PROFESSION

Was ist dein (Beruf|Job)
X X X_PERSONAL X_PROFESSION

Was ist dein Auftrag
X X X_PERSONAL X_MISSION
```