Technische Universität Berlin

Quality and Usability Lab

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	Prof. Dr. ???

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 January 6, 2018

_____

Unterschrift

Abstract

...

Zusammenfassung

...

Contents

List of Figures

List of Tables

Abbreviations

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

1 Introduction

1.1 Scope of this Thesis

1.2 Related Work

...

2 ALEX: Artificial Conversational Agent

...

2.1 System Overview

...

2.2 Hidden Markov Model

...

2.3 Tagging Interface

The modular structure of ALEXallows for easier separation of various functions and therefore easier replaceability of certain functionalities. One of these modules is the tagger, which is used to train a language model on the one hand and to assign tags to the words of a given input sentence on the other hand.

The implementation of this tagger utilizes a Hidden Markov Model (HMM), which is a statistical model that is particularly used for pattern recognition, speech recognition and part-of-speech tagging. ALEXuses an already existing implementation of the HMM Tagger from the Natural Language Toolkit (NLTK)[1], called `HiddenMarkovModelTagger`.

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

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) ]
```

3 Evaluation and Comparison

...

4 Discussion and Conclusion

...

A First appendix

A.1 test

...