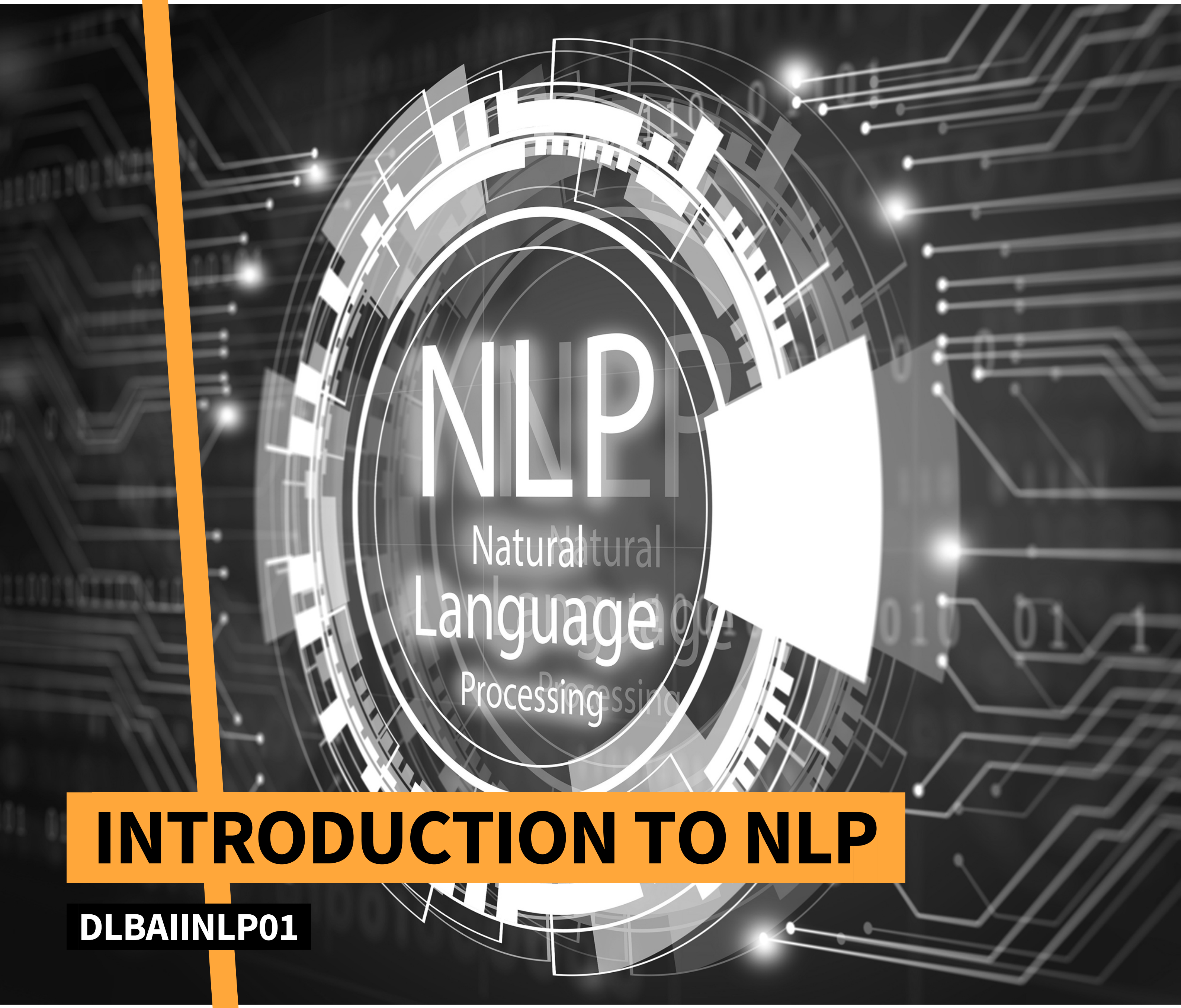
Course Book



INTRODUCTION TO NLP

DLBAIINLP01

INTERNATIONAL
UNIVERSITY OF
APPLIED SCIENCES

INTRODUCTION TO NLP

MASTHEAD

Publisher:
IU Internationale Hochschule GmbH
IU International University of Applied Sciences
Juri-Gagarin-Ring 152
D-99084 Erfurt

Mailing address:
Albert-Proeller-Straße 15-19
D-86675 Buchdorf
media@iu.org
www.iu.de

DLBAIINLP01
Version No.: 001-2024-0411

Prof. Dr. Kristina Schaaff

TABLE OF CONTENTS

INTRODUCTION TO NLP

Introduction
Signposts Throughout the Course Book . . . 6
Suggested Readings . . . 7
Learning Objectives . . . 8

Unit 1
Basic Terms and Concepts 9

1.1 What is NLP? . . . 10
1.2 Syntax . . . 14
1.3 Semantics . . . 15
1.4 Prosodics . . . 17
1.5 Grammar . . . 19

Unit 2
Language and Speech 25

2.1 Human Vocal Apparatus . . . 26
2.2 Speech Production . . . 29
2.3 Phonetics . . . 31

Unit 3
Challenges in NLP 41

3.1 Data for NLP . . . 42
3.2 Evaluation of NLP Systems . . . 46
3.3 Domain Challenges . . . 52
3.4 Multilingual Application . . . 55

Unit 4
Techniques 59

4.1 Rules Versus Statistics . . . 60
4.2 Regular Expressions . . . 61
4.3 N-Grams . . . 64
4.4 Vectorizing Data . . . 65
4.5 NLP Models . . . 71

Unit 5
Application Scenarios 77

5.1 Speech Recognition and Synthesis 78
5.2 Machine Translation 82
5.3 Information Extraction 83
5.4 Sentiment Analysis 86
5.5 Chatbots 89
5.6 NLP With Python 95

Appendix
List of References 102
List of Tables and Figures 113

INTRODUCTION

WELCOME

SIGNPOSTS THROUGHOUT THE COURSE BOOK

This course book contains the core content for this course. Additional learning materials can be found on the learning platform, but this course book should form the basis for your learning.

The content of this course book is divided into units, which are divided further into sections. Each section contains only one new key concept to allow you to quickly and efficiently add new learning material to your existing knowledge.

At the end of each section of the digital course book, you will find self-check questions. These questions are designed to help you check whether you have understood the concepts in each section.

For all modules with a final exam, you must complete the knowledge tests on the learning platform. You will pass the knowledge test for each unit when you answer at least 80% of the questions correctly.

When you have passed the knowledge tests for all the units, the course is considered finished and you will be able to register for the final assessment. Please ensure that you complete the evaluation prior to registering for the assessment.

Good luck!

SUGGESTED READINGS

GENERAL SUGGESTIONS

Jurafsky, D., & Martin, J. H. (2022) *Speech and language processing* (3rd ed. draft). Prentice Hall. Available online.

UNIT 1

Russell, S., & Norvig, P. (2021). *Artificial intelligence: A modern approach* (4th ed., Global ed.). Pearson Education. Chapter 24. http://search.ebscohost.com.pxz.iubh.de:8080/login.aspx?direct=true&db=cat05114a&AN=ihb.50081&site=eds-live&scope=site

UNIT 2

Levelt, W. J. (1999). Models of word production. *Trends in Cognitive Sciences, 3*(6), 223–232. http://search.ebscohost.com.pxz.iubh.de:8080/login.aspx?direct=true&db=edselp&AN=S1364661399013194&site=eds-live&scope=site

UNIT 3

Besacier, L., Barnard, E., Karpov, A., & Schultz, T. (2014). Automatic speech recognition for under-resourced languages: A survey. *Speech Communication, 56,* 85–100. http://search.ebscohost.com.pxz.iubh.de:8080/login.aspx?direct=true&db=edsbas&AN=edsbas.B38F3318&site=eds-live&scope=site

UNIT 4

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017, June 12). Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *31st Conference on Neural Information Processing Systems* (pp. 5999–6009). Curran Associates. http://search.ebscohost.com.pxz.iubh.de:8080/login.aspx?direct=true&db=edsbas&AN=edsbas.946C91CB&site=eds-live&scope=site

UNIT 5

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python.* O’Reilly. http://search.ebscohost.com.pxz.iubh.de:8080/login.aspx?direct=true&db=edsebk&AN=2694882&site=eds-live&scope=site

LEARNING OBJECTIVES

Natural language processing (NLP) stands at the crossroads of linguistics, computer science, and artificial intelligence (AI). In this course book, **Introduction to NLP**, you will learn about traditional approaches to NLP as well as state-of-the-art techniques in this field. The course starts with an introduction to the historical developments of NLP and presents its basic terms and concepts. Moreover, you will learn how language and speech are produced.

The course will give a comprehensive overview of the challenges that typically occur in NLP projects. Additionally, the most important techniques that are used in NLP are presented, including some related topics. Finally, the course will introduce how NLP technologies can be successfully used in various applications, such as machine translation, sentiment analysis, and chatbots. The course also provides an introduction to how Python can be used to build NLP applications.

UNIT 1

BASIC TERMS AND CONCEPTS

STUDY GOALS

On completion of this unit, you will be able to ...

- classify NLP as a field of research.
- distinguish between syntax and semantics.
- explain the most important characteristics of prosodics.
- understand the concepts of grammar.

1. BASIC TERMS AND CONCEPTS

Introduction

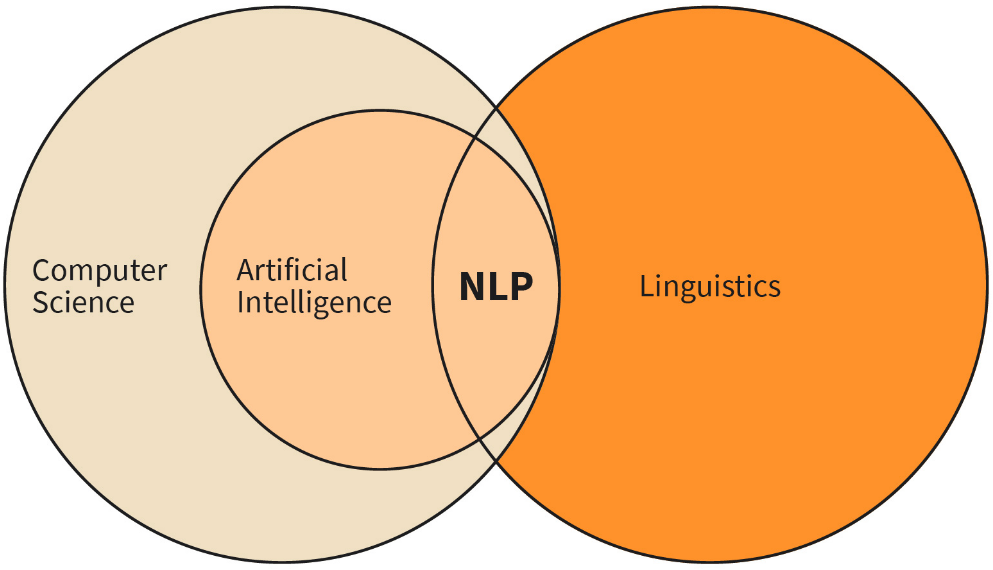
Natural language processing (NLP) deals with the interaction between humans and computers. In the past decades, it has become one of the main drivers for research in the area of artificial intelligence (AI).

In this unit, you will first learn what NLP is in general. This includes the historical developments of NLP as well as an introduction to the Turing test. As NLP has origins at the intersection between computer science and linguistics, the concepts of syntax and semantics will be explained. While syntax deals with the rules how sentences are formed, semantics are about the meaning of a text. Subsequently, you will learn more about the prosodics of speech, i.e., those components of speech that are related to melody, such as pitch or intonation. The unit ends with the most important concepts regarding grammar.

1.1 What is NLP?

Nowadays, natural language processing is one of the most important areas of artificial intelligence (AI). It is an interdisciplinary field with many overlaps between linguistics and computer science. The figure below illustrates how the disciplines are related.

Figure 1: NLP in Relation to Computer Science and Linguistics

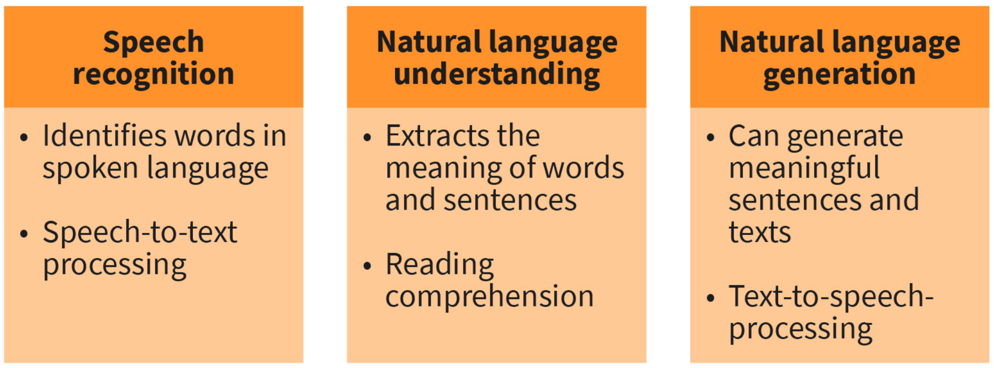


Source: Kristina Schaaff (2023).

The more computers are integrated into our everyday lives, the more important it is to make the interaction between humans and computers more natural. NLP plays a key role in achieving this goal. If a computer is able to interpret and use language in a way that is comparable to human communication, this can help simplify the communication between humans and machines. However, human–computer interaction is not the only use case for NLP. Other areas of application, such as automatic machine translation, automatic text summarization, or even automatic generation of text, are also among current research topics.

In general, there are three major subdomains in NLP: speech recognition, natural language understanding, and natural language generation. What these subdomains contain is summarized in the following figure.

Figure 2: Major Subdomains of NLP


Speech recognition
• Identifies words in spoken language
• Speech-to-text processing
Natural language understanding
• Extracts the meaning of words and sentences
• Reading comprehension
Natural language generation
• Can generate meaningful sentences and texts
• Text-to-speech-processing

Source: Kristina Schaaff (2023).

Speech recognition, natural language understanding, and natural language generation are built on methods that have their origins in the discipline of AI. These three domains form the basis for all other application areas of NLP.

Historical Developments

Early theoretical research in NLP dates back as far as the 17th century. Based on Descartes' concept of the "universal truth," Leibnitz made some first considerations of the representation of the fundamental concepts of knowledge production. Leibnitz believed that if the underlying logical concepts of combining symbols such as letters, words, and sentences are fully understood, it should be possible to generate new thoughts (Schwartz, 2019).

The development of NLP as a technical discipline originated in the 1950s. It was driven by the geopolitical tension between the United States and the former Soviet Union. This tension led to a greater demand for translations from Russian into English and vice versa. Outsourcing the translation to machines seemed to be a good way to tackle this problem (Hutchins, 1997).

The first results of automatic machine translation seemed to be quite promising. However, in the end, it turned out that machine translation was much more complex than originally expected, and the progress fell far short of expectations. In particular, the handling of word ambiguity turned out to be a big challenge. One famous example was the translation of the English sentence “out of sight, out of mind,” which ended up being translated into the Russian equivalent of “invisible idiot” (Hutchins, 1995).

As a consequence, in 1964, NLP technology was classified as hopeless by the US Automatic Language Processing Advisory Committee. The funding of research in this area was temporarily stopped, as it was regarded as neither faster, cheaper, nor as accurate as human translation (Automatic Language Processing Advisory Committee, 1966).

AI winter
An AI winter marked a period where funding, research, and interest in AI technologies were significantly reduced. Those periods were usually the result of expectations that were too high and could not be met.

The decreased interest – and therefore also the decrease of funding – in NLP research can be seen as one of the main causes of the first **AI winter**. It took almost 20 years after the first AI winter for interest in NLP to start to redevelop. The major drivers for the renewed increase in research activities were the increase of computing power, a shift of paradigms in research approaches, and the development of part-of-speech (POS) tagging.

As predicted by Moore’s law, computing power has significantly increased over the years. This increase allowed the use of algorithms that were more computationally intensive, paving the way for new methods in NLP. Moreover, the higher computational power allowed the processing of larger amounts of data to train the algorithms. In combination with the growing amount of electronic literature, which could be used for training, this opened up great possibilities for the improvement of the available algorithms.

Additionally, research approaches shifted from grammatical approaches toward statistical and decision-theoretic models. While early grammatical approaches were trying to address the complexity of everyday language based on the implementation of complex rule-based systems, more recent research started using statistical models, such as decision trees.

Markov model
In a Markov model, the next state is only defined based on the current state and a set of transition probabilities.

Finally, the development of POS tagging was another crucial step toward more robust NLP algorithms. In POS tagging, a given text is divided into smaller units, i.e., sentences, words, or even sub-words. To these units, categories and word functions are added. A text can then be described using **Markov models**. Using this approach, it is no longer necessary to consider the whole history of a text, which drastically reduces the complexity of the developed algorithms.

These developments drastically increased the robustness of NLP algorithms, especially for unseen scenarios and constellations.

NLP and the Turing Test

The mathematician and computer scientist Alan Mathison Turing was one of the very early pioneers in AI. With his research, he created an important theoretical foundation in the area of computer science. One of the major results of his research is the Turing test, which can be used to verify the presence of AI (Turing, 1950).

In the Turing test, a human interrogator uses a computer program to talk to two different conversational partners. One of the conversational partners is a human; the other isa machine. During the conversation, both aim to convince the interrogator that they are a human. A conversational agent passes the Turing test if the interrogator is not able to identify which of their conversational partners is human.

According to Turing, if a machine passes the Turing test, this might mean that the intellectual capabilities of a machine are able to compete with those of the human brain. However, there has been much criticism of the Turing test, as it is focused too much on functionality; the question of whether a machine is aware of its answers is not addressed.

The computer program ELIZA, which was developed by Josef Weizenbaum, was one of the early attempts to pass the Turing test. It can be seen as one of the first approaches to developing a type of system that is nowadays known as a chatbot or conversational AI. ELIZA simulates the conversation with a psychotherapist using rule-based pattern matching in combination with a **thesaurus** to process a sentence from an interrogator and generate an appropriate reply (Weizenbaum, 1966).

Thesaurus
A thesaurus is a structured dictionary that consists of a systematic collection of terms that are thematically related to each other.

In a dialogue, ELIZA will parse the input for a keyword that is contained in the thesaurus to look for synonyms or generic terms for this keyword. Based on the result, a phrase related to the term will be selected, which will often be an invitation to continue talking. If the program does not find an appropriate key phrase, it will reply with a general phrase pretending to not be sure if it understood correctly. A typical conversation with ELIZA could look as follows.

Table 1: Typical Conversation With ELIZA

ELIZA: How are you today? Tell me about your problems.
Human: I am feeling depressed.
ELIZA: Can you explain what made you depressed?
Human: I am not able to handle my workload.
ELIZA: Why is it important to you to handle your workload?
Human: Because I want to pass my NLP exam.
ELIZA: I am not sure if I have understood you correctly.

Source: Kristina Schaaff (2023).

When ELIZA was first published, it generated some remarkable enthusiasm in the AI community. However, the simplicity of Weizenbaum’s approach was quickly recognized, and as expected, ELIZA was not able to pass the Turing test.

A more recent attempt to pass the Turing test was the chatbot “Eugene Goostman.” In 2014, this chatbot seemed to be the first system to be able to pass the Turing test. However, the chatbot used a trick of pretending to be a 13-year-old Ukrainian boy. Being quite young and not being a native English speaker was used as an explanation that the bot made mistakes with the language and did not know everything. Due to this trick, the validity of this experiment was strongly questioned (Masnick, 2014).

1.2 Syntax

Syntax in NLP deals with the study of patterns of how to form sentences and phrases from single words as well as rules for the formation of grammatical sentences. Therefore, syntactical tasks are about features like word boundaries, categories, or grammatical functions. The meaning of a sentence is not considered in syntactical tasks.

Typical tasks in NLP dealing with syntactical information are tokenization and part-of-speech (POS) tagging.

Tokenization

In tokenization, a text is split into individual units, which are called tokens (Russell & Norvig, 2021, p. 876). Those units can, for instance, be sentences, words, or sub-word units. The tokens that have been generated can be seen as discrete elements of a text and can be used to generate vectors that represent that document.

Let us consider the sentence “I like natural language processing.” This sentence could be tokenized into [“I,” “like,” “natural,” “language,” “processing,” “.”].

There are different ways to perform tokenization. Some examples of tokenization are

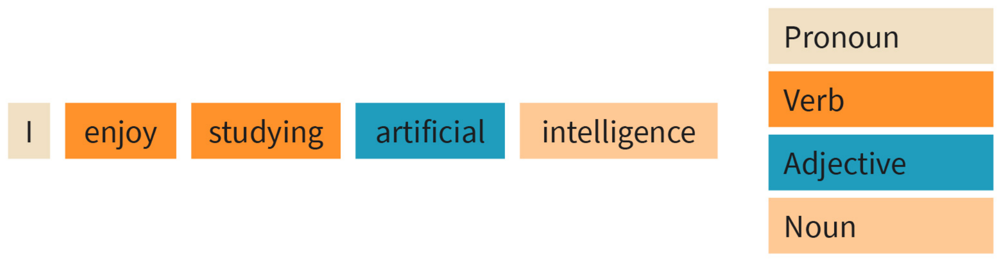
- white space tokenization: This is probably the simplest way to tokenize a text. As the name indicates, it uses the white spaces within a string as word delimiters.
- punctuation-based tokenization: This splits a sentence into word tokens based on punctuation and white space.
- treebank word tokenization: Punctuation and symbols are separated from a text without interference from the textual context. This means that, for example, “aren’t” will be tokenized to [“are,” “n’t”].

Part-of-Speech Tagging

While tokenization is mainly about splitting a text into smaller sub-units, POS tagging adds categories and grammatical word functions – often also referred to as lexical categories or tags – to a given text. This can, for instance, be categories such as “noun,” “verb,” or “adjective.” (Russell & Norvig, 2021, p. 880).

The following figure shows an example of POS tagging.

Figure 3: Example of Part-of-Speech Tagging



Source: Kristina Schaaff (2023).

POS tagging is an important step toward handling syntactic ambiguity, which is a big challenge in NLP. Syntactic ambiguity often makes it difficult to clearly assign a word to a category. If we look, for instance, at the word “present,” this could either be a verb and refer to a presentation or as a noun referring to a gift. Assigning the right category to a word can, therefore, not only help in machine translation; it can also be beneficial in tasks like speech synthesis, as the word “present” will be pronounced differently depending on the POS category. Another commonly used example to illustrate syntactic ambiguity in a sentence is

“Time flies like an arrow.”

There are many different ways to interpret this sentence. Two of those interpretations are:

1. There exists a particular arrow such that every time fly (a hypothetical insect) likes that arrow.
2. Time passes as quickly as an arrow.

In the first interpretation, the word “like” is a verb while, in the second interpretation, “like” is used as a comparative preposition.

1.3 Semantics

Semantics deals with the study of the meaning of language, while the syntax of a text is about providing the rules for a text. To solve semantical ambiguities of a sentence, we need external context, i.e., representations of the meaning of an expression. To create the meaning representations, we can use techniques like semantic parsing. In the following, you will learn more about the reasons why meaning representation is crucial in NLP, and especially in semantic analysis.

Verifiability

Verifiability refers to the ability of a system to verify a statement based on a given model in a knowledge base (Jurafsky & Martin, 2022, p. 311). This can, for instance, be used in tasks like automatic question answering. To illustrate the concept, consider the question

“Is it possible to study artificial intelligence at IU?”

To be able to verify this question, we need a knowledge base that contains information about the possibilities of what can be studied at various universities. The representation could, for instance, look like this:

$$Study(IU, Artificial\ Intelligence)$$

A system can then match the input against the knowledge base and answer the question with “yes.” If there is no appropriate answer in the knowledge base, it either has to respond “no” or reply with a message that it cannot give an answer based on the knowledge base used.

Ambiguity and Vagueness

Ambiguity is a critical issue in NLP, as different sentences can have different meanings depending on the context. Having unambiguous representations of a text is important for most NLP applications, otherwise, a system will not be able to reason over the representation of the text.

If we look at the sentence

“I want to study artificial intelligence at IU.”

most people will immediately know that “artificial intelligence” refers to the topic the speaker wants to study and “IU” to the university. However, the sentence could also be interpreted in the way that the speaker wants to study the topic “artificial intelligence at IU.”

Vagueness is another concept that is closely related to ambiguity. Vagueness occurs if parts of the meaning of a sentence are underspecified (Jurafsky & Martin, 2022, p. 307). The sentence

“I want to study.”

gives enough information to find out that the speaker wants to study in general. However, what and where the speaker plans to study exactly remains underspecified.

Canonical forms

The same meaning of a statement can be formulated with different sentences. Our example sentence could, for instance, also be formulated as

“Artificial intelligence is what I want to learn at IU.”

If canonical forms are used, it means that all inputs that describe the same thing have the same meaning representation in the knowledge base (Jurafsky & Martin, 2022, p. 307).

Using canonical forms can help simplify NLP applications that are related to reasoning, as only one representation of a fact has to be stored in the knowledge database. However, canonical forms will also complicate semantic parsing tasks, as the system has to be able to identify words or sentences that belong to the same thing.

Inference and Variables

Inference in general refers to the ability of a system to draw a conclusion from various inputs based on a knowledge base even though these conclusions are not explicitly represented in the knowledge base. This is done by logically deriving the desired conclusions from the known propositions (Jurafsky & Martin, 2022, pp. 307–308).

If, for example, a person asks the question

“Where can I study artificial intelligence?”

this does not explicitly refer to any particular university. In this case, simple matching does not work, as there is no specific university named. To answer the request, the system could use a representation such as

$$Study(x,\ Artificial\ Intelligence)$$

where x is a variable from the knowledge base, which can be replaced in a way that the answer of the system will match the question. Being able to handle variables and use them for logical inference is important for systems that are able to handle open questions like the one in the example.

Expressiveness

The last important property in meaning representation is expressiveness. This means that a system has to be able to handle a large range of topics, ideally for every utterance. One approach used to handle this challenge is **first-order logic**.

First-order logic
This can be used as a meaning representation language using a set of “atoms” that can be used to compose larger units, such as sentences.

1.4 Prosodics

Prosodics refers to properties of speech that cannot be derived from segmental phonemes. While phonemes are units of sound that distinguish one word from another, prosodics refers to larger units of speech and includes elements such as loudness, pitch, or duration, which are also called suprasegmental properties of speech (Nooteboom, 1997, p. 640). The term “prosody” comes from the Greek word “prosodia,” which can either mean song or syllabic accent (Gibbon, 2017, p. 4).

Prosody is an essential part of speech, as it can transport information that is not encoded in the pure sequence of words, such as irony, sarcasm, or the emotion of a speaker in general. It can, therefore, help disambiguate aspects of a text that are not reflected in the transcripts. Taking prosodics into account can have a huge impact on the performance in speech recognition tasks (Waibel, 1988).

There are many prosodic features of speech that have a huge influence on how a word or sentence can be interpreted. However, there are huge differences in prosodics dependent on the underlying language. Therefore, we will limit the following illustrations to English.

Prosodic Prominence

Fundamental frequency
the frequency of vibration of the vocal folds

There are several ways of emphasizing words in languages like English. A word could, for instance, be said slower or louder, or the **fundamental frequency** F0 could be varied, making a word's pitch higher or more variable.

Pitch accents

Prominence can be represented by using pitch accents, which are often also referred to as tone. Pitch accents refer to a specific melody that is applied to a word to add morphological functions or make phonemic distinctions (Gibbon, 2017, p. 3). Syllables or words that are more important can be emphasized by accenting them with pitch.

Stress

Stress refers to emphasis that is put on a sound, syllable, or word while speaking. Stress positions in English are usually indicated by pitch accents. If stress is put on single syllables of a word, this is referred to as lexical stress. As the following example shows, the position at which a word is stressed can change the meaning of a word.

Figure 4: Stress in Prosody

Paul got a **pre**sent from his wife.
Paul will pre**sent** the results to his boss.

Source: Kristina Schaaff (2023).

In the same way, the stress on single words of a sentence can also change the meaning of the whole sentence.

Prosodic Phrasing

In spoken sentences, some of the words will be grouped together naturally, while there will be noticeable breaks between other words. Breaking up utterances into meaningful chunks is important to being able to understand a sentence, especially as prosodic and syntactic structures are often correlated (Bennett & Elfner, 2019).

Typical signals for prosodic boundaries are, for instance, a final fall or rise, phrase- or utterance-end lengthening, or a continuation rise. In text-to-speech tasks, automatic prediction of prosodic boundaries is an important field of research, as it will help to make the results easier to understand.

Intonation

The intonation of a voice refers to variations in the fundamental frequency F0, as well as variations of the speed of an utterance. It includes rhythms and melodies that occur in spoken language and normally refers to constructs that go beyond single words, such as phrases or sentences. Intonation includes higher and lower melodic patterns, as well as acceleration and deceleration of rhythmic patterns (Gibbon, 2017, p. 4). A rise in pitch at the end of a sentence can, for instance, be seen as an indicator for a question, whereas a final drop can indicate declarative information.

The following example shows how different intonations and accenting can affect the meaning of a sentence.

Figure 5: Example of How Prosody Can Affect the Meaning of a Sentence

Yes, if it is necessary, I will bring my laptop.
Yes, if it is necessary. I will bring my laptop?
Yes? If it is necessary, I will bring my laptop.

Source: Kristina Schaaff (2023).

1.5 Grammar

Natural languages are built in a multi-level way: they are based on an alphabet, i.e., a finite number of characters, which are then formed into single words; words can be formed into sentences using a grammar. Therefore, a grammar is used to formally describe a language.

Formally, a language can be described as follows:

An alphabet Σ is a finite number of symbols σ. A sequence $w \in \Sigma^*$ of symbols from the alphabet is called a word. A set $L \subseteq \Sigma^*$ of words is called a language. The grammar of a language L is a set of rules that can be used to generate the sentences of the language L.

Formal languages can be categorized based on the way a language is composed using a grammar. However, natural languages normally come with a lot of exceptions and are therefore too complex to be completely described by a formal grammar. There is normally no hard boundary between sentences that are allowed or not allowed, and there is no definitive tree structure for the sentences. Nevertheless, hierarchical structures are very important in natural language to reduce ambiguity and to increase understanding of words and sentences.

Noam Chomsky was the first to propose a formal definition of grammars. A formal grammar consists of the following elements (Chomsky, 1956):

- a finite number of variables V, which are called nonterminals
- a start symbol $S \in V$
- an alphabet Σ, i.e., a finite number of symbols, which are also called terminals. A character cannot be a terminal and a nonterminal at the same time
- a finite number of production rules, which have the form $l \rightarrow r$, where $l \in (V \cup \Sigma)^* V (V \cup \Sigma)^*$ and $r \in (V \cup \Sigma)^*$

Nonterminals are symbols that are used to produce words and which have to be replaced by terminals in the end. Typical nonterminals would be elements like <sentence>, <subject>, <verb>, <object>, <article>, and <noun>. They are often indicated by < >. The words of a language consist only of terminals. As an example, we could define the following alphabet: {“the,” “cat,” “eats,” “mouse”}

Rules describe, from left to right, how sentences can be formed using the rules. The * is the Kleene star operator and means that the symbols in the brackets can be repeated zero or more times.

For our example, the set of rules could look like those in the following table.

Table 2: Example Grammar

<sentence>	$\rightarrow$	<subject><verb><object>
<subject>	$\rightarrow$	<article><noun>
<object>	$\rightarrow$	<article><noun>
<article>	$\rightarrow$	the
<noun>	$\rightarrow$	cat, mouse
<verb>	$\rightarrow$	eats

Source: Kristina Schaaff (2023).

We are now able to build a sentence using the above grammar:

<sentence>

<subject><verb><object>

<subject><verb><article><noun>

<subject><verb><article>mouse

<subject><verb> the mouse

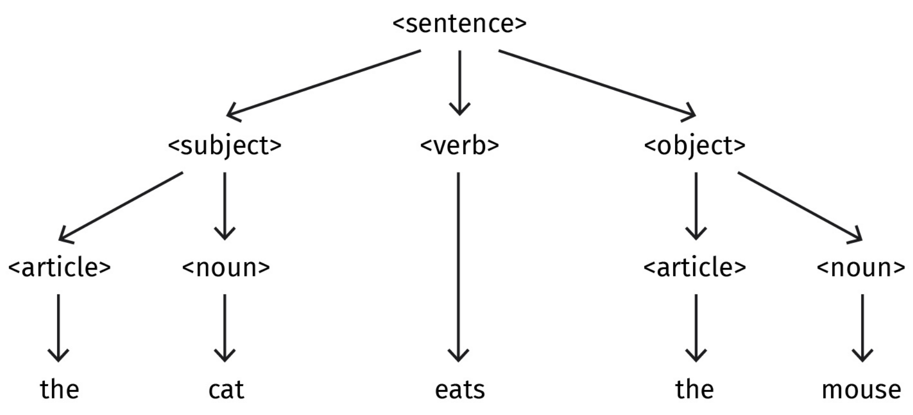
<subject>eats the mouse

<article><noun>eats the mouse

<article> cat eats the mouse the cat eats the mouse

The example above could also be illustrated using a parse tree.

Figure 6: Parse Tree



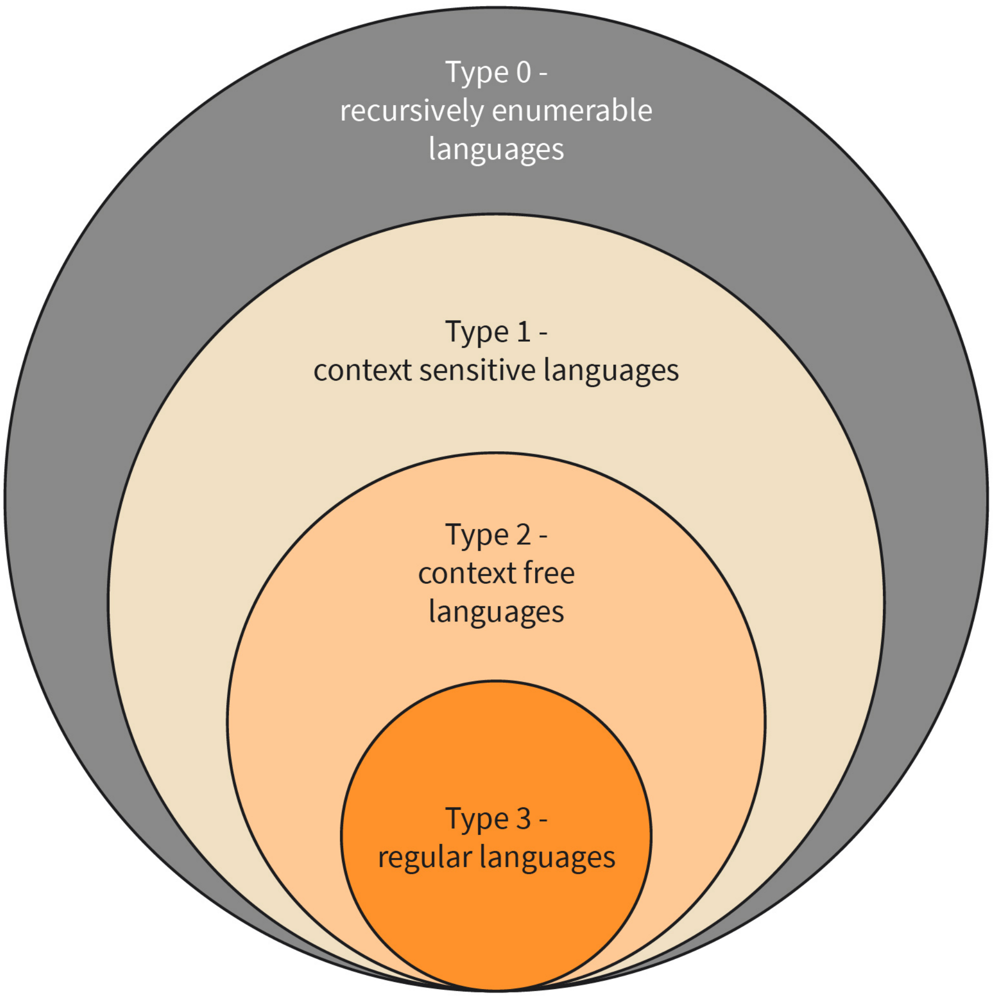
Source: Kristina Schaaff (2023).

The simple grammar we used in our example allows us to build simple sentences such as the one in the example. However, we can also build sentences that are wrong, such as “the mouse eats the cat.”

For this reason, in formal grammars, there are various limitations on how rules can be constructed. Chomsky developed a hierarchy – the Chomsky hierarchy – to describe different grammars of formal languages, which increase in complexity.

The Chomsky hierarchy ranges from the Type 0 grammar – also referred to as a phase structure grammar – that is recursively enumerable over context-sensitive (Type 1) and context-free (Type 2) grammars to regular grammars (Type 3). Each class is contained in the next class, as illustrated in the figure below.

Figure 7: Chomsky's Types of Grammars



Source: Kristina Schaaff (2023), based on Chomsky (1956).

Type 0 grammars can be used to generate the most general language class. It includes all formal grammars without any additional restrictions. Type 3 grammars, however, are the most restrictive grammars and are included in all other types of grammars.

SUMMARY

Natural language processing is an interdisciplinary field of research at the intersection of computer science and linguistics. It can be divided into the three subdomains: speech recognition, natural language understanding, and natural language generation. Early research dates back as far as the 17th century.

In NLP, syntax deals with the study of patterns – how to form sentences and phrases from words – while semantics deals with the meaning of language. Elements such as loudness, pitch, and duration of speech are referred to as prosodics. Grammar can be used to formally describe the structure of a language.

UNIT 2

LANGUAGE AND SPEECH

STUDY GOALS

On completion of this unit, you will be able to ...

- illustrate how sound is produced in the human vocal apparatus.
- understand the process of speech production.
- explain the concepts of phonetics.

2. LANGUAGE AND SPEECH

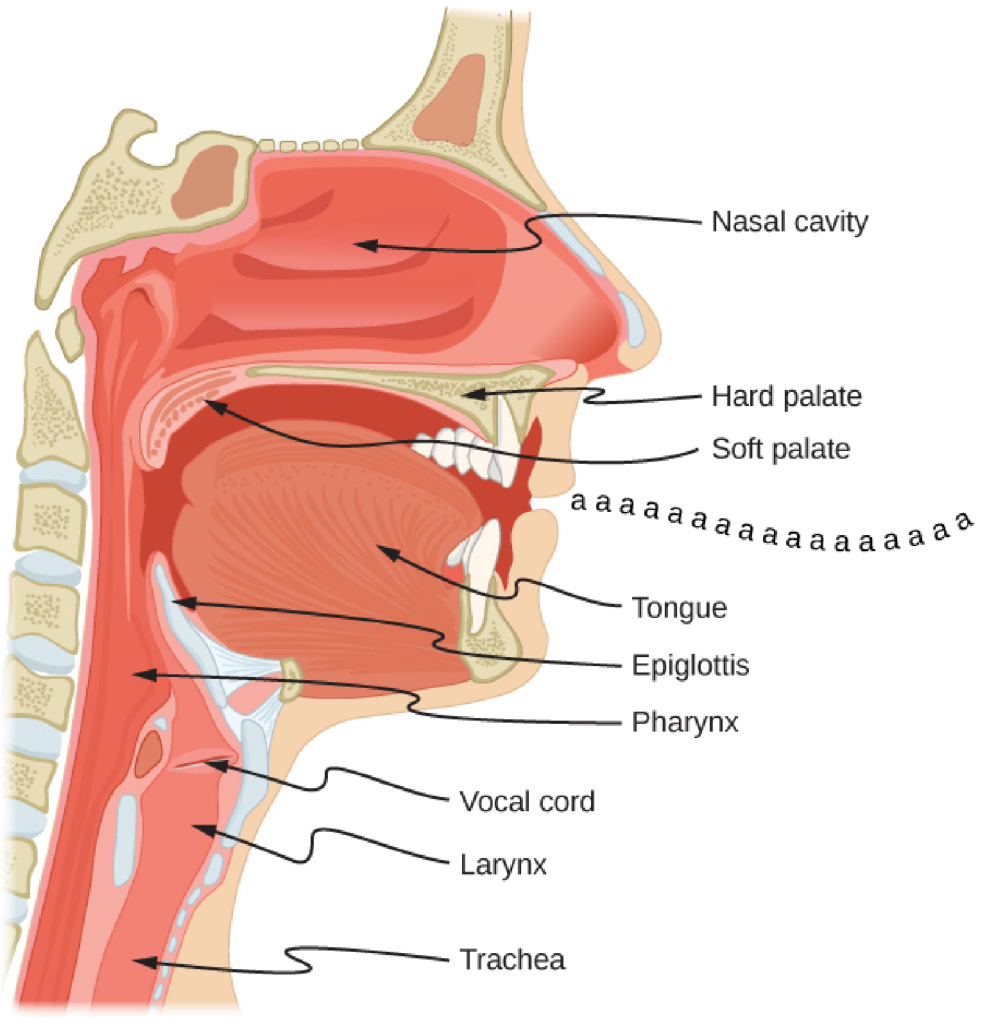
Introduction

This unit deals with the production and perception of language and speech. The process of speaking involves several parts of the body, beginning with the lungs, which are responsible for respiration, ranging from the voice production in the voice box to the articulation of sounds in the mouth. Therefore, you will first get an introduction to how the human vocal apparatus is structured and how the articulation of sounds works. After that, we look at how language is created. The focus of this part will be on the cognitive processes that are involved in the process of speaking. The last part of the unit is about phonetics, which involves the articulation of phones in the mouth; the process of how sounds can be digitalized; and finally, how the process of hearing functions.

2.1 Human Vocal Apparatus

When we speak, we use our lungs to produce an airstream, which passes through the mouth and nose. Viewed as a whole, the vocal tract can be divided into three elements: respiratory, vocal, and articulating elements. The respiratory elements are responsible for producing the airstream that passes over the vocal elements. The vocal elements will then produce the voice, which in the last step is shaped by the articulating elements. The figure below shows the human vocal tract.

Figure 8: The Vocal Tract



Source: OpenStax University Physics (2016). CC BY 4.0.

We will now go more into detail about how the components work.

Respiratory Mechanisms

The abdominal muscles, diaphragm, rib cage, and chest muscles cause the movement of the lungs. When we breathe, this happens in two phases:

1. Inhalation: The diaphragm and the intercostal muscles contract. This pulls down the lungs and the rib cage and fills the lungs with air as the volume of the lungs increases.
2. Exhalation: The intercostal muscles and the diaphragm relax. This causes the ribs to collapse and decreases the ribcage capacity. Therefore, the air is pushed out of the lungs again.

As the two steps of breathing illustrate, the lungs are not able to inflate by themselves. Instead, they are moved by the diaphragm and the intercostal muscles, the accessory muscles, and the abdominal muscles (Ratnovsky et al., 2008, pp. 82–83).

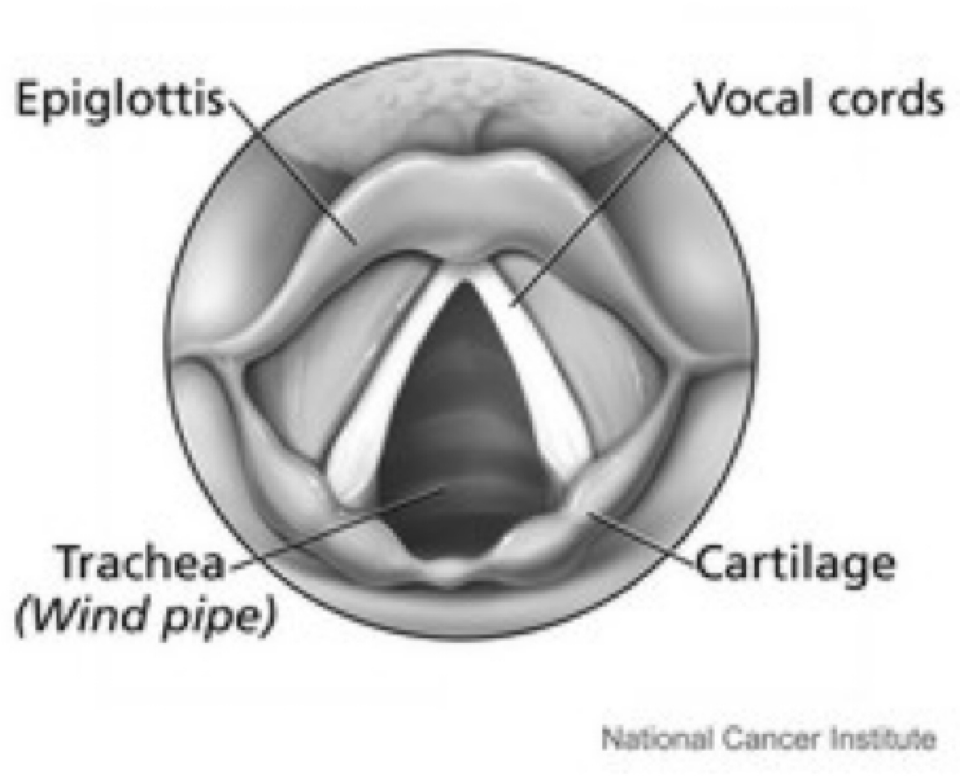
Voice Production

Once the airstream has been produced by the lungs, it will then flow through the windpipe (trachea), past the voice box (larynx) and the back of the throat (pharynx).

The voice box (larynx) contains two small muscles – the vocal folds, or vocal cords. Between the vocal cords, there is a small space called the glottis. Depending on how close the cords are together, they will vibrate when air passes through. This vibration produces sound waves.

The figure below shows a top view of the larynx. When we eat, the epiglottis closes the larynx to avoid liquids or food getting into the trachea and the lungs. The larynx is protected by the laryngeal cartilages.

Figure 9: Larynx (Top View)



Source: The National Cancer Institute (2003). Public Domain.

The closer the vocal folds are together, the more they will vibrate when air passes through. The vocal folds are important for voiced sounds, such as [b] or [g], and for English vowels. When voiced sounds are produced, the vocal folds will be close together, while for unvoiced sounds, such as [p], [t], or [k], they will be far apart and, therefore, will not vibrate.

Articulation

Once the sound waves have been generated, they will be modulated in the vocal tract. The vocal tract acts as a resonator and filter for the sound created by the vocal folds in the larynx. The pharynx, nasal cavities, and mouth act as resonators to amplify or attenuate the frequencies. The sound will then be modulated in the nasal and oral cavities.

The oral cavity consists of several structures that help shape the air to form different types of sounds. The roof of the mouth consists of the hard palate and the soft palate. Moreover, the positions of tongue, teeth, and lips play a significant role when it comes to articulating different sounds.

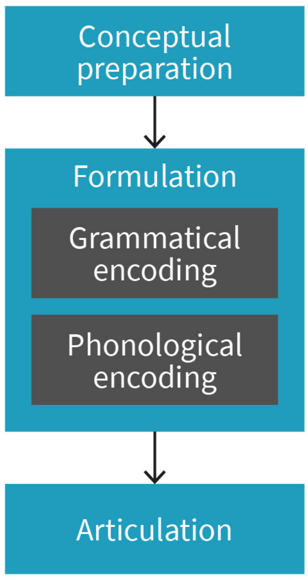
The nose is only required for nasal sounds, such as [m] and [n], while most sounds are formed in the oral tract. Nasal sounds are produced when the airstream is directed outward through the nasal cavity instead of the mouth.

2.2 Speech Production

Before words are articulated in the mouth, a number of cognitive processes are involved. The research area of speech production deals with the cognitive processes that are involved when thoughts are transformed into speech (Schriefers & Vigliocco, 2015, p. 255).

According to Levelt (1999), the process of speech production can be broken down into three levels, which are illustrated in the figure below.

Figure 10: Levels of Speech Production



Source: Kristina Schaaff (2023).

In the conceptual preparation phase, the speaker decides which information should be transmitted and transforms the thoughts into a message that can be verbalized. As a result, we will have a preverbal message (Levelt, 1999, pp. 226–227).

Once we have the intention to speak, the brain will select the relevant information from memory that is required to create the preverbal message. In this process, we decide what we want to say.

In the formulation phase, we give a verbal shape to the elementary messages from the conceptualization phase. Words are selected from the vocabulary and put into a correct syntactic order. The process of formulation is done in two steps:

1. Grammatical encoding: This is the selection of the content and forming the structure. A lemma that matches the preverbal message will be selected. The lemma represents the meaning of what we want to say but does not include any specific sounds.
2. Morpho-phonological encoding: This involves transforming the words into syllables based on **morphological** and **phonological** structures (Levelt, 1999, pp. 229–230).

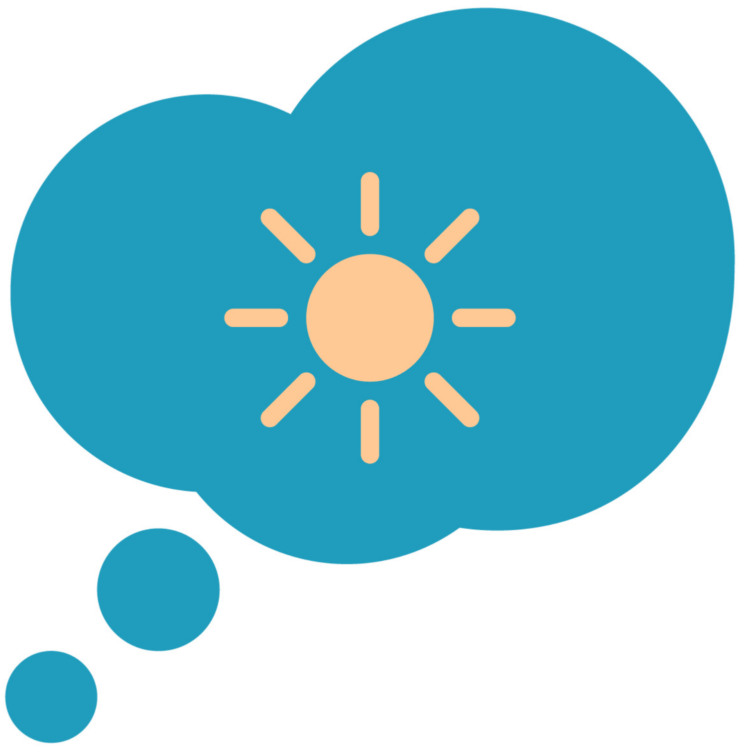
Morphology
how words are formed

Phonology
how sounds are organized

The third step deals with the articulation of the message. In this phase, the syllables that are required for the words are produced and assembled. When speaking, we receive the syllable programs from the syllable memory, which is also referred to as mental syllabary (Levelt & Wheeldon, 1994, p. 239). After this, the speech sounds can be produced by the vocal tract.

Let us look at a practical example to understand the process. If someone asks how the weather will be tomorrow, in the conceptualization phase we will make a concept of what we will reply. The concept for our reply could look like the following figure.

Figure 11: Example of Conceptual Preparation



Source: Kristina Schaaff (2023).

In the formulation phase, the appropriate vocabulary for our concept will be selected and transferred into a syntactically correct sentence, which could, for instance, be

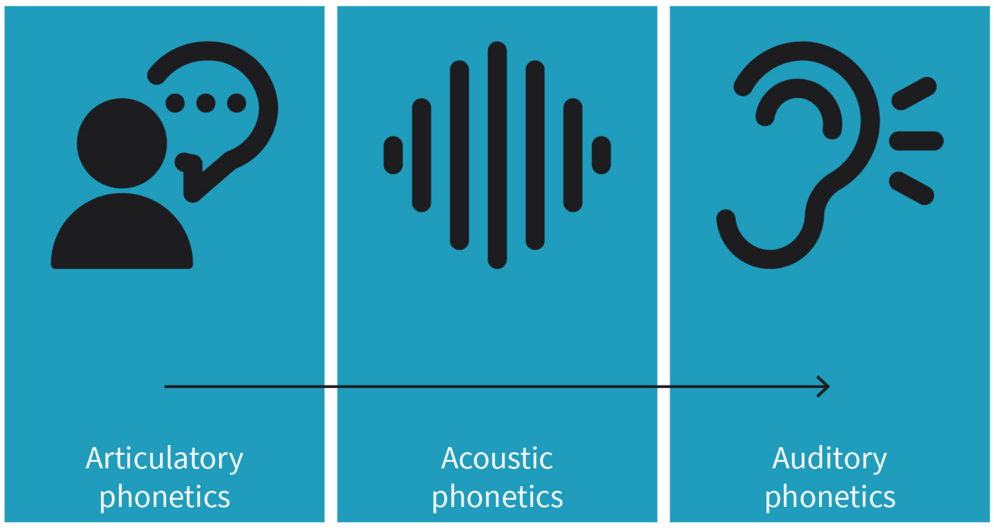
“The sun will be shining.”

Additionally, the syllables that are required to articulate the answer will be selected. Finally, in the articulation phase, the answer is transferred to speech.

2.3 Phonetics

Phonetics deals with the production and perception of sound. It can be divided into three subdisciplines: articulatory, acoustic, and auditory phonetics. The figure below illustrates how these three subdisciplines link together.

Figure 12: Subdisciplines of Phonetics



Source: Kristina Schaaff (2023).

Articulatory phonetics deals with the process of how humans produce speech by the interaction of the physiological structures. Acoustic phonetics refers to physical aspects of the speech sound, such as amplitude or frequency. Auditory phonetics addresses the perception and understanding of linguistic signals.

Before we dive deeper into the three subdisciplines of phonetics, we want to start with some basic terms and definitions.

Phones and Phonemes

Phonetics is based on phones. A phone can be seen as the smallest segmental unit of speech, no matter whether or not it is important for the meaning of a word. Phonemes, however, belong to the area of phonology and are the smallest sound unit that distinguishes two words from each other in a given language.

To differentiate between phones and phonemes, the transcription of speech sounds into phones is written in square brackets (e.g., [p] or [b]), while phonemes are indicated with slashes (e.g., /p/ or /b/).

The most commonly used system for phonetic notation (phones) is the International Phonetic Alphabet (IPA), which provides a standardized written representation of speech sounds. For American English, the ARPAbet provides a simpler phonetic alphabet based on ASCII symbols to represent the subset of the IPA that is required to transcribe speech in American English (Jurafsky & Martin, 2022, p. 527).

The table below illustrates some examples of the differences between IPA and ARPAbet.

Table 3: Examples of Differences Between IPA and ARPAbet

Word	ARPAbet symbol	IPA symbol
thin	[th]	[θ]
sing	[ng]	[ŋ]
dish	[sh]	[ʃ]

Source: Kristina Schaaff (2023).

If not otherwise noted, the following transcriptions will be based on the ARPAbet.

Articulatory Phonetics

In articulatory phonetics, sounds are divided into three different categories:

1. Vowels are phones that are normally voiced and longer and louder than consonants.
2. Consonants are produced by blocking the airflow. They can either be voiced or unvoiced.
3. Semivowels, like [y] or [w], are in between vowels and consonants; they are usually voiced but shorter and less syllabic than vowels.

Vowels

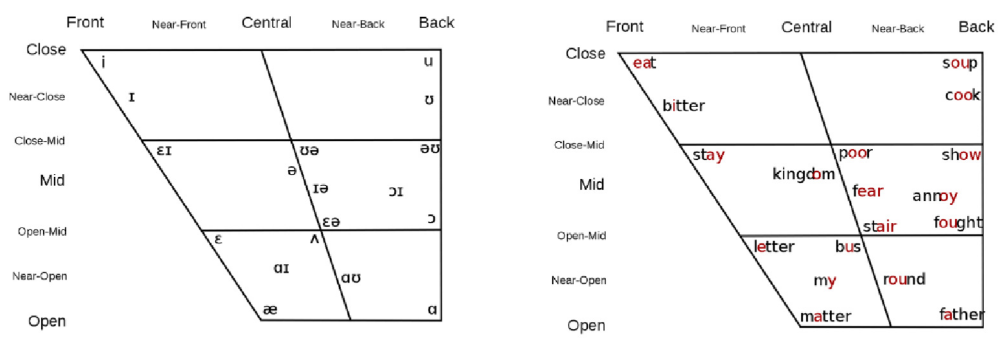
The sound of a vowel is characterized by the position of the articulators, namely the glottis, pharynx, velum, lips, and tongue. For vowels, the vocal tract is open, which means that the airflow is not obstructed. The most relevant parameters to characterize vowels are as follows (Jurafsky & Martin, 2022, p. 530):

- height of the highest part of the tongue
- “frontness” or “backness,” depending on whether the highest part of the tongue is located more to the front or to the back
- shape of the lips (rounded or not)

For some vowels, the position of the tongue changes while they are produced. These vowels are called diphthongs, which comes from the Greek and means having two different sounds.

The following figure illustrates how vowels and diphthongs can be characterized depending on the shape of the mouth (left) and gives some practical examples (right).

Figure 13: IPA English Vowels and Diphthongs with Examples



Source: TheCPMills (2018). CC BY-SA 4.0.

Consonants

The sound of consonants strongly depends on the place where the airflow is blocked.

For labial consonants, the lips are involved. Consonants, like [p] or [b], where both lips come together, are called bilabial, while labiodental consonants, such as [v] or [f], are produced by pressing the bottom lip against the upper teeth. Finally, for linguolabial consonants, the tongue and the upper lip are involved (Ladefoged & Maddieson, 1996, p. 16).

Dental consonants are produced by pressing the tip or blade of the tongue against the teeth. Dental consonants include, for example, the [th] in the word that.

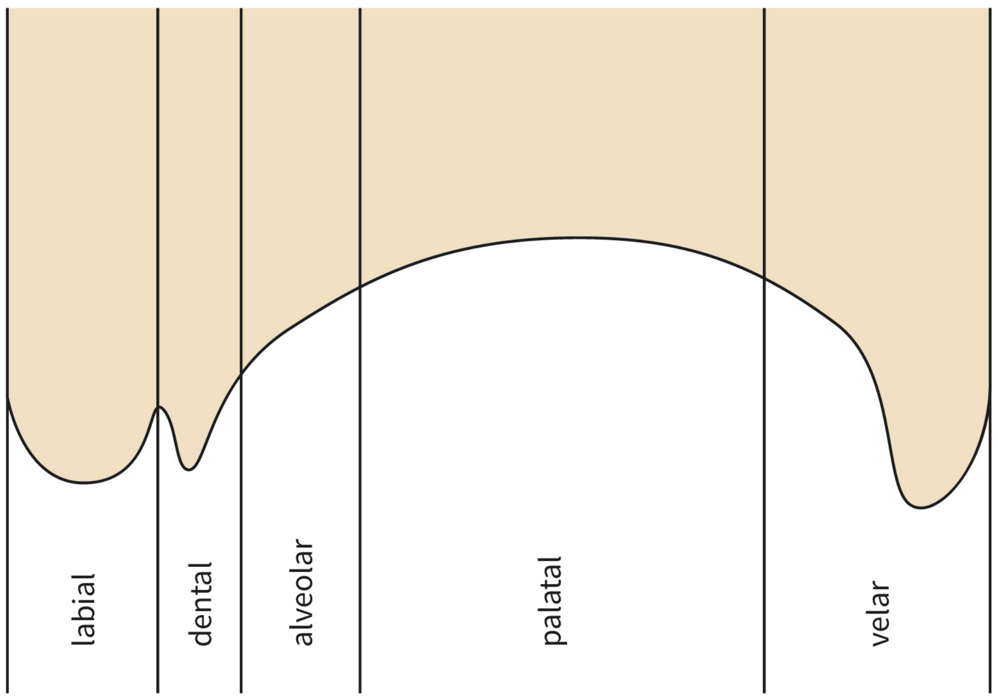
Alveolar consonants are produced in the part of the mouth that is located behind the upper teeth. Phones like [s] or [z] are made when the tip of the tongue is placed against the alveolar ridge.

Palatal sounds are produced at the roof of the mouth, which is also called the palate. If the front of the tongue is placed close to the palate, this will produce sounds like the [y] in yak. Palato-alveolar phones, such as the [sh] in the word ship, can be produced by pressing the tongue against the back of the alveolar ridge.

Velar sounds include sounds like [k] or [g] and are produced by using the tongue to block the air in the velum, which is located at the roof of the mouth at the back. Velars often occur in coarticulation with vowels, as both are produced using the tongue body. Similar to the vowel space, velar consonants can therefore be categorized into front, central, and back velars (Ladefoged & Maddieson, 1996, pp. 33–34).

The figure below illustrates where the positions in the mouth are located.

Figure 14: Regions of the Mouth



Source: Kristina Schaaff (2023).

In addition to the aforementioned places, glottal consonants like [q] are produced by closing the gap between the vocal folds. Glottal stops can also occur between vowels.

For consonants, it is not only important to know the place of articulation but also the way in which the vocal tract is modified, narrowed, or closed (Ladefoged & Johnson, 2011, p. 14). The most important ways in the English language will be explained in the following.

- Stops or plosives occur when the air flow is completely stopped for a brief time. This means that not only is the oral vocal tract blocked but the nasal air flow is stopped. The stop itself– also called a closure – is completely silent. The sound is produced when the air is released. Plosives can be either voiced (e.g., [b], [d], or [g]) or unvoiced (e.g., [k], [p], or [t]).
- When nasals are produced, the oral tract is completely closed, and air can only pass through the nose. To produce nasal sounds, the velum is lowered to direct the air into the nasal cavity (Jurafsky & Martin, 2022, p. 530). Nasals include the consonants [m] and [n].
- Fricatives – also referred to as spirants – are produced by a turbulent airflow (frication). To produce this sound, the vocal tract is partly blocked. The sound of a fricative depends on the area in which it is produced. To produce labiodental fricatives, such as [f] and [v], the lower lip is pressed against the upper teeth. Dental fricatives like [th] are produced between the tongue and teeth (Ladefoged & Johnson, 2011, p. 14). The most common fricatives in English are [s] and [z], which are high-pitched fricatives where the

tongue guides the airflow toward the teeth. Those fricatives are also called sibilants (Jurafsky & Martin, 2022, p. 530). If a stop is directly followed by a fricative, like the [ch] in child, the combination is called affricate.

- Approximants are produced by bringing the articulators close together but still not so close as to cause a turbulent airstream. An example of an approximant is the [w] in wood, where the back of the tongue is moved close to the velum.
- If the tongue is moved quickly against the top of the mouth, this is called a tap or flap.

The combination of the manner and the place of articulation will yield a unique consonant depending on how the sound is shaped.

Acoustic Phonetics

Acoustic phonetics deals with the description of human sounds as a combination of waves. These waves are used to transfer the sound from a speaker to a listener. The waves can be modeled as periodic functions. Periodic functions repeat after a certain distance or time.

The time T between two oscillations is called the wavelength or period. From the period, the frequency f of a wave can be computed as the number of oscillations within one second:

$$f = \frac{1}{T}$$

A sound with a frequency of 150 Hertz (Hz) means that the sound repeats itself 150 times within one second. The frequency of a sound wave reflects the pitch of a sound. The higher the frequency of a sound, the higher the pitch, and vice versa.

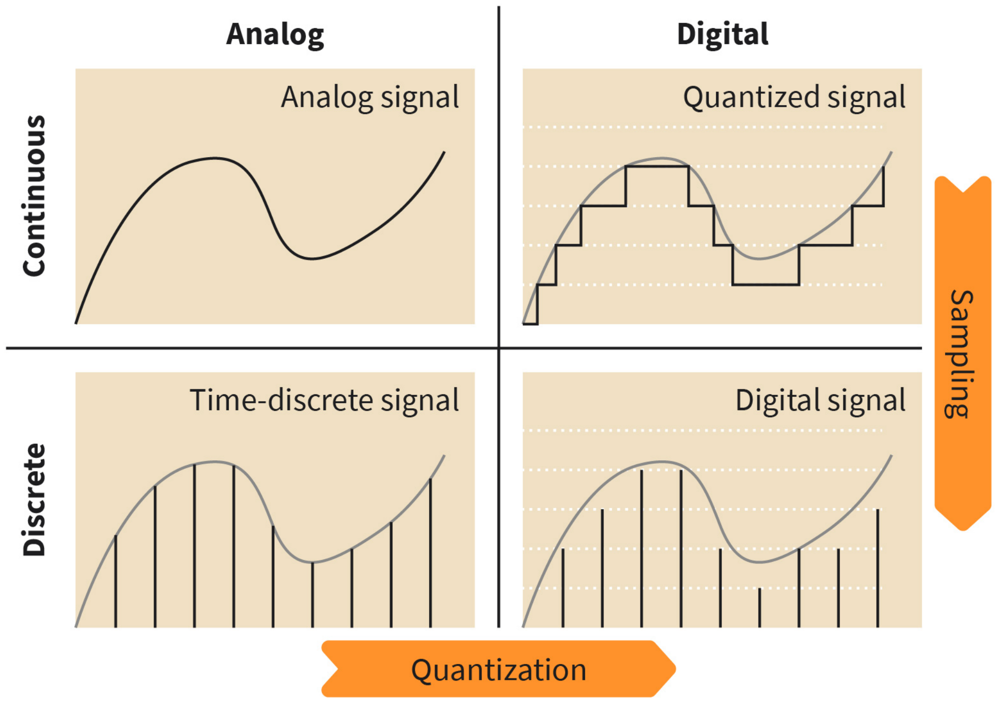
Another important characteristic of a sound is its amplitude A. The amplitude indicates the intensity of a sound, i.e., how loud it is. If the amplitude is high, the produced sound will be loud, while quiet sounds are characterized by a low amplitude.

Digitalization of speech

To process a sound using a computer, the speech signal has to be transformed into a signal that can be represented in a form the computer can understand. This is done by quantization and sampling.

Quantization transfers a signal from an analog to a digital signal, while sampling converts a continuous signal into a discrete signal. In the figure below, the process of transforming an analog signal to a digital signal is illustrated.

Figure 15: Analog-to-Digital Conversion of a Signal



Source: Kristina Schaaff (2023).

When a signal is sampled, the amplitude is measured at regular intervals. The number of samples per second is referred to as the sampling rate or sampling frequency. It is important to select a sampling rate at least twice as high as the maximum frequency that is to be reconstructed (Nyquist–Shannon sampling theorem). The maximum frequency is also referred to as the Nyquist frequency and can be computed as follows:

$$f_{nyquist} = \frac{1}{2} \cdot f_{sample}$$

When the sampling rate is too low, aliasing will occur, which produces a distortion from which it is not possible to recover the original signal. For quantization, the y-axis is typically partitioned into a fixed number of equally sized intervals. The number of intervals n is typically a power of 2, such as $2^{16} = 64436$. The amplitude of the quantized signal can only have values that are integral numbers of intervals. Of course, the quantization adds noise to the signal, i.e., induces errors. For a signal f that is in the range from f_{min} to f_{max}, the average quantization error can be computed using the following equation:

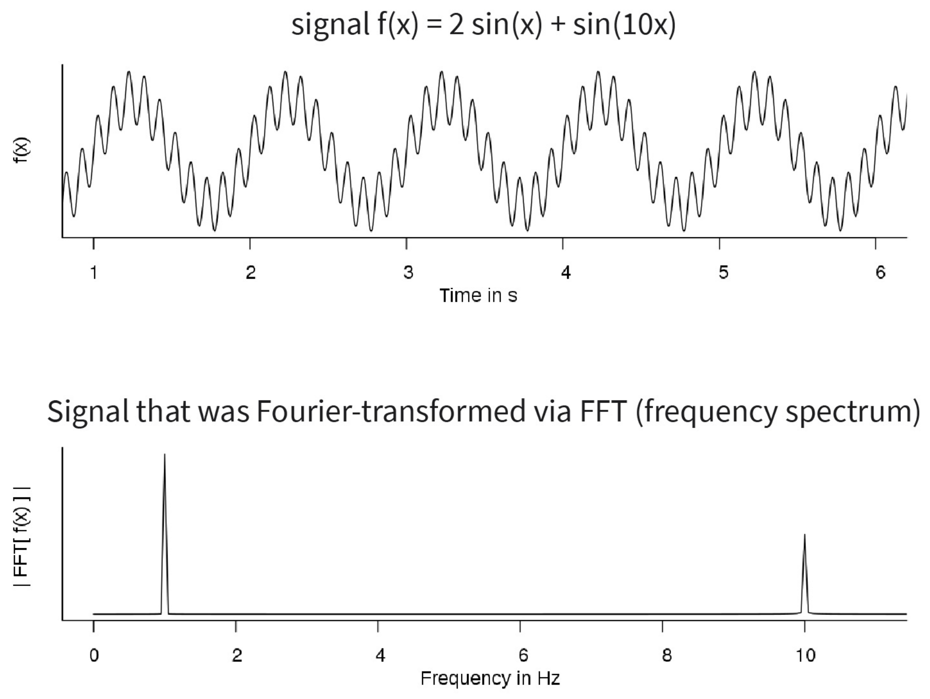
$$e = \frac{f_{max} - f_{min}}{2n}$$

Time and frequency domains

Signals can either be analyzed in the time or in the frequency domain. Analysis of features like the amplitude or the pitch of a signal can be interpreted directly from the time domain. However, many algorithms in speech recognition are based on features from the frequency domain. Using methods like the **Fourier transform**, a signal can be transferred from the time into the frequency domain. In the frequency domain, the waves from different frequencies will be summed up. This representation is called a spectrum.

Fourier transform
The Fourier transform is a mathematical method that can be used to decompose a function from the time domain into the frequency domain.

Figure 16: Representation of a Signal in Time and Frequency Domains



Source: Accountalive (2021). CC0 1.0.

The figure shows an example of a signal that is mapped from the time to the frequency domain using the fast Fourier transform (FFT) method. The signal on the top is generated by combining two sine functions: one sine function with an amplitude of two and a frequency of 1 Hz and another with an amplitude of one and a frequency of 10 Hz. The bottom signal shows the same signal in the frequency domain. As the amplitude of the 1 Hz signal is two times higher than the amplitude of the 10 Hz signal, the peak at 1 Hz is also two times as high as the peak at 10 Hz.

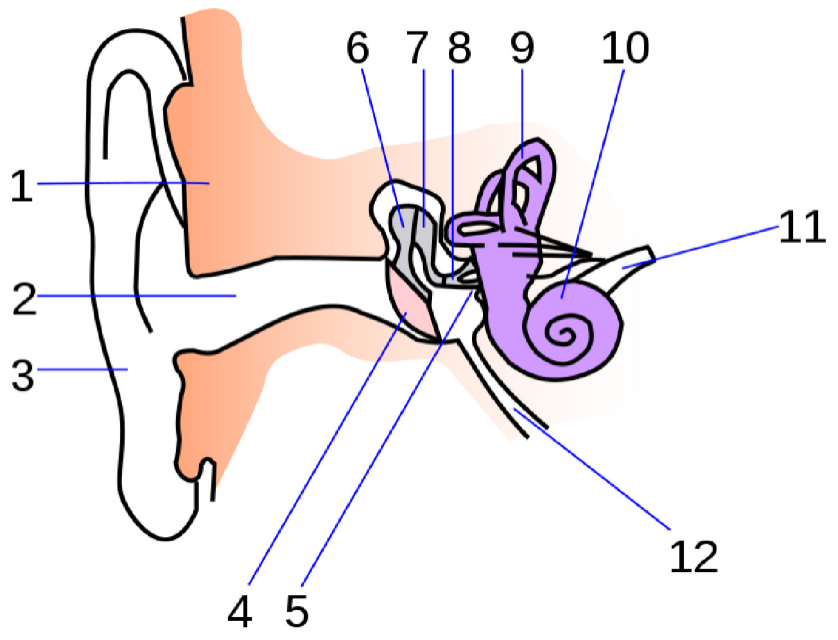
Auditory Phonetics

Auditory phonetics deals with the perception and processing of sounds.

When we hear a sound, we will perceive it with our ears. The ear consists of three parts: the outer ear, the middle ear, and the inner ear.

The anatomy of the human ear is shown in the figure below.

Figure 17: Anatomy of the Human Ear



Source: Surachit (2003). CC-BY

The outer ear is located outside of the skull (1) and includes the ear canal (2) and the pinna (3). The most important function of the outer ear is to encode the spatial and temporal information of the sound it collects.

The middle ear consists of the eardrum (*tymphanum* – 4), the *fenestra ovalis*, and three tiny bones (6, 7, 8), which are also called auditory ossicles. When the eardrum is moved by sound waves, it will move the hammer (*malleus* – 6). The hammer will then transmit the sound via the anvil (*incus* – 7) to the stirrup (*stapes* – 8), which will transfer the sound to the membrane of the *fenestra ovalis*, which is the opening to the vestibule of the inner ear.

The part of the inner ear that is involved in hearing is called the cochlea (10). It is located in the bony labyrinth (9). The cochlea converts the sound into nerve impulses, which can then be processed by the auditory nerve (11).

A healthy human is normally able to perceive frequencies between 20 Hz and 20 kHz. However, the frequencies we are able to hear are strongly influenced by factors like age and gender. The older we get, the lower the highest frequencies we are able to hear, and the highest frequency can get as low as 12 kHz.

SUMMARY

Speech is produced using the vocal apparatus. The respiratory elements produce an airflow that passes over the vocal elements. These elements then produce the voice, which can afterward be shaped by the articulating elements of the vocal apparatus.

Before speech can be produced, cognitive processes transform thoughts from the brain to speech.

Phonetics deals with the production and perception of sound. To transcribe phonetics, systems like IPA and ARPAbet can be used. Articulatory phonetics refers to the process of how humans produce speech by the interaction of the physiological structures. Acoustic phonetics deals with the physical aspects of speech sound, such as amplitude or frequency. Auditory phonetics is about the perception and understanding of linguistic signals.

UNIT 3

CHALLENGES IN NLP

STUDY GOALS

On completion of this unit, you will be able to ...

- describe what is important about data for NLP systems.
- compare different NLP systems.
- explain the domain challenges for NLP systems.
- understand the difficulties of multilingual applications.

3. CHALLENGES IN NLP

Introduction

In the past decades, algorithms for natural language processing (NLP) – be it for speech recognition, natural language understanding, or natural language generation – have become tremendously powerful. However, human language is extremely complex and subject to constant changes over time. This poses a lot of challenges to NLP, which will be addressed in this unit.

We will start with an introduction to the most important challenges that arise when collecting data for NLP applications. Having a high-quality data corpus from which to build a model is key to developing new systems. We will also discuss how NLP systems can be evaluated. The evaluation of NLP systems is an important step toward finding out which model works best for a certain task and comparing different models. Moreover, good evaluation metrics can also help optimize a given model's parameters.

Another important challenge in NLP is different application domains, as language may vary depending on the domain, but there are also other factors that can influence language, such as social groups or dialects. Finally, you will learn about the challenges in NLP when building multilingual applications.

3.1 Data for NLP

NLP can be divided into three different subdomains:

1. Speech recognition deals with the recognition of words and sentences from spoken language and is, therefore, also referred to as speech-to-text processing.
2. Natural language understanding is about the identification of the meaning of words and sentences.
3. In natural language generation, text is transformed into meaningful speech – also called text-to-speech processing.

All subdomains require data to train and develop a model that suits the respective NLP task.

In the early years of NLP, algorithms were limited by data storage capacities, computational power, and the available number of publicly available data corpora. Nowadays, this is no longer an issue, as computers have become faster, storage has become cheaper, and there has been a massive increase of the data available for many different application scenarios. The importance of training data was underpinned by an experiment, where Banko and Brill (2001) showed that the amount of training data has a higher influence on the performance of a model than the choice of machine learning approach.

The Risk of Biased Data

When a model is trained, it is important that the data fit the requirements of the desired application domain. Especially with the huge amount of data that is available nowadays, it is important to keep an eye on the quality of the data, as the quality of the data has a significant impact on the quality of the model. If a model is trained with data that are not appropriate for a certain purpose, the resulting model will most likely not perform well in this domain. This fact is often summarized by the term “garbage in, garbage out.” For instance, Buolamwini and Gebru (2018) found that many datasets in machine learning are systematically biased on axes such as race or gender. Likewise in NLP, datasets have been found to contain biases, for instance, toward the race of a speaker (Sap et al., 2019).

Most NLP systems are limited to a single language. So far, most research is still done on the English language, and data from other languages are often ignored (Bender, 2009). This creates some major problems: focusing on English will induce a bias to the results as there are many sources in other languages that are underrepresented in the analysis, especially when analyzing information from news, social media, or blogs (Loginova et al., 2021). Looking at the number of native speakers, languages such as Spanish or Mandarin have more native speakers than English.

Another challenge that comes with the over-representation of the English language compared to other languages is the curse of dimensionality. If a dataset is too wide, i.e., includes a large number of features compared to the sample size, this increases the chances ofoverfitting. This means it is quite likely for a machine learning model to find “fake” correlations that do not exist. Therefore, transferring high-dimensional NLP techniques based on English to languages with smaller user bases can yield bad results (Johns, 2019).

Data Challenges for Under-Resourced Languages

While data collection in general can already be a challenging task, under-resourced languages introduce further challenges for the generation of data corpora. Under-resourced languages are characterized by the following aspects (Besacier et al., 2014):

- no stable orthography or unique writing system
- limited or no presence on the internet
- no or only little linguistic expertise
- limited electronic resources for NLP tasks like annotated (monolingual) data corpora, transcribed speech data, bilingual dictionaries, vocabulary lists, etc.

Under-resourced languages are often also referred to as less-resourced languages, low-density languages, low-data languages, or resource-poor languages. It is important to differentiate between under-resourced languages and minority languages. The latter refers to languages that are only spoken by a minority of people but might still be well-resourced. One example of a minority language would be the Catalan language.

When collecting data, one major challenge for under-resourced languages is that there is usually a gap between language experts (i.e., the people who speak that language) and technology experts (i.e., the people developing a system). In many cases, it can be very difficult to find native speakers who also have the technical skills to build an NLP system in their language. Another problem is that most under-resourced languages are not described well in linguistic literature.

Crowdsourcing
In crowdsourcing, difficult tasks for computers are outsourced to a human "crowd" in the web.

Due to the lack of resources, innovative methods for data collection are necessary, such as **crowdsourcing** (Gelas et al., 2011) or multilingual acoustic models (Le & Besacier, 2009; Schultz & Waibel, 2001), which share information between languages. To address the poor documentation of under-resourced languages in linguistic literature, one possible solution is to use knowledge and resources from similar languages and try to map features, for instance, the phonetics, from more resourced languages to the under-resourced language (Besacier et al., 2014).

Data Transcription and Annotation

When NLP models are trained, it is important to differentiate between supervised and unsupervised learning. In supervised learning, labeled data are used to train a model, while in unsupervised learning, there are no labels.

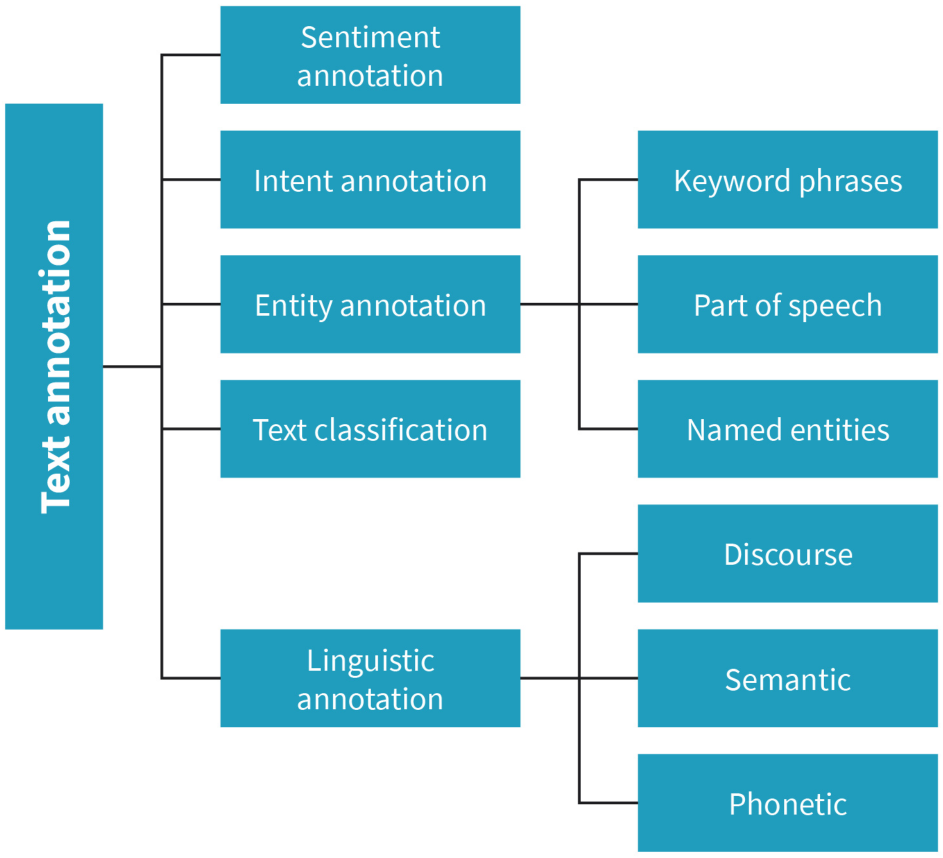
For supervised learning algorithms in NLP, we need labeled data. This means that, depending on the NLP task, we need to transcribe and/or annotate the dataset used to train a model with additional metadata. This process is necessary to make the data understandable to the computer.

To build a model for speech recognition, it is necessary to have a vast amount of transcribed data available. Speech transcription is about the conversion of spoken words into text – be it orthographic or phonetic transcripts.

In orthographic transcription, the standard spelling system of the target language is used to convert speech into text. In phonetic transcription, a text is transcribed into phones (i.e., speech sounds) using symbols like the International Phonetic Alphabet (IPA). Nowadays, especially for languages like English, there are powerful speech-to-text models that can be used to transform spoken language into text. However, for other languages, there is still a lack of fully transcribed data corpora.

Text annotation refers to the process of adding labels to text files about their content. There are several types of text annotation. The most important types are illustrated in the figure below.

Figure 18: Different Types of Text Annotations



Source: Kristina Schaaff (2023).

Sentiment analysis deals with the identification of the sentiments or emotions of a text. Therefore, in sentiment annotation, a text is annotated according to the emotion(s) it reflects. This information can, for instance, be used to analyze data from social media or from customer reviews.

Intent annotation is about finding out the intention behind a text. This could, for instance, be the classification of a customer request into categories such as request, confirmation, or command. According to the classification result, the request could automatically be routed to the responsible department.

In entity annotation, different key phrases, parts of speech, or named entities are identified. Key phrase extraction can quickly identify the content of a text or a document (Gollapalli et al., 2017). Part-of-speech (POS) tagging deals with the identification of words according to their grammatical categories, such as nouns, adjectives, verbs, or adverbs. Named entity recognition (NER) is about the identification of named entities, such as locations, persons, dates, or organization names (Li et al., 2022).

In text classification, a text is labeled according to its category. This can be done on a sentence level or for whole paragraphs and documents.

Linguistic annotations can be divided into three subcategories: discourse annotation, semantic annotation, and phonetic annotation. While discourse annotation deals with the identification of contextual knowledge, semantic annotation concerns the annotation of word definitions. Phonetic annotation labels parameters such as stress, intonation, and pauses in speech.

Data Corpora and Toolkits

As NLP is an important field of research in the area of AI, there are a large number of data corpora available that can be used to develop and train models and algorithms.

Platforms like Kaggle provide datasets from different areas. NLP frameworks such as spaCy provide fast statistical NER, including a named entity visualizer. Using spaCy, it is either possible to train your own model or to use a pre-trained model, which is included in the framework. The Natural Language Toolkit (NLTK) is another framework that provides, for instance, a pre-trained model for NER (Bird et al., 2009). Gensim is another NLP framework, which focuses on topic modeling.

3.2 Evaluation of NLP Systems

The importance of NLP systems in our everyday life is constantly increasing. Therefore, it is important to have systems that are as reliable as possible. To make NLP systems more robust, it is important to continually improve the underlying models. The goal is to find a model that is optimally able to fit future data. This means that the error rate of the predictions should be minimized.

Training, Validation, and Test Sets

For proper model evaluation, the data that are used to develop and optimize a model is randomly split into three distinct datasets: training, validation or development, and test sets. In supervised learning, the sample data consist of a pair of an input and an output vector, where the output vector contains the labels of the dataset.

The training dataset is used while a model is trained. It contains sample data that are used during the training process to fit the model parameters. During the training, the model will be run with the training data (James, 2013, p. 176). The results of the model are compared to the labels of the output vector. Depending on the results, the model will be adjusted accordingly. Besides the fitting of the model parameters, the training of a model can also include the selection of the variables that are best suited for the estimation of the target variable.

Validation set
The validation set is often also referred to as the development set.

In the next step, the **validation set** is used to further optimize the performance of the model developed based on the training data. It is important that the validation data have not been used for the training, so as to obtain an unbiased evaluation of the model developed during the training.

Finally, the test set is the dataset that is used to evaluate the performance of the final model. It is important that the samples in the test set are not used during training or optimization of the model, to obtain an unbiased final evaluation. The test set is used only once. In the event that the results produced by the test set do not match expectations, a completely new test set has to be used if the process of model training is repeated.

To obtain a robust model, it is crucial to have datasets that follow similar probability distributions and are independent of each other.

Evaluation Metrics

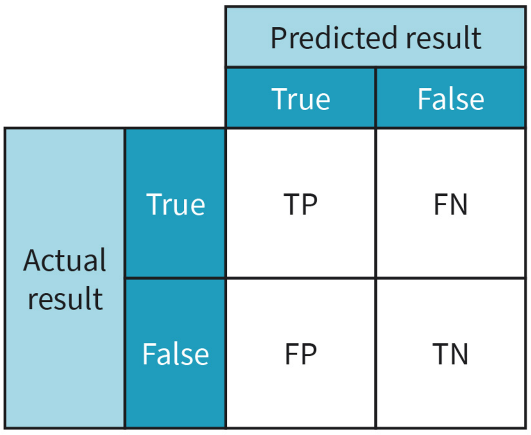
When algorithms are developed and tuned, we will need some metrics to evaluate the developed models and compare them to other systems. For binary classification tasks, commonly used metrics for model evaluation are precision and recall, the F-score, and accuracy.

To understand these metrics, we will consider credit scoring. Credit risk scoring is a typical binary classification problem, where AI is commonly used to decide whether a customer is creditworthy or not. Depending on how the decision for the creditworthiness of a customer is made, the results can be categorized as follows:

- If a customer is classified as creditworthy and turns out to be creditworthy, this is called a true positive (TP), i.e., the algorithm's prediction of a positive outcome was correct.
- If a customer is classified as creditworthy but turns out not to be (e.g., because of failure to pay), this is called a false positive (FP). In this case, the algorithm had predicted a positive outcome, but the outcome turned out to be negative.
- If an algorithm predicts a customer to not be creditworthy and it turns out that the customer actually goes bankrupt, this is called a true negative (TN), meaning that the algorithm's prediction of a negative outcome was correct.
- If a customer is classified as not being creditworthy but would have been in fact, we call this a false negative (FN), i.e., the algorithm has predicted a negative outcome even though it would have been positive.

The results of the classification can be represented in a contingency table, which is called a confusion matrix or error matrix. The confusion matrix has two dimensions: one for the predicted and one for the actual outcome. The table below shows what the confusion matrix looks like and how the aforementioned classification results can be displayed in the matrix.

Figure 19: The Confusion Matrix



Source: Kristina Schaaff (2023).

Once we have the classification results, the aforementioned evaluation metrics can be computed.

Accuracy

Accuracy is probably the performance measure that is the easiest to understand and interpret. It is defined as the ratio of those samples that have been correctly classified in relation to the total number of samples:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

For the credit risk assessment, it would be computed by taking the number of decisions that have been correctly made by dividing them by the total number of decisions about the credit risk.

While the accuracy is a very straightforward performance measure, it has the disadvantage that it is not very robust toward unbalanced data, i.e., if there are large differences in the number of samples for each class. For credit risk assessment, for example, it is quite likely that the number of positive decisions exceeds the number of negative decisions.

Precision

Precision reflects how many positive samples have been classified correctly with regard to the total number of samples in this class:

$$Precision = \frac{TP}{TP + FP}$$

In our example with the credit risk assessment, this would be the number of customers who have been identified as creditworthy and paid back their credit in relation to the total number of customers who have been identified as creditworthy, no matter whether they were able to pay back their credit or not.

Recall

Recall denotes the number of positive samples that have been identified correctly in relation to the total number of samples that should have been predicted to be positive:

$$Recall = \frac{TP}{TP + FN}$$

If we look at the creditworthiness prediction, this would be the number of customers who have correctly been identified to be creditworthy in relation to the total number of customers who would have been creditworthy.

F-Score

The F-score is a score that combines precision and recall in one single number using the harmonic mean:

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

The F-score can range from 0 to 1. A value of 1 or close to 1 is correlated with high values of precision and recall, which means very accurate classification results. A value of 0 means that either the value of precision or recall is zero.

The ROC Curve

If we look at classification tasks, most algorithms will return a percentage of how likely it is that a sample belongs to a particular class. For instance, in a sentiment analysis task, this could be the decision of whether a customer review is classified as positive or negative. Therefore, we need to set a threshold or cutoff value, which indicates when a sample is classified into a certain category. Setting this threshold to 75 percent could, for instance, mean that all model outputs from our sentiment analysis task with an output value higher than 75 percent would be classified as positive.

To find the optimal threshold, the receiver operator characteristic (ROC) curve can be used. For every possible cutoff value, the ROC curve shows the trade-off between the true positive (TP) and the false positive (FP) rates. Therefore, the shape of the ROC curve indicates how strongly the classes are distinguished when the cutoff value is varied (Kulkarni et al., 2021, p. 9). Ideally, the TP rate should be 100 percent and the FP rate 0 percent. However, as this is normally not the case, the ROC curve can help find the cutoff value that maximizes the TP rate while minimizing the FP rate. TP and FP rates can be computed as follows:

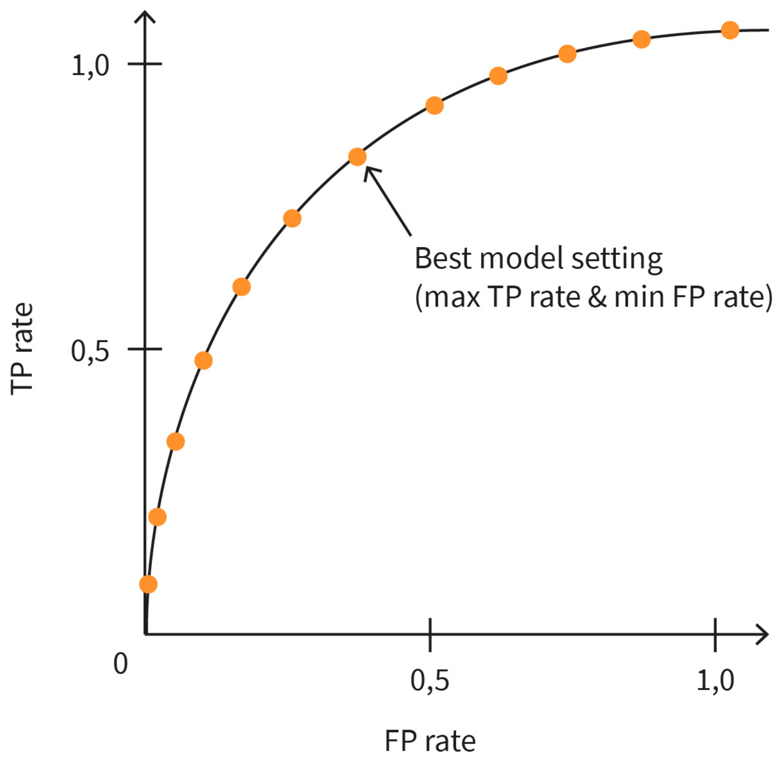
$$\text{TP rate} = \frac{TP}{TP + FN}$$
$$\text{FP rate} = \frac{FP}{FP + TN}$$

The TP rate and the FP rate form the axes of the ROC curve. The computation of the ROC curve can be done with the following steps:

1. Set the cutoff value to a value between 0 and 100 percent.
2. Compute the TP, TN, FP, and FN values for the testing set according to the classes defined by the cutoff value.
3. Calculate the TP rate and FP rate.
4. Enter the resulting point on the ROC curve.
5. Set a new cutoff value and continue with step 2.

The following figure shows an example of a ROC curve.

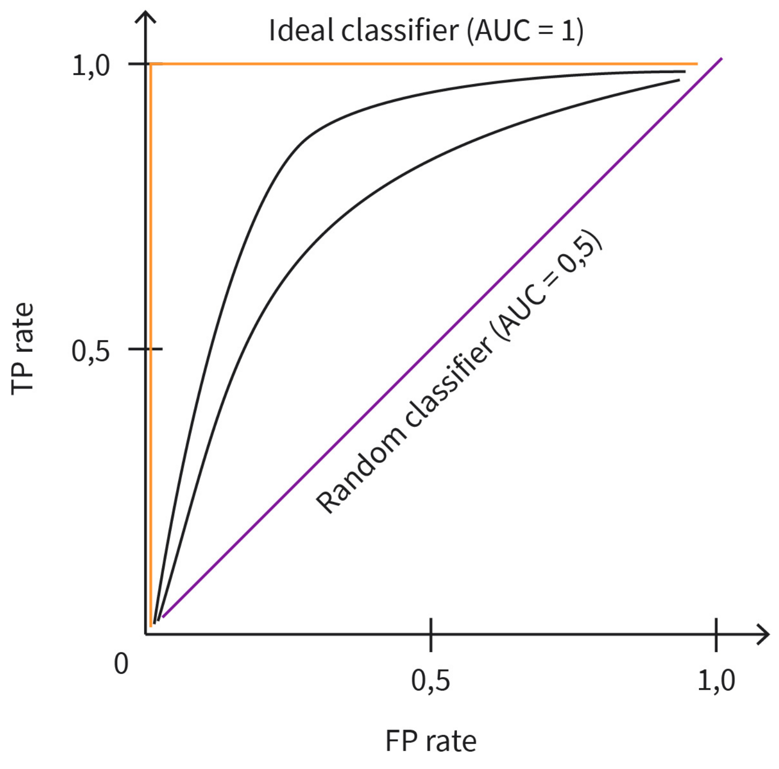
Figure 20: The ROC Curve



Source: Kristina Schaaff (2023).

The closer the curve is to the left upper corner, the better is the predictive power of the model. To measure this, the area under the curve (AUC) can be calculated. In the figure below, different shapes of ROC curve are illustrated.

Figure 21: Different Shapes of the ROC Curve



Source: Kristina Schaaff (2023).

For an ideal model, the AUC would be 1, while for a random model, the AUC would be 0.5. In reality, the value is somewhere in between.

Evaluation of Machine Translation

Early evaluations of MT were normally done manually by evaluating metrics such as fluency and adequacy. When fluency is evaluated, the person evaluating the system has to be fluent in the target language to be able to judge if the output is fluent or not. The accuracy of the translation of the source words is not analyzed.

Adequacy, does not analyze how fluently a text is written but evaluates how well the information from the source is contained in the system output. For this purpose, the annotator must know both languages, but it is not necessary that they are fluent in both languages.

Both metrics are normally measured separately for each sentence and evaluated on a five or seven point scale (Przybocki et al., 2009). These scores can also be averaged to give a single score for the system evaluation (Snover et al., 2006).

Evaluating the results of MT is a time-consuming – and therefore also expensive – task for humans. Moreover, manual metrics often lack repeatability and objectivity, as the output strongly depends on human judgments. For this reason, automated metrics can help evaluate the results of the translation. The most popular metric for MT is the bilingual evaluation understudy (BLEU), which rates the quality of a translation based on how well a machine translation corresponds to a human translation. The basic idea is that a machine translation is better the closer it is to a professional human translation (Papineni et al., 2001, p. 311).

The BLEU score can be computed by comparing “n-grams of the candidate with the n-grams of the reference translation and count[ing] the number of matches. These matches are position independent. The more matches, the better the candidate translation” (Papineni et al., 2001, p. 312). The calculations are based on n-grams at a word level, meaning a sequence of *n* words. The resulting score will be a number between 0 and 1. The score can be seen as a similarity measure between the hypothesis and the reference text. The closer a value is to 1, the higher the similarity between both texts. As “systems have been known to generate more words than are in a reference text” (Celikyilmaz et al., 2018, p. 71), a modified form of precision is used for BLEU.

BLEU can also be used to evaluate other NLP tasks, such as text summarization or language generation.

3.3 Domain Challenges

When NLP is used to solve a task, it is important to take the specific domain and/or setting of the input data – be it text or speech – into account.

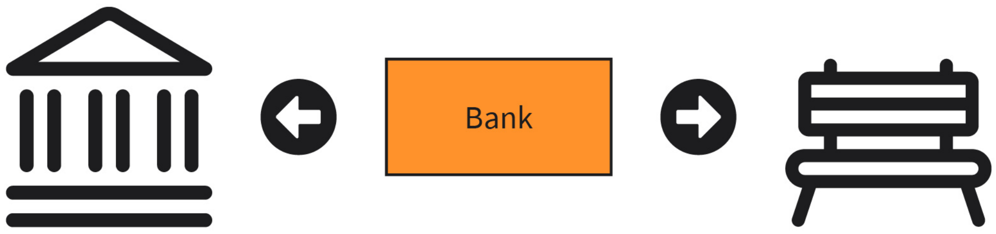
Variations of Application Domain

Different words and phrases can have various meanings depending on the application domain. This phenomenon is also referred to as domain mismatch. The best performance can be achieved if a system is adapted to the domain that best matches the respective use case. If, for instance, a dictation software tool is trained for lawyers, it might not perform well when being used by a reporter trying to dictate an article about sports. In the same way, a sentiment analysis program that has been trained on tweets might not necessarily have the same performance on customer reviews. A system for automatic short answer grading that has been designed for questions from business administration might have bad performance on math exams.

The quality of the outcome strongly depends on the quality and relevance of the text corpora that are used for training. If a text corpus from the wrong domain is used for training, the model will most likely have a poor performance in the domain for which it has been designed. In general, the level of diversity of the data used as an input for a system has a great influence on the generalization and abstraction abilities of a model. A model will perform better in a specific domain if it is trained with a more domain-specific input. However, this will reduce its ability to generalize and to perform on out-of-domain topics.

The figure below illustrates an example of how a word can have different meanings depending on the domain in which it is used.

Figure 22: Different Meanings of One Word



Source: Kristina Schaaff (2023).

To illustrate this, Chu et al. (2017) used a Chinese–English machine translation system that was originally developed and trained for texts in the patent domain. Without any adaption, this system performed poorly when trying to translate TED talks. After a domain-specific adaption with texts from other TED talks, it was possible to improve the performance of the system up to seven times.

In statistical machine translation (SMT), the domain adaption is usually done by adapting the entries in the language model and the phrase table (Su et al., 2012).

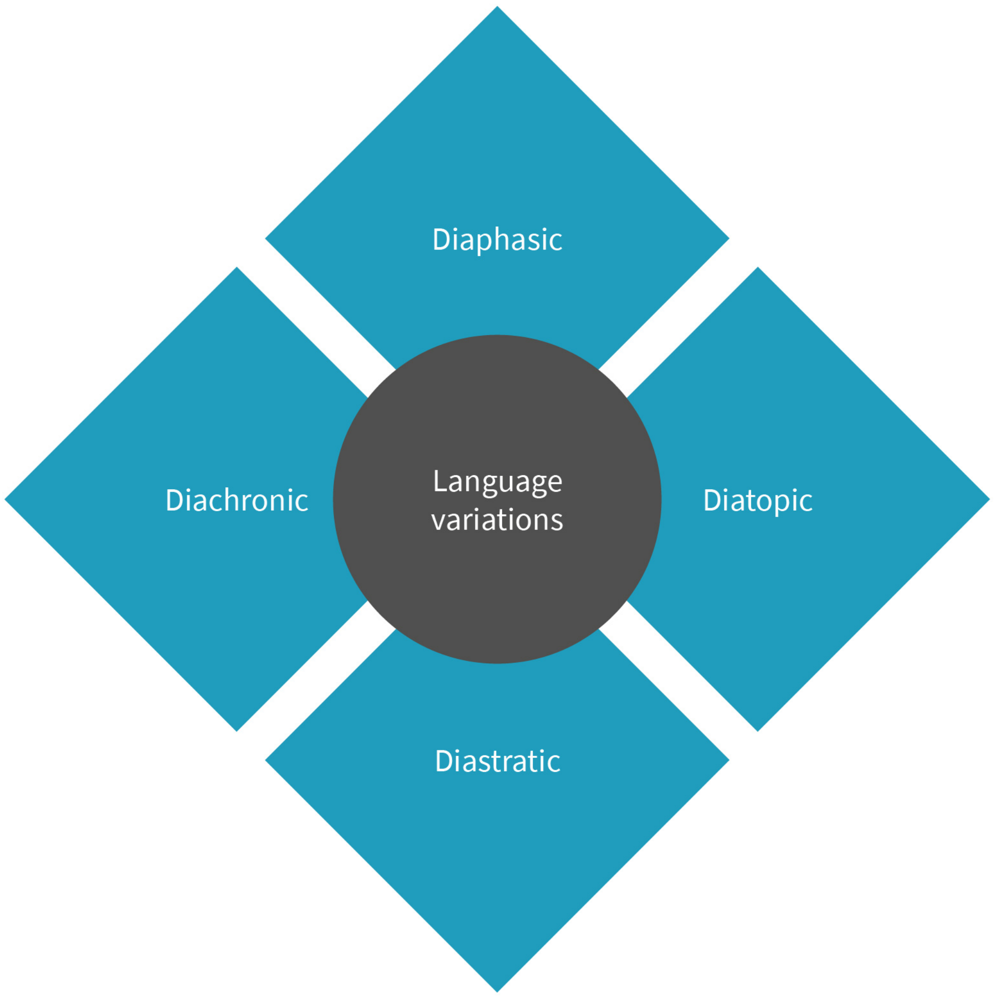
Looking at neural machine translation (NMT), a popular method for domain adaption is to first train a system for a general domain and then perform a training on domain-specific data for some of the epochs (Freitag & Al-Onaizan, 2016; Koehn & Knowles, 2017a).

Language Variations

Language can vary depending on the environment and the circumstances. On the phone, we will communicate differently than when writing an email; when talking to a customer, we will talk differently than when we talk to friends.

There are four dimensions of language variation: diaphasic, diatopic, diachronic, and diastratic variation (Zampieri et al., 2020).

Figure 23: Dimensions of Language Variations



Source: Kristina Schaaff (2023).

Diaphasic variations are related to the situation where communication happens. This means the variation is related to the communication medium or setting. This can, for instance, be the difference between oral and written communication or different degrees of formality of a language. In Japanese, for instance, there are four different levels of formality. While the vocabulary of the lower levels is quite similar, the politest level uses different vocabulary for some of the words.

Diatopic variations refer to the linguistic area, which can, for instance, be variations in race, different dialects, or also national varieties of the same languages – such as American versus British English.

One challenge about data dealing with dialects is that they are typically underrepresented in written resources. Therefore, when analyzing dialects, new text corpora have to be produced by transcribing speech. Depending on the dialect, this can either be done automatically (Ali et al., 2015) or manually (Scherrer et al., 2019).

For variations of the same languages, like British and American English, diatopic variations are less challenging, as both have their own written standards and sometimes even use different words, such as “rubbish” in British English and “garbage” in American English (Zampieri et al., 2020).

Diachronic variations are about the variations of language over time. This includes, for instance, old-fashioned and obsolete words but also changes in terminology in recent vocabulary. One example of how the meaning of a word can change over time is the German word *Querdenker*, which, before 2020, was used as a term for someone who thinks in an unconventional way and was mostly perceived in a positive way. After the start of COVID-19, it started to become a synonym for someone who denies COVID and the associated measures; nowadays, it has a mostly negative connotation.

Diastratic variations of language relate to the variations in languages that can be traced back to social groups, such as age or gender (Zampieri et al., 2020). For example, the sentence “This is no good” could also be expressed as “This ain’t no good.”

3.4 Multilingual Application

Multilingual applications in NLP are a big challenge. Besides the bias introduced by most research being done in English, under-resourced languages are another big challenge in NLP, especially when it comes to multilingual applications. In the following, we will have a closer look at the challenges of multilingual applications.

Code Switching

Code switching refers to the process when a speaker or writer changes the language (i.e., the code) while speaking/writing. This can happen within a dialogue, between sentences, or even within a sentence. Code switching commonly happens in multilingual communities when people with different languages and/or cultural backgrounds communicate (Auer, 2001). As most speech recognition systems are limited to one single language, code switching is an important topic to be addressed in multicultural settings.

To handle code switches in spoken language, the position in an utterance where the switch happens has to be detected. Most code switches occur at positions where the syntactical rules of the languages involved are not violated (Bokamba, 1989). The most frequent points for switches are between verb and object noun phrases and between determiners and nouns (Adel et al., 2013).

Features that can be used to predict code-switching points include, for instance, part-of-speech tags, language identity, and word form (Solorio & Liu, 2008).

Multilingual Sentiment Analysis

In the past decades, sentiment analysis, i.e., the detection of emotions based on language, has become an important research area in NLP (Wankhade et al., 2022). In multilingual sentiment analysis, sentiments are detected and classified based on information from texts that are written in multiple languages. While there has been a great deal of research on well-resourced languages like English or Chinese, low-resourced languages are still underrepresented. Further developments of NLP technologies for these languages are, therefore, limited, especially when it comes to research about NLP and socio-cultural and multicultural factors. There have been limited insights from past research (Lo et al., 2017).

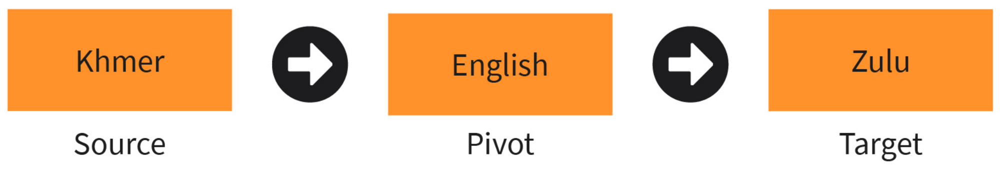
Machine Translation

Machine translation (MT) is another example of multilingual applications where under-resourced languages can be a challenge. In MT, speech or text is translated from one language to another in an automated way. The research field dealing with the automatic translation of under-resourced languages is called low-resource MT. In low-resource MT, there are no large bilingual text corpora of source and target language available.

To avoid the problem of data scarcity when training the MT system, a commonly used approach is pivot MT (Kim et al., 2019; Wu & Wang, 2009). Pivot MT tackles the problem by using a pivot language to close the gap between source and target language (Deng & Liu, 2018, pp. 147–183).

The figure below illustrates the translation process for the example of translating from Khmer to Zulu.

Figure 24: Example of Pivot Machine Translation



Source: Kristina Schaaff (2023).

In the example, two cascading systems are used instead of only one single direct system. A text is first translated from the source language (Khmer) into the pivot language (English). After that, the second system continues the translation from the pivot language into the target language: Zulu.

This approach has the advantage that there are significantly more data in the language pairs Khmer – English and Zulu – English than we would have using a direct language model for Khmer – Zulu.

SUMMARY

One of the key elements ofdeveloping a good NLP model is the availability of reliable data to train the system. One of the problems that must be faced when collecting data is that biased data and factors like under-resourced languages can make it difficult to find a data corpus that is suitable for a certain task.

To develop an NLP system, data are split into training, validation, and test data. Moreover, it is important to be able to compare the system to other systems and have metrics for good parameter optimization. For this purpose, metrics like accuracy, precision, recall, and the F-score can be used.

Variations in the application domain and in language pose further challenges for NLP systems. When developing multilingual applications, phenomena like code switching or under-resourced languages are problems that have to be dealt with.

UNIT 4

TECHNIQUES

STUDY GOALS

On completion of this unit, you will be able to ...

- differentiate between rule-based and statistics-based systems.
- work with regular expressions.
- know the basics of n-grams.
- understand the vectorization of data.
- explain the concept of NLP models.

4. TECHNIQUES

Introduction

Early natural language processing (NLP) systems were mostly based on rule-based techniques. Since then, systems have shifted toward statistical models. Therefore, this unit starts with an introduction to the difference between the two systems. Afterwards, the concept of regular expressions is introduced; they are a powerful tool to search, for instance, for a specific term in a given text.

To determine the meaning of a whole text, it is important to not only look at single words but also at the combination of words. Therefore, the concept of n-grams will be introduced. After that, we will look at different techniques for vectorizing data. We will start with the simple bag-of-words approach and, afterwards, consider neural models to build word and sentence vectors.

The unit closes with an overview of how NLP models can be used for text processing. We will start with the most important steps for text preprocessing, as this is an important part of the pipeline when using pre-trained models. Afterwards, you will get an overview about the underlying concepts of statistical models and neural models.

4.1 Rules Versus Statistics

In early NLP systems, rule-based systems were applied to linguistic structures. In most cases, these rules were hand-written for a specific domain, which made it hard to transfer a system from one domain to another. Recently, there has been a shift toward systems that are based on statistical methods from machine learning, which has helped to make these systems more powerful. In the following, we will look more closely at the details of rule-based and statistical systems for NLP.

Rule-Based Systems

In rule-based systems for NLP, a given problem is addressed using a set of predefined rules. The rules used in those kinds of systems are built in a way that tries to reproduce the way humans construct sentences.

To illustrate how rule-based systems work, we will consider a very simple example of a system that extracts single words from a text. This can be done by using only one single rule, which divides a given text at every blank space. At first sight, this might look like a very simple and good solution to the problem. However, if we look at terms like “Los Angeles,” this already illustrates that the problem is more complicated than one might have originally thought. Therefore, more complex systems are required, which use formal grammars and are based on linguistic structures.

Development of rule-based systems usually involves human knowledge to build the system. This brings us to one of the major advantages of rule-based systems: explainability. Explainability is about the ability to make it unambiguously comprehensible how a system came to a certain solution. This makes it easier for humans to locate errors and to understand how a specific task has been processed.

Another advantage of rule-based systems is that they can be developed and improved in a very flexible way. When rules are changed or added, this does not necessarily mean changes to the core of the application. Moreover, the development of rule-based systems requires a comparatively small amount of training data.

However, one of the major drawbacks of rule-based systems is their lack of flexibility when it comes to the application domain. Being built for a specific domain makes it difficult to use these systems in a domain that differs from the domain it has been designed for. Moreover, setting up the rules for a rule-based system requires human experts to build the rules.

Statistics-Based Systems

In the past decades, computational power has significantly increased. This paved the way for the application of statistical methods. Statistical methods are often summarized under the term “machine learning.” Nowadays, those systems have replaced most of the rule-based systems.

Statistical methods follow a data-driven approach. To generate a model in statistics-based methods, a huge amount of training data is used for a given task. Once a model is trained, it can be used to make predictions for an unknown set of data.

One of the major differences to rule-based systems is that statistics-based systems do not require a human expert with domain knowledge to set up the rules. This comes, of course, at the expense of the explainability of the systems.

Statistics-based systems can be set up quite easily based on existing systems by adapting them with appropriate data. Additionally, it is much easier to transfer a model from one domain to another than for a rule-based system.

4.2 Regular Expressions

One common task in NLP is to search for a specific pattern in a given text or string. A powerful tool to tackle this problem is to use regular expressions (also referred to as “RegEx”). Most programming languages, like Python, JavaScript, or Perl, and even shell scripts or the UNIX command line, are able to handle regular expressions. Also, some editors, like Vim or Emacs, are able to work with regular expressions for operations like search and replace.

In rule-based NLP techniques, regular expressions are commonly used to perform tasks like extracting data from text. Above all, this can help if well-defined patterns like times, prices, or dates have to be identified and extracted.

Basic Concepts of Regular Expressions

In the following, we want to introduce the basic concepts of regular expressions.

There are some "metacharacters" that often occur in regular expressions: `\, ^, $, .,` `|, ?, *, +, (, ), [, ], {,` and `}`. If some of these characters are used as a literal, they have to be escaped with a backslash. Otherwise, they will be executed as a regular expression.

Anchors

Anchors are used to mark a position in a string:

- `^` marks the start of a string.
- `$` indicates the end of a string.

Some examples are as follows:

- `^Apple` will match the string "Apples are not pears."
- `apple$` will match the string "I want to eat an apple" but not "Apples are not pears."
- `^Apple$` will only match the string "Apple" but not the other two sample sentences.

Character classes and disjunctions

Disjunctions represent a logical OR. Disjunctions are either separated using the pipe sign `|` or written in square brackets `[]`.

Some examples are as follows:

- The expressions `pe[aeu]r`, `pe(a|e|u)r`, or `pear|peer|peur` would all match the words "pear," "peer," and "peur."

Character classes are used to represent a certain group of characters in a regular expression:

- `\d` or `[0-9]` will match any digit.
- `\w` or `[0-9_A-Za-z]` will match alphanumeric characters and the underscore.
- `\s` will match all white spaces.
- `.` will match any arbitrary character.

Character classes and disjunctions can be negated using `^.` For example, `\^d` would mean that a character is not a digit.

Quantification

Quantifiers make it possible to define repetitions of preceding elements. In regular expressions, the following quantifiers are used:

- `?` indicates that the preceding element occurs zero or one time.
- `*` marks zero or more occurrences of a character.
- `+` marks one or more occurrences.
- `{n}` indicates that the preceding element occurs exactly *n* times.
- `{n,}` indicates that the preceding element occurs at least *n* times.
- `{n,m}` indicates that the preceding element occurs at least *n* and at maximum *m* times.
- `{,m}` indicates that the preceding element occurs at maximum *m* times.

Some examples are as follows:

- `ap{1,3}le` will match aple, apple, and appple.
- `app?le` will match aple, or apple.
- `a[p]+le` will match aple, apple, appple, and so forth.

Combinations of Regular Expressions

The regular expressions introduced above can be combined in various ways to search for specific terms in a text.

If we want to find, for instance, all Euro prices in a given document, this could be done using the following regular expression:

```
€[0-9]+
```

This expression would return all whole number prices. However, if a price has decimals, only the whole number would be returned.

Therefore, we need to extend the expression in the following way:

```
€[0-9]+.[0-9][0-9]
```

This pattern can also detect prices with decimal digits. However, this expression will not be able to detect whole number prices anymore. Therefore, we need to make the cent digits optional:

```
€[0-9]+(.[0-9][0-9])?
```

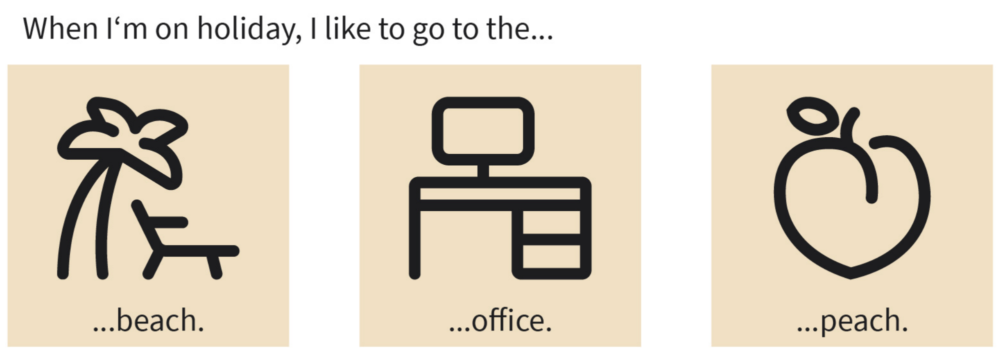
Now all prices in the document can be detected correctly.

4.3 N-Grams

So far, we have learned a lot about the identification of single words. However, when we only look at single words, we will not be able to determine the meaning of a whole text. Therefore, it is important to not only look at the occurrence of single words but also at the combination of different words. This can be done using **language models**. To understand why we need language models, let us start with a simple example illustrated in the figure below.

Language model
A language model is used to model a probability distribution for a sequence of words.

Figure 25: Example of the Use of Language Models



Source: Kristina Schaaff (2023).

A good language model will most likely be able to identify that it is more likely that a person wants to go to the beach while being on holiday than wanting to go to the office and therefore complete the sequence of words accordingly. Completing the sequence from the example with the word peach will have the lowest probability, as the sentence does not make sense.

N-grams are the simplest language model and can be used to assign probabilities to a sequence of words, or sentences. In general, a sequence of *n* words is called an n-gram. For example, a sequence of two words will be called a 2-gram (also known as bigram). This could, for instance, be "hello world." The sentence "how are you" would be called a 3-gram (or trigram). N-grams can be used to estimate the probability of the last word of a sequence of n words based on the previous words. The task is, therefore, to predict the probability P of a word w based on the history h:

$$P(w \ \ occuring \ \ after \ \ h) = P(w|h)$$

For the example above, it could look like this:

$$P(w|h) = P(beach|I \ \ like \ \ to \ \ go \ \ to \ \ the)$$
$$P(w|h) = P(peach|I \ \ like \ \ to \ \ go \ \ to \ \ the)$$

The easiest way to estimate this probability is to use the frequency counts based on a large data corpus, such as the internet:

$$P(w|h) = \frac{C(hw)}{C(h)}$$

Looking at our example, we will receive the following equations:

$$P(\,beach|I\ \ like\ \ to\ \ go\ \ to\ \ the) = \frac{C(I\ \ like\ \ to\ \ go\ \ to\ \ the\ \ beach)}{C(I\ \ like\ \ to\ \ go\ \ to\ \ the)}$$
$$P(\,peach|I\ \ like\ \ to\ \ go\ \ to\ \ the) = \frac{C(I\ \ like\ \ to\ \ go\ \ to\ \ the\ \ peach)}{C(I\ \ like\ \ to\ \ go\ \ to\ \ the)}$$

In summer 2022, the count for the sequence *"I like to go to the beach"* on Google was more than 8.5 million, while the count for the sequence *"I like to go to the peach"* was zero. The word sequence *"I like to go to the"* occurred more than 63 million times (Google, n.d.). In probabilities, this means

$$P(\,beach|I\ \ like\ \ to\ \ go\ \ to\ \ the) > P(\,peach|I\ \ like\ \ to\ \ go\ \ to\ \ the)$$

The example illustrates how n-grams can help in speech recognition, as both sentences might sound quite similar in spoken language. However, it is more likely that the last word of the sentence is beach, not peach.

In our example, we looked at the whole history of a word. In a bigram language model, we would only analyze sequences of two words (i.e., "the beach" versus "the peach"), in a trigram language model a sequence of three words, and so forth. The higher n, the more accurate the predictions can be. However, it also increases the computational power and the risk of sparse data.

Working with the counts of a word sequence is a very straightforward approach. However, it has some disadvantages. As the example has already illustrated, even for large data corpora, it can happen that the count of a sentence is zero. This will lead to a probability of zero even though the probability might be larger than zero. Moreover, when wanting to predict the probability of a sequence of words, the number of counts can get very large if we want to compare it to all possible sequences of that length, which will consume a lot of resources.

4.4 Vectorizing Data

The input data for machine learning algorithms have to be in a numerical format. Therefore, information from unstructured text has to be represented in a way that enables the computer to process that text. To transfer a text in a numerical format, we need to find a way to embed words in a semantic vector space.

In the following, you will learn more about how a text can be vectorized using simple approaches like bag-of-words but also other concepts to vectorize words and sentences.

Bag-of-Words

The bag-of-words (BoW) model is one of the easiest approaches when converting information from text into numbers. BoW represents a text as a vector that contains information about how often a word occurs in a given text. All words from a text are put into one unique set of words – referred to as “bag.” During this process, information about the word order or the structure of this text gets lost. Let us look at an example to illustrate the BoW approach. For the example, we will use the following sentences:

1. I like to drink coffee.
2. I do not like tea.
3. Tea is not like coffee.

First, we have to extract all unique words from the sentences. This can be done using tokenization. We will receive the following words for the above sentences:

I, like, to, drink, coffee, do, not, tea, is

Using the words, we can build a word vector. For our example, we will receive a vector with a length of nine. Using this vector, we can perform a scoring for the words in the respective sentences. For the sentences from our example, the word vectors will look like this:

1. [1, 1, 1, 1, 1, 0, 0, 0, 0]
2. [1, 1, 0, 0, 0, 1, 1, 1, 0]
3. [0, 1, 0, 0, 1, 0, 1, 1, 1]

To summarize the scores of the BoW model, different approaches can be used. In a Boolean representation, the resulting vector is a simple indicator of whether a word occurs in a sentence or not. In our example, the vector summarizing all three sentences would look like this:

[1, 1, 1, 1, 1, 1, 1, 1, 1]

Using the count of words approach, the resulting vector will reflect the number of occurrences of a word in a given text, which would look as follows in our example:

[2, 3, 1, 1, 2, 1, 2, 2, 1]

Both representations have in common that the information about the word order in the text gets completely lost. Being a very simple approach, the BoW model comes with some major disadvantages:

- It is important to select the vocabulary carefully to find the right balance between the number of words and sparsity. When the size of the model increases, this will also increase sparsity of the BoW vectors. Higher sparsity of the model can also increase computational costs.

- The meaning of the words can get lost when BoW is used, as neither context, word order, nor sense is analyzed. If we look at our example, we will notice that the way the word “like” is interpreted in the respective sentences strongly depends on the context – once it is used as a verb, the other time as a preposition. To distinguish between two meanings of a word, BoW does not perform well.

Word Vectors

To be able to use words as an input for models in machine learning, like linear classifiers or artificial neural networks, the words need to be transformed into word vectors. This will allow the words to be embedded in a semantic vector space. Once the words have been transformed, similarities and word analogies can easily be found by applying linear operations. In order to do so, methods like cosine similarity can be used. Cosine similarity measures the similarity between two vectors based on the cosine of the angle between those two vectors. The cosine similarity can be computed as the dot product of the vectors divided by the product of the vector lengths:

$$\text{cosine similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}|| \quad ||\mathbf{B}||}$$

Cosine similarity can have values between -1 and 1. A value of 0 means that the two vectors are orthogonal, i.e., independent of each other. A value of -1 means that the two vectors are opposite, while +1 means that they are pointing in the same direction. When word vectors are built based on word frequencies, the value of the cosine similarity will range from 0 to 1, as the underlying word frequencies cannot be negative.

We will now have a look at some methods for word vectorization: Word2Vec, the TD-IDF algorithm, and GloVe.

Word2Vec

The Word2Vec model is a rather simple approach that is based on a neural network. When Google Research first published Word2Vec, this denoted an important milestone in NLP research. The neural network uses one single hidden layer to generate word embeddings (Mikolov et al., 2013). The neural network in Word2Vec expects a one-hot vector as an input. This one-hot vector is built using BoW. In order to build the vector, all values of that vector are set to 0. Only the index of the word that is being analyzed is set to 1.

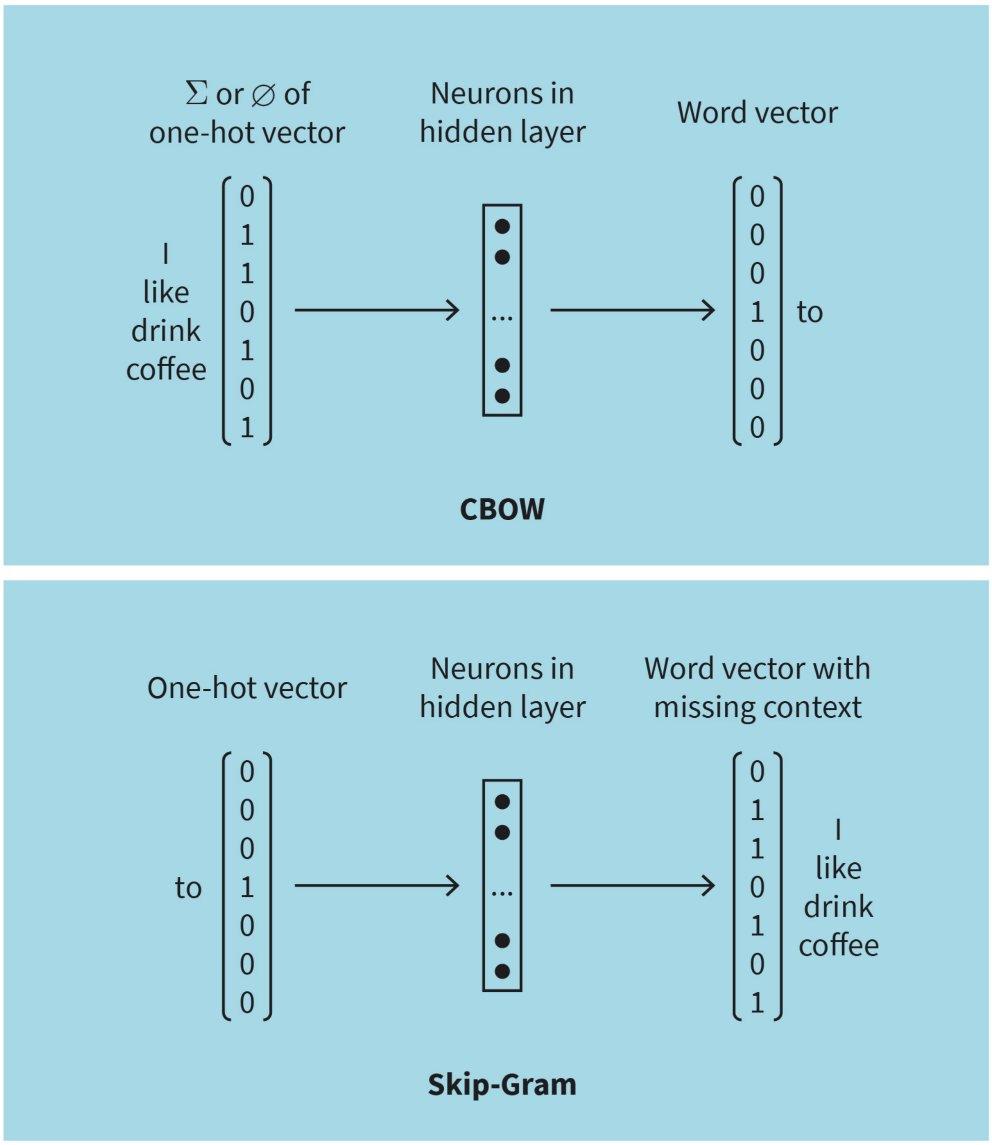
To obtain a proper model for Word2Vec, a large text corpus – for instance, a Wikipedia dump – is required. To train the model, a sliding window with a fixed length of N is moved over the text. Typical sizes of the sliding window would be $N = 5$ or $N = 10$.

Two different models are used for predictions: continuous bag-of-words (CBOW) and skip-gram.

CBOW is used if we have a sentence of length N and want to predict a missing word based on the context of the $N - 1$ other words of the sentence. The input vector can be constructed using either the sum or the average of the one-hot vector.

Skip-gram, however, uses one single word in a fixed window of length N to predict the other $N-1$ context words. The figure below illustrates the difference between the two models.

Figure 26: Comparison of CBOW and Skip-Gram



Source: Kristina Schaaff (2023).

In CBOW, the order of the context words does not influence the outcome of the prediction. In skip-gram, however, the context words which are closer to the input word get more weight than context words that are more distant. While the skip-gram architecture is better suited to infrequent words, CBOW will perform faster to predict words.

The goal when training Word2Vec for a sample from the data corpus is the maximization of the probabilities of the words that appear in the fixed window. As a result, we will receive a function that is called the objective function.

Term frequency: Inverse document frequency

If we use BoW to analyze a given document, all words have the same weight. The word vectors only reflect which words are contained in the text. To get more information about the importance of a word, approaches like term frequency – inverse document frequency (TF-IDF) can be used. TF-IDF is a statistical measure that comes from information retrieval (Beel et al., 2016). To get information about the relevance of a word, term frequency (TF) and inverse document frequency (IDF) are combined.

The computation of TF-IDF is based on the following parameters:

- The term frequency (TF) reflects the number of occurrences of a term t in a document d in relation to the total number of words in the document. It will increase the more often a term occurs in a given document:

$$TF(t,d) = \frac{number\ \ of\ \ occurences\ \ of\ \ t\ \ in\ \ d}{number\ \ of\ \ words\ \ in\ \ d}$$

- The document frequency (DF) gives information on how many documents include the term t with respect to the total number of documents D. The document frequency indicates how important a text is in relation to other documents:

$$DF(t,d,D) = \frac{number\ \ of\ \ documents\ \ d\ \ containing\ \ t}{total\ \ number\ \ of\ \ documents\ \ D}$$

- The inverse document frequency (IDF) reflects how relevant a term is. It is the logarithmically scaled inverse of the document frequency:

$$IDF(t) = \log\frac{1}{DF(t,d,D)}$$

To compute the final TF-IDF score, the term frequency is multiplied by the inverse document frequency.

$$TF - IDF(t,d) = TF(t,d) \times IDF(t)$$

If the value of TF-IDF is high, this is an indicator of a word that occurs frequently in a document while the total number of documents that include that term is relatively small in comparison to the total number of documents. Therefore, more specific requests will get a higher weight. TF-IDF can, therefore, be used to identify those terms in a document that are most important in a given text.

Global vectors for word representation

Matrix factorization
a method to simplify complex matrix operations by reducing a matrix into its components

Another vectorization method that is commonly used in NLP is global vectors for word representation (GloVe). In contrast to Word2Vec, GloVe works in an unsupervised way. It is based on the word counts in a text. GloVe was developed to obtain a model that – in contrast to skip-gram – also considers statistical information about word co-occurrences. For this reason, Pennington et al. (2014) combined skip-gram with **matrix factorization**. In the GloVe approach, a co-occurrence matrix is used, which reflects information about the context of a word. Especially for similarity tasks and named entity recognition, GloVe has been proven to outperform other related models (Pennington et al., 2014).

Sentence Vectors

Converting words into vectors that a machine is able to process is an important step in NLP applications. However, in tasks like sentiment analysis or question answering, it is not enough to analyze a single word. Instead, it is necessary to look at whole sentences or paragraphs. This requires methods to transform a sequence of words into a format that can be understood by the learning algorithm.

There are various approaches to handle text snippets of various lengths in NLP algorithms. In the following, we will present a selection of the most prominent approaches. In the descriptions, the term “sentence” will not be used in a strict grammatical way but to represent a whole text paragraph.

Skip-thought

The skip-thought vectors approach (Kiros et al., 2015) transfers the skip-gram architecture from Word2Vec form a word to a sentence level.

Similarly to Word2Vec, a large text corpus is necessary to train the model. While Word2Vec uses a sliding word window, in skip-thought, the analysis window comprises a triple of three consecutive sentences. As a result, we will get a model that follows a typical encoder-decoder architecture. The encoder uses the middle sentence from the triple as an input. Based on this input, it will produce an output and send it to the decoder. The model can be further optimized by using the decoder to selectively predict the previous or the next sentence.

For NLP tasks where no prediction model is required, once the model has been trained, the decoder part can be discarded. The resulting output vector from the encoder can be used as the vector representation of the sentence.

It is possible to use the model to predict only the previous or the following sentence. The resulting vector is then called a uni-skip vector. If two uni-skip vectors are concatenated in a way that one predicts the next and the other one predicts the previous sentence, the result is called a bi-skip vector. When n-dimensional bi-skip vectors are combined with n-dimensional uni-skip vectors, the resulting vector is referred to as a combine-skip vector. In a comparison, the combine-skip model produced slightly better results than the other skip-thought models (Kiros et al., 2015).

For English, there is a pre-trained model publicly available, which is based on the BookCorpus dataset (Zhu et al., 2015).

Universal sentence encoder

The universal sentence encoder (USE) was developed by Google Research (Cer et al., 2018). USE provides a model family for sentence embedding, which is available in two different variations: either based on a deep averaging network (DAN) or based on a transformer model. While the DAN-based variant is faster than the transformer-based model, it is less accurate.

As for skip-thought, there are pre-trained models available, one multilingual and one English model, which are both based on the DAN architecture (Chidambaram et al., 2018).

Bidirectional encoder representations from transformers

The bidirectional encoder representations from transformers (BERT) model is based on the transformer architecture (Devlin et al., 2019). This model was also developed by Google Research and made available open source. The model has been pre-trained using a big text corpus using two different unsupervised and combined methods: next sentence prediction and a masked language model.

When next sentence prediction is used for training, the model is trained with a pair of sentences. The goal of the model is to predict if the second sentence follows the first sentence. The focus of the resulting model is therefore on the relation between the sentence pairs.

Using a masked language model, about 15 percent of the words from a training set are masked. This could, for instance, look like this:

“The most [mask1] thing in the morning is to [mask2] a good cup of coffee.”

In the example, the words “important” and “drink” have been masked. When the model is trained, the goal is to predict the words that are missing in the sentence. This helps the model to understand the context of the words.

Both models were trained together using unlabeled data from the BookCorpus.

4.5 NLP Models

Many NLP applications need to analyze unstructured data. To obtain a vectorized representation of the unstructured text, pre-trained NLP models can be used. The models provide labels for a text, which are either extracted from the text data or predetermined. Using those models makes it possible to quickly build NLP applications without having to train a model yourself.

There are a large number of NLP models, which employ various methods for prediction and classification tasks. The models can be categorized into two groups: statistical and neural supervised learning models.

To be able to use pre-trained models, the input data have to be prepared in a way the model can understand. Therefore, you will first get an overview of the most important text preprocessing techniques before we dive deeper into NLP models.

Text Preprocessing

Before NLP models can be used, the data – which are usually provided as human-readable text – have to be converted into a format that can be used as an input by the models. Depending on the model that will be used, this requires different preprocessing steps. Some typical preprocessing steps will be described next.

Tokenization is used to split a text into smaller sub-units, which are referred to as tokens. The tokenization can, for instance, be done using white space and punctuation.

Stop-word removal removes words that have no impact for a specific NLP task. Typical stop-words are articles and pronouns. There are hand-curated word lists for different languages that contain words that occur frequently across different text corpora.

Using lemmatization, the words of a text are converted to their base form, which is called a lemma. For instance, the words "going" and "went" would all be converted to the word "go;" the word "stories" would be lemmatized to "story." Lemmatization often requires look-up tables and can therefore be computationally intensive.

Stemming is another method to convert words to their base form. However, in contrast to lemmatization, only the suffix (i.e., the last few characters) from a word is removed in stemming, which can sometimes lead to incorrect results. If we look, for instance, at the word "caring," lemmatization will correctly return the word "care," while stemming would return "car." However, especially for large datasets, using stemming can be beneficial due to performance. In order to perform the preprocessing, toolkits like spaCy or NLTK provide many methods that can used.

Statistical Models

Statistical language models are based on a statistical probability distribution over strings on a given alphabet. Most commonly, those models work at the word level. Knowing the statistical distribution of words can, for instance, be used for auto-completion tasks or the detection of spelling errors, but also in tasks like named entity. One of the simplest examples of statistical NLP models are n-grams, which we covered in previously in this unit.

Statistical models like n-grams are based on the Markov assumption, meaning that given a current word, the following words are independent of the past words. Therefore, the probability of a word is calculated only on the probability of a (limited) history of words.

The advantage of statistical models is that they are comparatively easy to train given a large data corpus. However, samples that have not been observed in the training data will be assigned zero probabilities if they occur in a document. Moreover, statistical models lack generalization abilities compared to neural models.

Neural Models

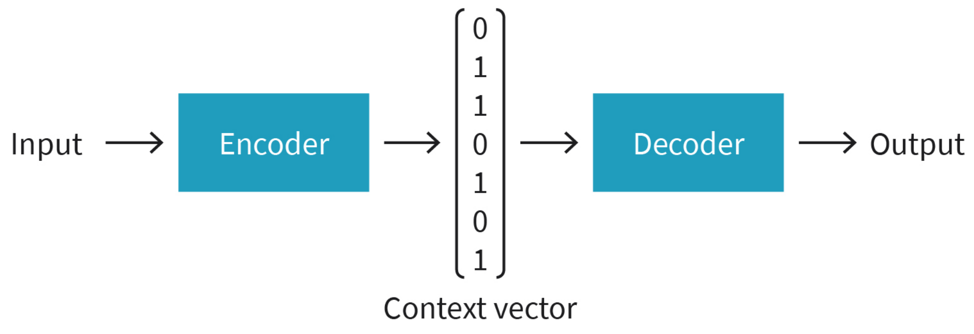
In recent years, neural models for NLP have become increasingly popular. Neural NLP models are based on deep learning strategies, such as recurrent neural networks (RNN) or convolutional neural networks (CNN).

The architecture of CNNs is based on several convolution kernels. For every layer, the convolution kernel is slid over the input matrix to generate a feature map, i.e., a filtered version of the input matrix. In the subsampling layers, the dimensionality of the feature maps is further reduced until, in the last layers, we receive a feature representation that is reduced on a very high level. This feature representation can then be passed by the fully connected layer to a layer of artificial neurons. These neurons learn how these high-level features can be mapped to the output classes. CNNs are mainly applied in computer vision tasks (Valueva et al., 2020). However, for NLP tasks, they have been shown to perform well when used for learning different n-gram patterns from a word-embedding matrix in sentence classification (Kim, 2014).

In RNNs, connections between nodes allow the output of certain nodes to affect the subsequent input into the same node. This makes it possible to model temporal correlations. RNNs have an internal memory, which allows them to process input sequences of variable length (Tealab, 2018). In contrast to CNNs, RNNs are therefore able to preserve the sequential order of a text, which makes them more suitable for most NLP tasks. The major drawback of RNNs compared to CNNs is that computations cannot be parallelized, which can slow the training down (Shankar & Parsana, 2022).

Many deep learning models are based on an encoder-decoder architecture. In an encoder-decoder architecture, the encoder converts the input text into a vector, which encapsulates all important information from the input sequence. The decoder then takes the information from the encoded vector and converts it back to the original representation. In most cases, the decoder will use the same network structure as the encoder (e.g., RNNs) – just in the opposite direction. The figure below shows a simplified version of a typical encoder-decoder architecture.

Figure 27: Simplified Encoder-Decoder Architecture



Source: Kristina Schaaff (2023).

If we look, for instance, at an example from machine translation where a sentence has to be translated from English to German, the encoder will first encode the English sentence to a feature vector that holds all information about the original sentence. In the next step, output from the encoder is passed to the decoder, which will then translate the information to German.

In an encoder-decoder architecture, it is possible to train the encoder with the output of the decoder. For this purpose, the output of the decoder is compared to the input of the encoder. This automatic way to train the model is referred to as an autoencoder architecture (Hubens, 2018).

In 2017, Google introduced transformer models, which are currently the most powerful models for NLP tasks. Transformer models are based on an encoder-decoder architecture. In contrast to traditional encoder-decoder models, however, they rely on **self-attention** mechanisms instead of using CNNs or RNNs (Vaswani et al., 2017).

Self-attention
The concept of self-attention in NLP relates to the relationship between different positions of a word sequence when computing a representation of a sentence.

To understand the concept of self-attention, let us look at the following example sentences:

"I moved from Munich to Berlin because I like it there."

In this sentence, we know that "there" refers to Berlin, while in the following sentence, "there" refers to Munich:

"I moved from Munich to Berlin because I did not like it there."

The example illustrates how important it is to identify the correct relationships between the parts of a sentence for correct comprehension.

Transformer models are not only significantly faster to train than RNN- and CNN-based models; they also outperform those models if it comes to accuracy in tasks like machine translation (Vaswani et al., 2017).

Nowadays, there are a large number of pre-trained models available based on the transformer architecture, such as Bidirectional Encoder Representation from Transformers (BERT; Devlin et al., 2019), a Robustly optimized BERT pretraining Approach (RoBERTa; Liu et al., 2019), DistilBERT (Sanh et al., 2019), and XLNet (Yang et al., 2019).

SUMMARY

Over the years, a large number of techniques have been developed for NLP tasks. Early systems were built on sets of rules that tried to reproduce the way humans produce sentences. Later, statistics-based systems emerged, which follow a more data-driven approach.

Regular expressions are a powerful tool that can be used in a large number of NLP tasks, such as preprocessing or pattern matching in search queries. Combining regular expressions can help answer complex search queries.

An important approach in text understanding is the use of n-grams, which can help determine the meaning of a whole text.

To be able to process data by computer, data vectorization is an important step. The simplest approach is the BoW model. However, the BoW model does not perform well for more complex tasks. Therefore, approaches like Word2Vec or TF-IDF can be used to build word vectors. To build sentence vectors, there are approaches like Skip-thought or USE.

Nowadays, there are a large number of pre-trained NLP models available. For text preprocessing, the most popular models are spaCy and NLTK. Statistical models are built on probability distributions, while neural models use more sophisticated approaches, such as autoencoder architectures.

UNIT 5

APPLICATION SCENARIOS

STUDY GOALS

On completion of this unit, you will be able to ...

- describe typical tasks and challenges in automatic speech recognition and speech synthesis.
- explain how machine translation works.
- discuss the most important techniques for information extraction.
- explain how NLP can be used for sentiment analysis and chatbots.
- know the most important libraries for NLP in Python.

5. APPLICATION SCENARIOS

Introduction

This unit introduces some areas of application of NLP. We will start with speech recognition and synthesis, which are the basis for many NLP tasks. After that, you will learn more about machine translation.

In the following, you will learn about the most important concepts of information extraction – the process of automatically extracting structured information from a text. In this context, you will learn more about named entity recognition, relationship extraction, and coreference resolution.

In the following section, we will look at the basics of sentiment analysis and the challenges that must be faced when trying to extract emotional information from a text. The last application area we are going to focus on is chatbots – a domain that has been growing massively in recent years, as chatbots can bring many potential savings across all industries.

The unit closes with an introduction to how you can build your own NLP project in Python. For this purpose, you will also get to know NLTK and spaCy, which are commonly used frameworks for NLP projects. Finally, we will give some examples of how to implement selected NLP concepts.

5.1 Speech Recognition and Synthesis

Speech recognition and synthesis are important parts of natural language processing (NLP) applications. Therefore, we now want to dive deeper into automatic speech recognition (ASR) – also known as speech-to-text (STT) – and speech synthesis, which is often also referred to as text-to-speech (TTS).

Speech Recognition

Using speech for communication – be it with other people or with machines – is much more intuitive than using text. Not only can we use our hands for other activities while speaking – speech on average actually transports information three times faster than typing (Ruan et al., 2018). ASR can be used to transcribe spoken language (i.e., speech) into text. It is a subfield at the intersection of computer science and computational linguistics (Soni, 2019, p. 257) and offers a wide range of application scenarios, such as text dictation or using speech to control voice assistants.

In recent decades, there have been a lot of improvements in ASR. Therefore, nowadays it is not only used in the professional but also in the private environment. Even though the quality of transcription still does not compare to the quality of human transcribers, it is normally faster to first use ASR and then perform manual post-processing instead of transcribing everything manually. This semi-automated process of speech transcription can save a lot of time and money.

As in all other areas of artificial intelligence (AI), the quality of the training data has a massive influence on the quality of the transcribed speech. The best quality can still be achieved when using domain-specific monolingual text corpora.

In the past, ASR systems typically combined an acoustic model, a pronunciation dictionary, and a language model. In recent years, end-to-end systems for ASR that are based on deep learning have become increasingly popular. End-to-end in the context of ASR means that a system directly maps an input sequence of acoustic features that have been derived from the speech signal to a sequence of words or **graphemes** (Wang & Li, 2019).

Grapheme
In written language, a grapheme is the smallest meaningful unit that differs between words.

Typical tasks in ASR

Considered in itself, ASR is only about the automatic transcription of speech signals. However, there are many tasks that include ASR as a part of another – more complex – task. This can, for instance, be speech-to-speech translation; voice assistants; speaker recognition; or other related tasks, such as speaker diarization.

In speaker recognition, speaker-specific information that is included in the speech waves is used to recognize who is currently speaking (Zhang, 2000, p. 179). Speaker diarization automatically determines who was speaking at what time in an audio or video recording (Anguera Miro et al., 2012, p. 356).

Let us now look at the automatic transcription of speech. Nowadays, speech-to-text services are commonly used, not only by organizations but also by individuals. The dictation capability many systems provide nowadays is probably one of their most important advantages, as this allows a user to easily insert text into documents or control devices. Recent speech recognition technology can convert speech into text as fast as the words are spoken (i.e., in “real-time”), which can massively increase the process of writing documents – even though, in most cases, it is still necessary to manually post-process the produced text.

Voice assistants are another steadily growing application area for ASR. Nowadays, they are integrated into almost every smartphone, in smart speakers, and in modern cars. Especially when driving a car, using a voice assistant can be very helpful, as the driver does not have to remove their hands from the steering wheel.

Challenges in ASR

As every person speaks differently and we are not always in a quiet environment, there are numerous challenges in ASR. These challenges include noise and channel variability, speaker anatomy and gender, accented and non-native speech, and homophones.

Noise and channel variability refers to effects such as echo, or other background noises such as music. Therefore, algorithms can be applied that focus on the frequency bands of the speech signal.

Speaker anatomy and gender have a considerable influence on how speech sounds. For instance, men have longer vocal tracts than women or children, which, therefore, leads to a lower fundamental frequency in their speech signal. To address this issue, vocal tract length normalization can be used (Garau et al., 2005). If vocal tract length normalization is used, a parameter that transfers the spectrum toward the spectrum of an average vocal tract is applied to the speech signal (Saheer et al., 2012, p. 2135).

Non-native and accented speech can cause a shift of parameters in the feature space. Commonly used methods to tackle this challenge are maximum a posteriori (MAP) adaption and maximum likelihood linear regression (MLLR). In MAP adaption, "adapted model parameters are estimated separately for each of the 'styles' of speaking, and then interpolated using a global interpolation weight" (Tomokiyo & Waibel, 2001, p. 3). "In MLLR adaption, transformation classes are defined, and model parameters of the entire class are shifted in the same direction" (Tomokiyo & Waibel, 2001, p. 4).

Homophones are words that are pronounced similarly but are spelled in a different way and also have a different meaning. For example, the words "rows" and "rose" are pronounced in a similar way. As it is not possible to distinguish homophones at an acoustic level, language models that contain contextual information are required to be able to identify which orthographic form of a word is correct.

Speech Synthesis

Speech synthesis or text-to-speech (TTS) is about producing human speech in an artificial way based on a given text (Tan et al., 2021), i.e., a written text is transformed into spoken language. In speech synthesis, several disciplines are combined: computer science, linguistics, acoustics, and signal processing (Ning et al., 2019). Nowadays, TTS is integrated into many applications of our everyday life – be it voice assistants, navigation apps, speech-to-speech translation, or news readers. TTS can also be used to assist people with reading disabilities or vision impairment by reading a written text (Isewon et al., 2014) and enables people to communicate despite restrictions in speaking – Professor Stephen Hawking was a prominent example of this application scenario (Medeiros, 2015).

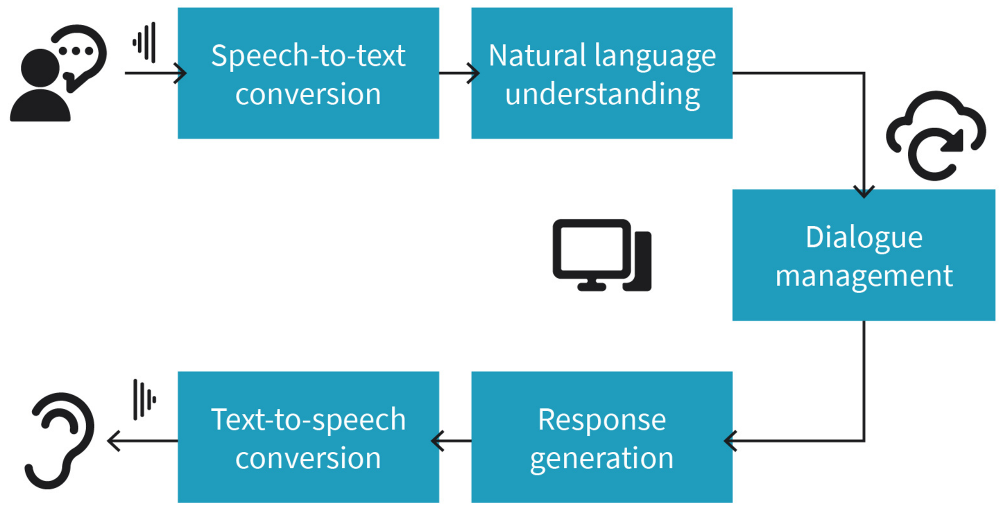
Typical tasks in speech synthesis

The most common applications for TTS are voice assistants and speech-to-speech translation.

A speech-to-speech translation system needs both ASR and speech synthesis. The ASR component first needs to convert the speech into text before a machine translation system can translate the text. The output of the machine translation process will again be a text, which has to be converted into a speech signal. Since computational power has been increasing massively in recent years, speech-to-speech translation can now be done almost in real-time.

Also, digital voice assistants include ASR, as well as speech synthesis. The figure below illustrates how a voice assistant can be constructed.

Figure 28: ASR and Speech Synthesis in a Voice Assistant



Speech-to-text conversion
Natural language understanding
Dialogue management
Response generation
Text-to-speech conversion

Source: Kristina Schaaff (2023).

Typically, a user will speak a voice command, which will be converted into text by an ASR component. A natural language understanding component processes the text and passes it to the dialogue management component. The dialogue management component will produce an appropriate reaction: If, for instance, the user asks a question, the dialogue management component can look for an appropriate answer. In the next step, the response generation component will generate an answer in a textual representation, which will then be transformed into speech by the TTS component.

Challenges in speech synthesis

When techniques like parametric speech synthesis or concatenative speech synthesis are used, a grapheme-to-phoneme conversion is necessary. This will convert every word into its corresponding pronunciation, i.e., the phonetic transcription.

Parametric speech synthesis simulates the process of how speech is produced in the human vocal tract to approximate the parameters that generate speech (Ning et al., 2019). Those parameters can, for instance, be the duration, the fundamental frequency F0, or the magnitude spectrum. In concatenative speech synthesis, waveforms are directly concatenated from a speech waveform database to output a continuous speech stream (Ning et al., 2019).

For grapheme-to-phoneme conversion, the pronunciation can be looked up in a dictionary. In the past, these dictionaries have often been generated manually, which has been a time-consuming process. Data-driven grapheme-to-phoneme converters that are trained on existing word-pronunciation pairs can help tackle this issue by directly providing pronunciations for words (Schlippe et al., 2014). There are open source tools, such as Sequiter G2P (Bisani & Ney, 2008), which can be used for the conversion.

Evaluation of speech synthesis systems

There are different aspects that can be evaluated for TTS systems: intelligibility, naturalness, preference, and comprehension.

Intelligibility reflects how accurately each word is produced. Naturalness refers to the quality of the generated speech signal in terms of pronunciation, timing, and emotions. Preference is about the choice of a listener between speech synthesis tools. Finally, comprehension deals with how well a message is understood.

5.2 Machine Translation

Machine translation (MT) has always been an important subfield of NLP, starting with early attempts of translation from Russian to English during the Second World War. The goal of MT is to automatically translate text or speech from one language to another language.

Even though there have been significant advances in MT in recent decades, the quality of MT does not compare to the quality of human translations, especially when it comes to understanding context or cultural connotations. Nevertheless, it has nowadays become faster to combine MT with manual post-processing by a human expert than to have the whole text translated by a human. Moreover, the output quality of the translation depends strongly on the quality of the data that are used for training. As in any other area of NLP, the results will be better if the model is trained with domain-specific data.

Statistical machine translation
In SMT, translations are made based on statistical models, which have usually been built from bilingual text corpora.

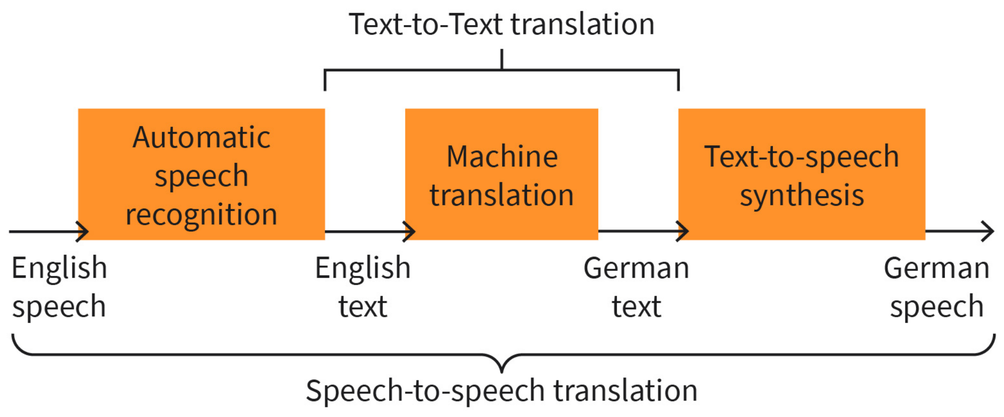
Neural machine translation
In NMT, artificial neural networks are used to model a sequence of words.

Early MT approaches were based on rule-based translation. However, these models did not perform well. Only as **statistical machine translation** (SMT) emerged did MT start to regain interest. In recent years, **neural machine translation** (NMT) has made rapid progress and is now the most commonly used method (Koehn & Knowles, 2017b).

MT covers both text-to-text and speech-to-speech translations. Text-to-text translation can help translate websites or text documents, or accelerate the translation process by professional translators. By extending text-to-text translation by a component for automatic speech recognition and text-to-speech synthesis, a speech-to-speech translation system can be created.

The following figure illustrates how text-to-text translation and speech-to-speech translation are connected.

Figure 29: Connection Between Text-to-text Translation and Speech-to-Speech Translation



Source: Kristina Schaaff (2023).

Both text-to-text and speech-to-speech translation have gained increasing significance in recent years. The dramatically increasing number of video chats and virtual conferences has further accelerated the whole process. By using applications such as the Skype translator, MT can help to bridge language barriers between speakers.

Two of the biggest challenges MT has to deal with today are **domain mismatch** and under-resourced languages. Therefore, domain adaption is an important step when an MT system is developed for a specific use case. To deal with under-resourced languages, techniques such as **pivot MT** can be used.

Domain mismatch
Depending on the domain, words and sentences can have different meanings and, therefore, translations.

Pivot MT
Using a pivot language to bridge the gap between source and target language.

5.3 Information Extraction

The goal of information extraction (IE) in NLP is to automatically extract structured information from a given text (Adnan & Akbar, 2019). It is closely related to information retrieval (IR), and both terms are often used synonymously. However, there is a key difference between them: While for IE, the relevant facts that are of interest are specified beforehand, IR is about discovering information or documents that contain facts of interest that are not known in advance (Bach & Badaskar, 2007, p. 1).

Typical IE tasks include named entity recognition (NER), relationship extraction, and coreference resolution. Those tasks are often key for more complex NLP tasks, such as natural language understanding, question answering, digital assistants, or text summarization (Singh, 2018). NER deals with the classification of named entities from an unstructured text into categories, such as date or location (Li et al., 2022, p. 50). Relationship extraction – which is also often referred to as relation extraction – deals with the identification of semantic relationships between the entities of a text (Bach & Badaskar, 2007, p. 1). Coreference resolution is used to identify words in a text that refer to the same entity (Clark & Manning, 2016).

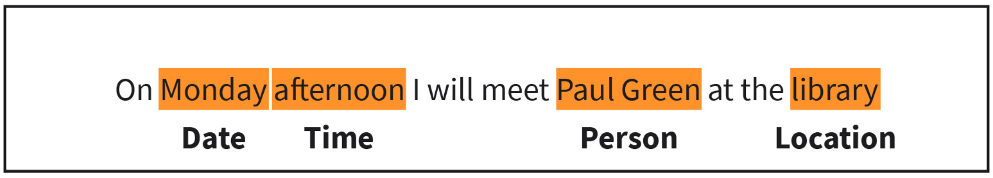
In the following, we will look more deeply into these tasks.

Named Entity Recognition

As previously mentioned, the main goal of NER is to locate named entities in an unstructured text and assign them to categories such as time and date expressions, locations, organizations, quantities, names, etc. It has been shown, that in some cases, NER can improve the results of machine translation (Babych & Hartley, 2003). It is also commonly used in tasks where it is important to understand the content of a text. Therefore, for tasks like data organization and text analysis, NER is often the first step toward further analyses.

In the figure below, you can see an example sentence where NER is used to identify entities such as person and location from a given sentence.

Figure 30: Example of Named Entity Recognition



Source: Kristina Schaaff (2023).

In the example, the following types of entities are identified:

- Date: Monday
- Time: afternoon
- Person: Paul Green
- Location: library

There are different ways named entities can be labeled. A commonly used format is the BIO prefix scheme as described by Ramshaw and Marcus (1995). The BIO labels (sometimes also referred to as IOB) indicate the following positions of a named entity:

- Beginning (B): beginning of an entity
- Inside (I): continuation of an entity
- Outside (O): token that does not belong to an entity

One commonly used NER corpus is the CoNLL-2003 Shared Task corpus (Tjong Kim Sang & Meulder, 2003). This corpus is based on a set of news articles from Reuters that have been annotated by hand. The following labels are used in the CoNLL dataset:

[“B-LOC,” “I-LOC,” “B-MISC,” “I-MISC,” “B-PER,” “I-PER,” “B-ORG,” “I-ORG,” “O”]

B, I, and O refer to the BIO prefix scheme, while the other tokens have the following meanings:

- LOC: location
- MISC: miscellaneous name
- PER: person
- ORG: organization

For our above example, the annotation would look like this:

Table 4: CoNLL Annotation Example

On	O
Monday	O
afternoon	O
I	B-PER
will	O
meet	O
Paul	B-PER
Green	I-PER
at	O
the	O
library	B-LOC

Source: Kristina Schaaff (2023).

Using the BIO scheme prefixes, we can distinguish whether we have one person (Paul Green) or two persons (Paul and Green) based on the annotations. The word tokens that do not belong to any of the four named entity categories are labeled with ‘O’.

Examples for NER include all domains where it is useful to organize text in categories. This can, for instance, be the categorization of tickets in customer support according to their topics; depending on the categorization, tickets can then be automatically forwarded to the responsible expert. Another example would be the anonymization of data according to privacy regulations. If personal data are automatically identified and reduced, this can help to save costs. If the quality of the data is high enough, it is no longer necessary to manually cleanup the data. NER can also help to reduce the workload of HR in application processes by automatically extracting information from the applicants’ resumés (Zimmermann et al., 2016).

One of the biggest challenges in NER is that a large amount of annotated data is required for training. Moreover, the model will later be limited to the specific task it has been trained on.

Relationship Extraction

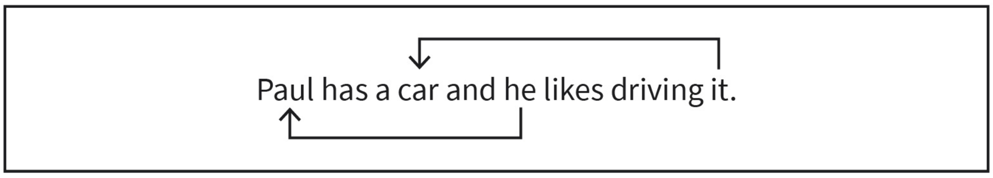
In relationship extraction, the goal is to extract semantic relations between the entities of a text (Bach & Badaskar, 2007, p. 1). The extracted relations are usually binary, such as "FATHER-OF (Darth Vader, Luke Skywalker)" or "LOCATED-IN (International University, Germany)."

One important application domain for relationship extraction is question answering. For example, if a system receives the question "What is the capital of France?", the system might search for a relational tuple that matches the pattern "CAPITAL-OF (France, ???)."

Coreference Resolution

In coreference resolution, words that refer to the same entity are identified (Clark & Manning, 2016). This is important to properly understand the context of a text. For instance, in the sentence "Paul has a car and he likes driving it," the word 'he' refers to the entity Paul and the word 'it' refers to the car. The figure below illustrates the relationship between those entities.

Figure 31: Coreference Resolution



Source: Kristina Schaaff (2023).

5.4 Sentiment Analysis

So far, we have mostly been dealing with the objective aspects of text and speech, such as the recognition of single words or the translation of a text from one language to another. In human interaction, besides the actual vocabulary and grammar, a text contains much more information, which is often expressed "between the lines." This is where sentiment analysis comes into play. Sentiment analysis deals with the analysis of the subjective aspects of a text (Nasukawa & Yi, 2003). This can, for instance, be the mood of an author or a tweet on Twitter. As in topic identification, sentiment analysis is a typical text classification problem. However, while topic identification is about the identification of objective aspects of a text, sentiment analysis deals with subjective features, such as emotions or moods. There is a wide range of possible applications for sentiment analysis. One field that has recently become the focus of much attention is customer sentiment analysis. Being able to track how customers feel about a certain product over time can reflect how people react to a change in a product or service. Moreover, sentiment analysis makes it possible to analyze how external factors like a global crisis or relevant news influence the perceptions of customers.

Using social networks, such as Twitter, Instagram, or Facebook, it is easy to collect a huge amount of data about a certain product. If a company is able to gain a better understanding of the needs of its customers, products and services can be modified accordingly.

Types of Emotions

Emotions can be categorized in different ways. The two basic approaches are categorical models and dimensional models. While categorical models assume a clearly defined set of basic emotions, such as anger, joy, or fear (Ekman, 1992), dimensional models represent emotions as points in a multidimensional space. The advantage of the dimensional representation is that, in contrast to categorical models, the emotions do not have to be classified within predefined boundaries but rather in a continuous space within the emotion dimensions.

According to Bradley and Lang (1994), the dimensions of emotions can be defined as follows:

- arousal: quantitative degree of activation (calm versus excited)
- valence: quality of the emotion (negative versus positive)
- dominance: degree of control a person has over a situation (weak versus strong)

The most frequently used dimensions are arousal and valance, as dominance has a comparatively small influence for most of the variance on emotional scales (Russell, 1979).

Sentiment Analysis Tasks

A common task in sentiment analysis is polarity detection. Polarity detection refers to the valence dimension of emotions and usually categorizes a text as positive or negative. The number of categories ranges from two categories (being only positive or negative; Turney, 2002) up to 5-star rating scales (Snyder & Barzilay, 2007). There is also research toward the identification of categorical emotions, such as anger, sadness, disgust, enjoyment, fear, or surprise (Ho et al., 2019).

Another common task is the identification of the subjectivity/objectivity of a text (Pang & Lee, 2008). This task can sometimes be even more challenging than polarity detection (Mihalcea et al., 2007). One of the big challenges of this task is the question of how subjectivity is defined. Nevertheless, removing subjective sentences from a text or document can help to increase the accuracy of polarity detection (Pang & Lee, 2004).

Intensity ranking is another sub-discipline of sentiment analysis and refers to the intensity with which an emotion is expressed in a text. Depending on how intensely an emotion is expressed in a text, the meaning of a sentence can vary significantly (Sharma et al., 2017). Therefore, analyzing not only the polarity of an emotion but also the intensity can add meaningful information to sentiment analysis.

Challenges

As sentiment analysis and emotion detection work with user-generated content, there are some big challenges to be faced when trying to identify the user state:

- negation
- irony/sarcasm
- multipolarity

When detecting the sentiment of a statement, negation is a big challenge, as it can invert the meaning of a whole statement. One problem is that negation can not only be explicit, using words like “not,” but also implicit, using prefixes like “non-,” or “dis-” or suffixes like “-less.” Moreover, language constructs like double negation can be easily misinterpreted. While, in mathematics, double negatives cancel out, depending on the context, double negation might even make the negation more intense. Therefore, it is important to consider negation in the model when performing a sentiment analysis to increase the accuracy of the system (Sharif et al., 2016).

Sarcasm and irony pose another challenge to sentiment analysis. Especially in the context of social media, sarcasm is quite common. Even for humans, it is sometimes challenging to properly recognize sarcasm. Therefore, for a machine, it can be even more difficult. As an example, we will use the following sentence:

“Wow, you have an iPhone 5?”

If we look only a few years back to the year 2012 when the iPhone 5 was released, this statement would have been perceived as genuinely enthusiastic and therefore non-sarcastic. Nevertheless, if you hear this statement nowadays, it is easy for a human to tell that it is intended to be sarcastic. This example illustrates why dealing with sarcasm still remains a challenging task, even though there has recently been some success when applying methods from deep learning for sarcasm detection (Ghosh & Veale, 2016).

Multipolarity of text is another big challenge. Multipolarity means that one text can consist of parts with different polarities, i.e., the sentiment of some parts can be positive while other parts are negative. To illustrate multipolarity, let us look at the customer review in the figure below.

Figure 32: Multipolarity Example

“I got my new phone today.
The quality of the display
is awesome, but the audio
quality is really poor.”

Source: Kristina Schaaff (2023).

The first sentence is neutral. In the next sentence, the first part about the display is very positive, while the second part about the audio quality is negative. There are several approaches to handle reviews like this. For instance, we could simply calculate the average of the sentiment. However, this will inevitably lead to a loss of information. Therefore, a better approach would be to split the review into several parts and analyze the parts of the sentence separately.

Evaluation of Sentiment Analysis Tasks

One major challenge when it comes to the evaluation of algorithms for sentiment analysis is that the classification of the subjective elements of a text is – as the name already indicates – a subjective task. While, for tasks like information extraction, there is an objective ground truth for training and evaluation of the task, this is more difficult with subjective aspects. The classification of emotions can even be a challenging task for humans. Therefore, accuracies close to 100 percent are not realistic, as – even among humans – there would be a disagreement of about 20 percent (Roebuck, 2012).

5.5 Chatbots

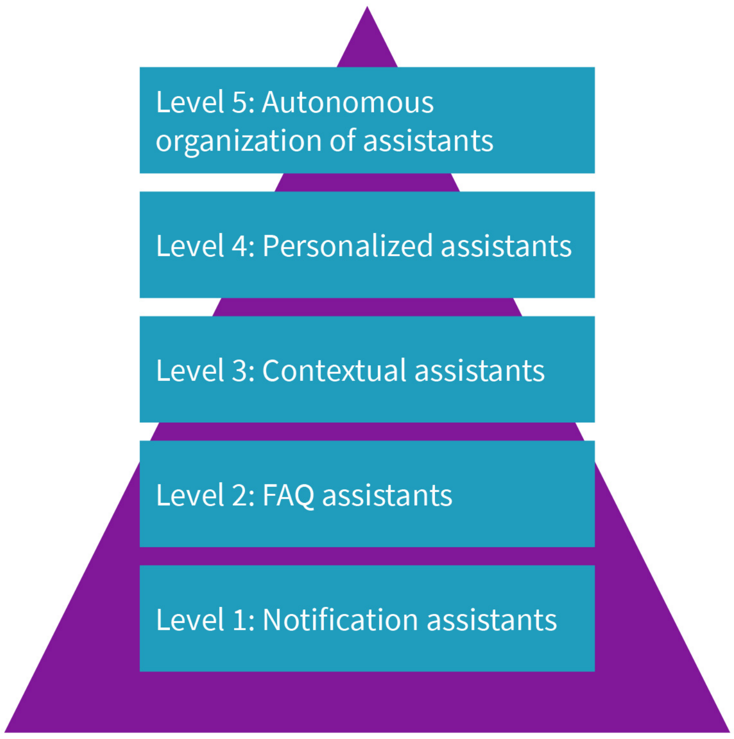
Chatbots – often also referred to as conversational AI – are dialogue systems that are based on textual communication. Chatbots make it possible to interact with a computer using text in natural language. Depending on the input, the user will receive a reply that is also in natural language. Some chatbots mimic a certain personality or character in combination with an avatar or image. A popular example of a chatbot was ELIZA (Weizenbaum, 1966), which simulated a psychotherapist.

Chatbots often appear in messenger apps, such as Facebook, or in website chats. Moreover, digital assistants, such as Siri, Alexa, or Google Assistant, are based on chatbots.

Levels of AI Assistants

Chatbots are an important part of AI assistants and can be categorized into five levels (Nichol, 2018). The five levels are illustrated in the figure below.

Figure 33: Five Levels of AI Assistants


Level 5: Autonomous organization of assistants
Level 4: Personalized assistants
Level 3: Contextual assistants
Level 2: FAQ assistants
Level 1: Notification assistants

Source: Kristina Schaaff (2023).

Notification assistants (Level 1) interact with the user only in a unidirectional way. Typically, they are used for tasks likes notifications about updates or events.

In contrast to notification assistants, frequently asked questions (FAQ) assistants (Level 2) can interact bi-directionally with a user. If the user asks a question, they can interpret the request and find an appropriate answer from a knowledge base.

Contextual assistants (Level 3) extend the communication by being context-aware of the conversational history with the user. For instance, they might remember your past purchases in an online shop.

Personalized assistants (Level 4) get to know the user over time. They will, for instance, know, based on the context, when it is a good time to interact. Moreover, personalized assistants are able to learn the user’s preferences and will, therefore, be able to provide a personalized interface.

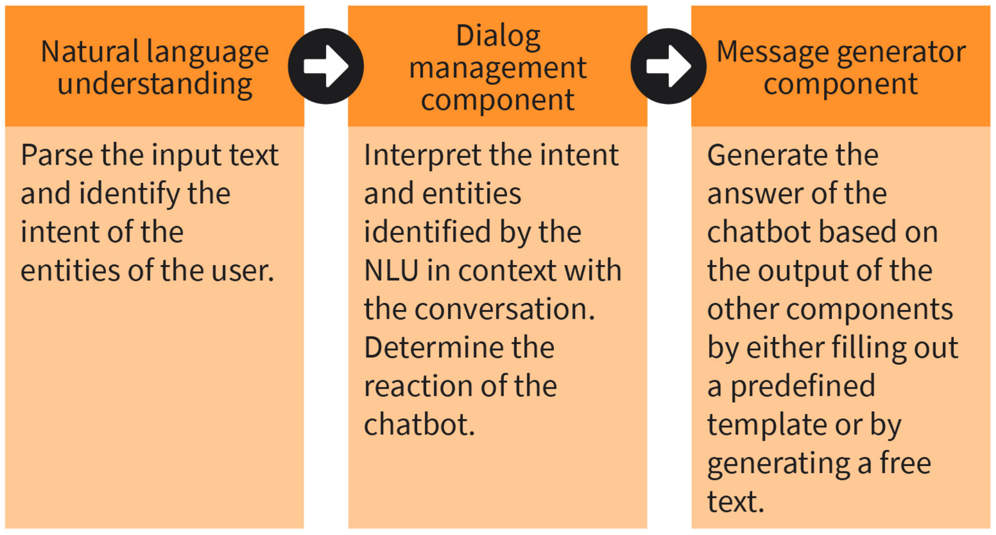
Finally, autonomous organizations of assistants (Level 5) are a group of assistants that have information about every customer personally. They can independently run broad areas of an organization, such as sales, marketing, or lead generation.

At present, most assistants are still on the level of notification assistants. However, there are already a few applications, such as the Google Assistant, that can be categorized as contextual assistants. It is quite likely that, in the future further, levels of chatbots will evolve.

Components of Chatbots

In general, chatbots consist of three components, which are illustrated in the figure below.

Figure 34: Components of a Chatbot


Natural language understanding
Parse the input text and identify the intent of the entities of the user.
Dialog management component
Interpret the intent and entities identified by the NLU in context with the conversation. Determine the reaction of the chatbot.
Message generator component
Generate the answer of the chatbot based on the output of the other components by either filling out a predefined template or by generating a free text.

Source: Kristina Schaaff (2023).

Techniques

The implementation of chatbots can either be rule-based or statistics-based. We will look at both techniques next.

Rule-based chatbots

Rule-based chatbots parse the textual input using a set of regular expressions. Based on the regular expressions, the intent of a user can be determined. A common way to specify conversation rules in a rule-based way is Artificial Intelligence Markup Language (AIML). AIML is based on Extensible Markup Language (XML) and is available open source. There are AIML interpreters for a large variety of programming languages.

The figure below is an example how a conversation rule in AIML might look.

Figure 35: Example of AIML

```
<category>
   <pattern>WHAT IS *</pattern>
   <template>
      Here you can find information about
      <star/>.
      <button>
         <text>Click</text>
         <url>https://en.wikipedia.org/wiki/
         <star/></url>
      </button>
   </template>
</category>
```

Source: Kristina Schaaff (2023).

The tags of this conversation rule are specified as follows:

- <category>: container for the conversation rule.
- <pattern>: text pattern of the user query. The * represents a sequence of characters.
- <template>: specifies the answer.
- <star/>: is replaced by the text from the user query.

In our example, the system might receive the request “WHAT IS ARTIFICIAL INTELLIGENCE?” The input will first be normalized by removing interpunctions and ignoring the case. When the question matches the pattern, the answer can be displayed as specified in the template. The template in our example will first print the text “Here you can find information about artificial intelligence,” replacing the <star/> tag by the text which matched the pattern. Additionally, it will show a hyperlink that will redirect to the related Wikipedia article.

Statistics-based chatbots

Chatbots can also be built using statistical machine learning techniques to train the chatbot using example conversations.

To illustrate statistics-based approaches, we will now have a look at the conversational AI toolkit Rasa (Bocklisch et al., 2017), which is available open source.

The first phase is intent classification, which is about determining the user intent. For instance, “Hi” or “Good evening” can be seen as an indicator for a greeting. We could call this intent `greeting`. The intent behind questions like “How are you?” or “How are you feeling today?” is to know how a person feels. This intent could therefore be classified as `feeling`. A reply like "not so well" or "sad" could then be mapped to the intent `negative_mood`, while a reply like "I am great" or "fine" could be mapped to `positive_mood`.

A representation of this conversation in Rasa can be modeled as follows:

```
## intent:greeting

• hi
• good morning
• good evening

## intent:feeling

• how are you
• how are you feeling today
• are you ok

## intent:negative_mood

• sad
• not so well
• unhappy

##intent:positive_mood

• fine
• i am great
• awesome
```

The intent classification is the base of the dialogue component of our system. Actions are another important concept. Actions define how – once the intent of a user has been identified – the chatbot will react. They can range from simple replies to more complex answers like getting data from a knowledge base or external service, such as Wikipedia or a news website.

For our example, we will define different actions for the bot. The first action `utter_greet` will make the chatbot reply to a greeting of a user with the sentence "Hi, how are you?" The second action models a reaction to the user's reply depending on if it's positive or negative: `utter_sad` will be `Sorry to hear that`, while `utter_happy` will be "Happy to hear that."

Now that we have defined the answers, we need to model the dialogue management, which connects intents and actions. The training of the dialogue management model in Rasa is done using sample conversations, which are referred to as stories. A story could, for instance, look like the following conversation:

```
User: Hi
```

```
Chatbot: Hi, how are you?

User: I am great

Chatbot: Happy to hear that
```

If we represent the story as a set of intents and actions, the story could look as follows:

```
## positive conversation

* greeting

- utter_greet

* positive_mood

- utter_happy
```

Based on a sequence of intents, the dialogue management module will be trained to predict the next action. In contextual assistants, not only the last intent will be analyzed but also the history of, for instance, the last five intents. In the example above, as an input we have the features `greeting` and `positive_mood`. The output we expect in this situation is `utter_happy`.

Of course, the chatbot can also use named entity recognition to extract information like dates, locations, or names from the input data. This technique is referred to as slot filling.

Use Cases

Use cases for chatbots are increasing continuously as their use offers many advantages. Integrating chatbots into a communication can save a lot of time and money, as chatbots are normally available seven days a week, 24 hours a day at comparatively low costs. If necessary, they can easily be scaled.

If conversational agents are used in customer service, they can help by, for instance, replying to the requests of customers. They can improve the customer experience by giving personalized product recommendations according to the needs of a user. If an artificial agent gets a request that is too complicated to handle, it can still be forwarded to a human support team.

If we look at company websites, many companies nowadays have integrated chatbots to ask sales-related questions to visitors who would otherwise remain anonymous. If a customer uses the chat, the chatbot can, for instance, present information about special offers or new products or even provide guidance on the navigation of the website.

In general, using chatbots can help to reduce the workload of human support staff, which can allow them to deal with more complex tasks.

5.6 NLP With Python

Now that you have learned the most important aspects of NLP and the related techniques, it is time to learn how to practically develop NLP applications. Therefore, in this section, you will learn more about how to use Python as a programming language, Jupyter Notebooks (Randles et al., 2017), and the most important NLP frameworks, such as the Natural Language Toolkit (NLTK; Bird et al., 2009) and spaCy (n.d.).

NLP & Python

Python is a general-purpose, high-level interpreted programming language. It is possible to use Python for many purposes – be it software development, web development, or data science. The syntax rules are quite simple, which makes it easy to obtain a code base that is easy to read.

There are many well-written data science libraries available. This has made Python very popular for everyone who wants to start out in the field of machine learning or data science.

There are a large number of integrated development environments (IDEs) available for Python, such as PyCharm, Sublime Text, or Spyder. Moreover, Jupyter Notebooks offer an excellent interactive integrated development environment (IDE) for software development in Python.

A Quick Start to Jupyter Notebooks

Jupyter Notebooks are often used for machine learning or data science projects. Jupyter Notebooks provide a web-based interactive development environment for Python. The web application is provided open source and can contain not only code but also narrative text, equations, and visualizations. The interpreter is web-based and makes it easy to structure the code into cells. Each of these cells can be run individually. Moreover, it is possible to add comments between the cells. This makes it easy to debug the code, generate visualizations, or apply changes to the code.

In this section, you will get a brief introduction on how to install Jupyter Notebook, create a notebook, and write code snippets and comments in that notebook.

To install Jupyter Notebook, we can use the related PIP package. For this purpose, we have to run the following command:

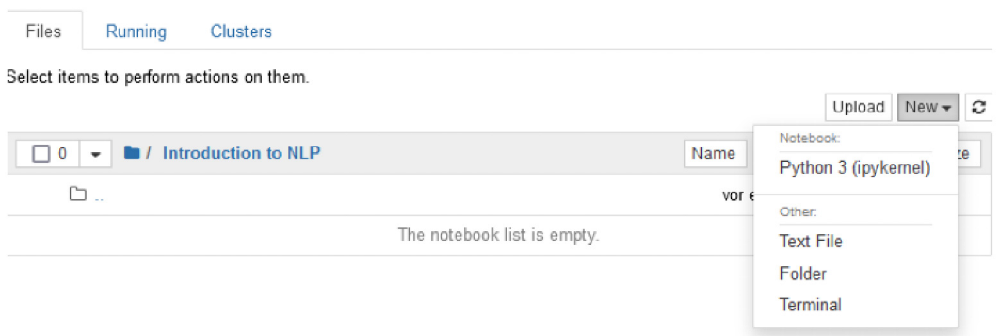
```
Pip install jupyter notebook
```

Once the installation is complete, the Jupyter Notebook can be launched with the following command:

```
Jupyter notebook
```

The Jupyter Notebook will then open in the default browser. You can create a new notebook by selecting the “New” dropdown on the top right of the window.

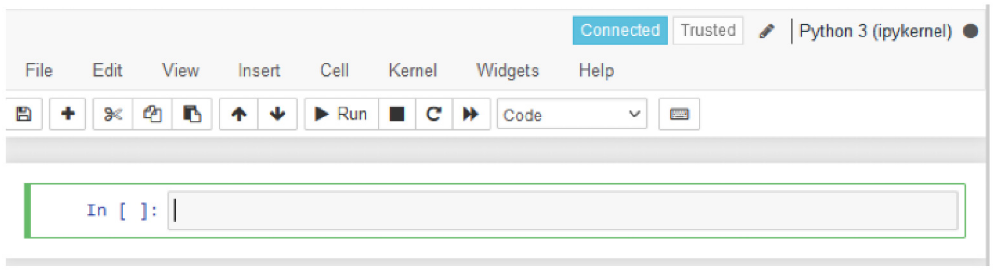
Figure 36: Creating a New Notebook



Source: Kristina Schaaff (2023), based on Jupyter (n.d.). 3-Clause BSD License.

Initially, the new notebook will contain one single empty code cell.

Figure 37: The New Notebook

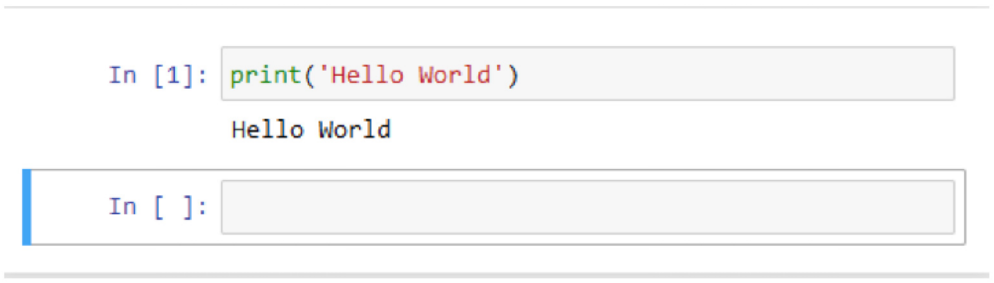


Source: Kristina Schaaff (2023), based on Jupyter (n.d.). 3-Clause BSD License.

In the code cell, you can then enter your code; you can execute the code by pressing Shift + Return.

If we start with a simple “Hello World” example, the result will look as shown in the following figure.

Figure 38: Hello World Example



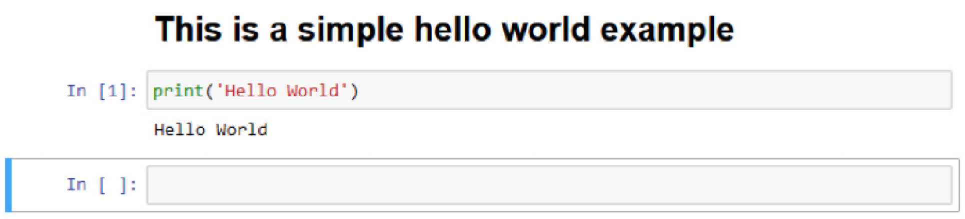
```
In [1]: print('Hello World')
        Hello World
In [ ]:
```

Source: Kristina Schaaff (2023), based on Jupyter (n.d.). 3-Clause BSD License.

If you want to add markdown to your code, you can do this by changing the cell type to "markdown." To change the cell type, you can either use the menu bar at the top or one of the many shortcuts. A list of shortcuts can be found in the top menu under Help > Keyboard Shortcuts.

Using markdown, you can add useful information about your application to your notebook to document and explain your work.

Figure 39: Markdown Example



This is a simple hello world example

```
In [1]: print('Hello World')
        Hello World
In [ ]:
```

Source: Kristina Schaaff (2023), based on Jupyter (n.d.). 3-Clause BSD License.

Additionally, when using Jupyter Notebooks, you can also export your code to any other format, such as HTML or Python script (.py). If you want to export your code, you can do this by selecting the option "Download as" from the "File" tab in the top menu bar.

Introduction to NLTK and spaCy

There are several frameworks for NLP. The most popular frameworks are NLTK and spaCy because of their ease of use and their functionalities. They can be useful for the implementation of a huge number of tasks, be it chatbots, sentiment analysis, text summarization, or entity extraction.

To install the packages, the Python package manager PIP can be used with the following commands:

```
pip install nltk
```

and

```
pip install spaCy
```

Additionally, both frameworks use external resources. NLTK uses task-specific datasets, while spaCy provides complete language models, which are available in different sizes for multiple languages. These resources have to be downloaded separately.

There are a few differences between the two frameworks. First of all, in NLTK, for a particular problem, there are a vast number of algorithms from which you can choose, while in spaCy, for a particular problem, you will only find the best state-of-the art algorithm according to the developers of spaCy. Moreover, input and output in spaCy is based on an object-oriented model centered around the corresponding document object. In NLTK, strings are used as input and output for the respective functions. Another important difference is that NLTK does not support word embeddings, while in spaCy vector-based word embeddings are used.

Implementing Selected NLP Concepts with spaCy

We now want to have a look at selected concepts from NLP and how they can be implemented in Python using spaCy.

In our example, we want to work with the small English language model which can be installed using the following command:

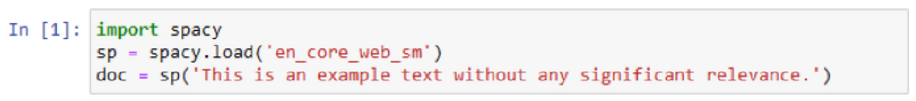
```
python -m spaCy download en_core_web_sm
```

Tokenization

Tokenization is often the first step that is performed in NLP tasks. Using spaCy, word and sentence tokenization can be implemented in the following steps:

1. Create a spaCy document: In the first line, we import the spaCy library. The English model is loaded in line 2, and in line 3 we create a document.

   Figure 40: Creating a spaCy Document

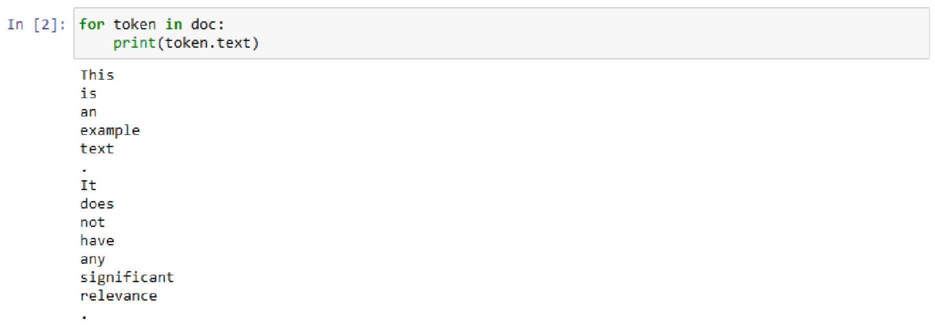
   

   ```
   In [1]: import spacy
           sp = spacy.load('en_core_web_sm')
           doc = sp('This is an example text without any significant relevance.')
   ```

   Source: Kristina Schaaff (2023), based on spaCy (n.d.). MIT License.

2. Extract the word tokens: In the first line, the word tokens are accessed by iterating over the document object. In line 2, the tokens are printed.

Figure 41: Tokenization in spaCy

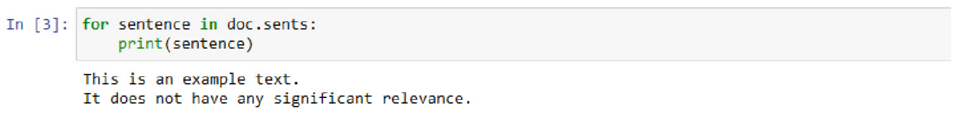


```
In [2]: for token in doc:
            print(token.text)

        This
        is
        an
        example
        text
        .
        It
        does
        not
        have
        any
        significant
        relevance
        .
```

Source: Kristina Schaaff (2023), based on spaCy (n.d.). MIT License.

3. Extract the sentence tokens: Similarly to the extraction of the word tokens in line 1, the tokens are accessed and then printed in line 2.

Figure 42: Sentence Tokenization with spaCy



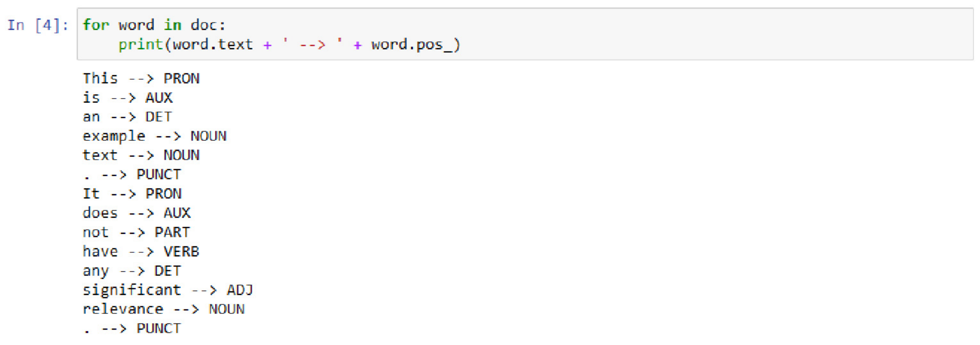
```
In [3]: for sentence in doc.sents:
            print(sentence)

        This is an example text.
        It does not have any significant relevance.
```

Source: Kristina Schaaff (2023), based on spaCy (n.d.). MIT License.

Part-of-speech tagging

In the next step, we want to use our example to perform POS tagging. Again, we will initialize our program with step one from the last example. To perform POS tagging, we will iterate over the document object doc (line one) and print the POS attribute for every token (line two).

Figure 43: Part-of-Speech Tagging with spaCy



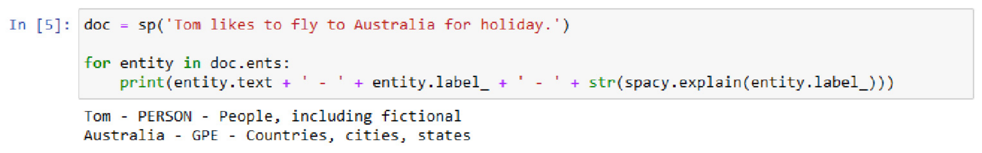
```
In [4]: for word in doc:
            print(word.text + ' --> ' + word.pos_)

        This --> PRON
        is --> AUX
        an --> DET
        example --> NOUN
        text --> NOUN
        . --> PUNCT
        It --> PRON
        does --> AUX
        not --> PART
        have --> VERB
        any --> DET
        significant --> ADJ
        relevance --> NOUN
        . --> PUNCT
```

Source: Kristina Schaaff (2023), based on spaCy (n.d.). MIT License.

Named entity recognition

Extracting named entity labels from a text using spaCy is also remarkably simple. For this purpose, we have to iterate over the entities from our document and in the next step for every token print the label attribute.

Figure 44: Named Entity Recognition with spaCy



```
In [5]: doc = sp('Tom likes to fly to Australia for holiday.')

        for entity in doc.ents:
            print(entity.text + ' - ' + entity.label_ + ' - ' + str(spacy.explain(entity.label_)))

        Tom - PERSON - People, including fictional
        Australia - GPE - Countries, cities, states
```

Source: Kristina Schaaff (2023), based on spaCy (n.d.). MIT License.

In addition to the label information, it is possible to print an explanation of why a certain word has been assigned to a certain entity.

SUMMARY

Speech recognition and synthesis are important components for systems that interact with humans via spoken language – be it in machine translation or when using voice assistants.

Using machine translation, a text or speech signal can automatically be translated into another language.

In order to properly process a text, relevant information has to be extracted. This can be done using techniques such as named entity recognition, relationship extraction, or coreference resolution.

Sentiment analysis is an important field of research in NLP that deals with the task of extracting the emotions of a user from a given text or speech signal. Another increasingly important area of application is chatbots.

NLP systems are often implemented using Python, which provides a large set of libraries to solve common problems. NLTK and spaCy are two Python toolkits that provide a huge set of algorithms and techniques that can be useful for all sort of NLP tasks.

BACKMATTER

LIST OF REFERENCES

Accountalive. (2021, April 22). *Simple time domain vs frequency domain* [image]. Wikimedia. https://commons.wikimedia.org/wiki/File:Simple_time_domain_vs_frequency_domain.svg

Adel, H., Vu, N. T., Kraus, F., Schlippe, T., Li, H., & Schultz, T. (2013). Recurrent neural network language modeling for code switching conversational speech. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 8411–8415). IEEE. https://doi.org/10.1109/ICASSP.2013.6639306

Adnan, K., & Akbar, R. (2019). An analytical study of information extraction from unstructured and multidimensional big data. *Journal of Big Data*, *6*(1). https://doi.org/10.1186/s40537-019-0254-8

Ali, A., Dehak, N., Cardinal, P., Khurana, S., Yella, S. H., Glass, J., Bell, P., & Renals, S. (2015). *Automatic dialect detection in Arabic broadcast speech.* arXiv. https://doi.org/10.48550/arXiv.1509.06928

Anguera Miro, X., Bozonnet, S., Evans, N., Fredouille, C., Friedland, G., & Vinyals, O. (2012). Speaker diarization: A review of recent research. *IEEE Transactions on Audio, Speech, and Language Processing*, *20*(2), 356–370. https://doi.org/10.1109/TASL.2011.2125954

Auer, P. (Ed.). (2001). *Code-switching in conversation: Language, interaction and identity*. Routledge.

Automatic Language Processing Advisory Committee. (1966). *Language and machines*. National Academies Press. https://doi.org/10.17226/9547

Babych, B., & Hartley, A. (2003). Improving machine translation quality with automatic named entity recognition. In *Proceedings of the 7th international EAMT workshop on MT and other language technology tools, improving MT through other language technology tools: Resources and tools for building MT* (pp. 1–8). Association for Computational Linguistics.

Bach, N., & Badaskar, S. (2007). *A review of relation extraction*. Carnegie Mellon University. https://www.cs.cmu.edu/~nbach/papers/A-survey-on-Relation-Extraction.pdf

Banko, M., & Brill, E. (2001). Scaling to very very large corpora for natural language disambiguation. In B. L. Webber (Ed.), *Proceedings of the 39th annual meeting on association for computational linguistics* (pp. 26–33). Association for Computational Linguistics. https://doi.org/10.3115/1073012.1073017

Beel, J., Gipp, B., Langer, S., & Breitinger, C. (2016). Research-paper recommender systems: A literature survey. *International Journal on Digital Libraries*, *17*(4), 305–338. https://doi.org/10.1007/s00799-015-0156-0

Bender, E. (2009). Linguistically Na¨ ive != language independent: Why NLP needs linguistic typology. In *Proceedings of the EACL 2009 workshop on the interaction between linguistics and computational linguistics: Virtuous, vicious or vacuous?* (pp. 26–32). Association for Computational Linguistics. https://aclanthology.org/W09-0106

Bennett, R., & Elfner, E. (2019). The Syntax–Prosody Interface. *Annual Review of Linguistics*, *5*(1), 151–171. https://doi.org/10.1146/annurev-linguistics-011718-012503

Besacier, L., Barnard, E., Karpov, A., & Schultz, T. (2014). Automatic speech recognition for under-resourced languages: A survey. *Speech Communication*, *56*, 85–100. https://doi.org/10.1016/j.specom.2013.07.008

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python*. O'Reilly. https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=443090

Bisani, M., & Ney, H. (2008). Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, *50*(5), 434–451. https://doi.org/10.1016/j.specom.2008.01.002

Bocklisch, T., Faulkner, J., Pawlowski, N., & Nichol, A. (2017). *Rasa: Open source language understanding and dialogue management.* https://doi.org/10.48550/arXiv.1712.05181

Bokamba, E. G. (1989). Are there syntactic constraints on code-mixing? *World Englishes*, *8*(3), 277–292. https://doi.org/10.1111/j.1467-971X.1989.tb00669.x

Bradley, M. M., & Lang, P. J. (1994). Measuring emotion: The self-assessment manikin and the semantic differential. *Journal of Behavior Therapy and Experimental Psychiatry*, *25*(1), 49–59. https://doi.org/10.1016/0005-7916(94)90063-9

Buolamwini, J., & Gebru, T. (2018). Gender shades: Intersectional accuracy disparities in commercial gender classification. *Proceedings of Machine Learning Research, 81,* 77–91. *http://proceedings.mlr.press/v81/buolamwini18a.html*

Celikyilmaz, A., Deng, L., & Hakkani-Tür, D. (2018). Deep learning in spoken and text-based dialog systems. In L. Deng & Y. Liu (Eds.), *Deep learning in natural language processing* (pp. 49–78). Springer. https://doi.org/10.1007/978-981-10-5209-5_3

Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., St. John, R., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y.-H., Strope, B., & Kurzweil, R. (2018, March 29). *Universal sentence encoder.* In *Proceedings of the 2018 conference on empirical methods in natural language processing: System demonstrations* (pp. 169–174). Association for Computational Linguistics. http://dx.doi.org/10.18653/v1/D18-2029

Chidambaram, M., Yang, Y., Cer, D., Yuan, S., Sung, Y.-H., Strope, B., & Kurzweil, R. (2018). Learning cross-lingual sentence representations via a multi-task dual-encoder model. In *Proceedings of the 4th workshop on representation learning for NLP* (pp. 250–259). Association for Computational Linguistics. http://dx.doi.org/10.18653/v1/W19-4330

Chomsky, N. (1956). Three models for the description of language. *Institute of Radio Engineers, Transactions on information theory, 2*(3), 113–124.

Chu, C., Dabre, R., & Kurohashi, S. (2017). An empirical comparison of domain adaptation methods for neural machine translation. In R. Barzilay & M.-Y. Kan (Eds.), *Proceedings of the 55***th** *Annual Meeting of the Association for Computational Linguistics* (pp. 385–391). Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-2061

Clark, K., & Manning, C. D. (2016). Deep reinforcement learning for mention-ranking coreference models. In J. Su, K. Duh, & X. Carreras (Eds.), *Proceedings of the 2016 conference on empirical methods in natural language processing* (pp. 2256–2262). Association for Computational Linguistics. https://doi.org/10.18653/v1/D16-1245

Deng, L., & Liu, Y. (Eds.). (2018). *Deep learning in natural language processing*. Springer. https://link.springer.com/book/10.1007/978-981-10-5209-5

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *In Proceedings of the 2019 conference of the North American chapter of the Association for Computational Linguistics: Human language technologies* (Vol. 1; pp. 4171–4186). Association for Computational Linguistics. http://dx.doi.org/10.18653/v1/N19-1423

Ekman, P. (1992). An argument for basic emotions. *Cognition and Emotion*, *6*(3–4), 169–200. https://doi.org/10.1080/02699939208411068

Freitag, M., & Al-Onaizan, Y. (2016, December 20). *Fast domain adaptation for neural machine translation*. arXiv. https://doi.org/10.48550/arXiv.1612.06897

Garau, G., Renals, S., & Hain, T. (2005). Applying vocal tract length normalization to meeting recordings. *Interspeech, 2005,* 265–268. ISCA. https://doi.org/10.21437/Interspeech.2005-154

Gelas, H., Abate, S. T., Besacier, L., & Pellegrino, F. (2011). Quality assessment of crowdsourcing transcriptions for African languages. In *12th annual conference of the International Speech Communication Association* (pp. 3065–3068). ISCA. http://dx.doi.org/10.21437/Interspeech.2011-767

Ghosh, A., & Veale, D. T. (2016). Fracking sarcasm using neural network. In A. Balahur, E. van der Goot, P. Vossen, & A. Montoyo (Eds.), *Proceedings of the 7th workshop on computational approaches to subjectivity, sentiment and social media analysis* (pp. 161–169). Association for Computational Linguistics. https://doi.org/10.18653/v1/W16-0425

Gibbon, D. (2017). *Prosody: The rhythms and melodies of speech.* https://doi.org/10.48550/arXiv.1704.02565

Gollapalli, S. D., Li, X., & Yang, P. (2017). Incorporating expert knowledge into keyphrase extraction. *Proceedings of the AAAI Conference on Artificial Intelligence*, *31*(1). https://doi.org/10.1609/aaai.v31i1.10986

Google. (n.d.). *Google search.* https://google.com

Ho, V. A., Nguyen, D. H.-C., Nguyen, D. H., Pham, L. T.-V., Nguyen, D.-V., van Nguyen, K., & Nguyen, N. L.-T. (2019). Emotion recognition for Vietnamese social media text. *In L. M.* Nguyen, X. H. Phan, K. Hasida, and S. Tojo (Eds.), *Proceedings of the 16th International Conference of the Pacific Association for Computational Linguistics* (pp. 319–333). Spinger. https://doi.org/10.1007/978-981-15-6168-9_27

Hubens, N. (2018). *Deep inside: Autoencoders*. Medium. https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f

Hutchins, J. (1995). "The whisky was invisible," or persistent myths of MT. *MT News International*, *11*, 17–18.

Hutchins, J. (1997). From first conception to first demonstration: The nascent years of machine translation, 1947–1954: A chronology. *Machine Translation*, *12*(3), 195–252. https://doi.org/10.1023/A:1007969630568

Isewon, I., Oyelade, J., & Oladipupo, O. (2014). Design and implementation of text to speech conversion for visually impaired people. *International Journal of Applied Information Systems*, *7*(2), 25–30. https://doi.org/10.5120/IJAIS14-451143

James, G. (2013). *An introduction to statistical learning: With applications in R*. Springer. https://doi.org/10.1007/978-1-4614-7138-7

Johns, R. (2019). *Dying tongues and the curse of dimensionality: Can natural language processing force endangered and non-Western languages out of science?* Medium. https://towardsdatascience.com/dying-tongues-and-the-curse-of-dimensionality-96b42c628559

Jupyter. (n.d.). *Jupyter.* https://jupyter.org/

Jurafsky, D., & Martin, J. H. (2022). *Speech and language processing* (3rd ed.). Prentice Hall. https://web.stanford.edu/~jurafsky/slp3

Kim, Y. (2014). Convolutional neural networks for sentence classification. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (pp. 1746–1751). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1181

Kim, Y., Petrov, P., Petrushkov, P., Khadivi, S., & Ney, H. (2019, September 20). Pivot-based transfer learning for neural machine translation between non-English languages. In K. Inui, J. Jiang, V. Ng, & X. Wan (Eds.), *Proceedings of the 2019 Conference on Empirical*

Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (pp. 866–876). Association for Computational Linguistics. http://dx.doi.org/10.18653/v1/D19-1080

Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R. S., Torralba, A., Urtasun, R., & Fidler, S. (2015, June 22). *Skip-thought vectors*. arXiv. https://doi.org/10.48550/arXiv.1506.06726

Koehn, P., & Knowles, R. (2017a). Six challenges for neural machine translation. In T. Luong, A. Birch, G. Neubig, & A. Finch (Eds.), *Proceedings of the First Workshop on Neural Machine Translation* (pp. 28–39). Association for Computational Linguistics. https://doi.org/10.18653/v1/W17-3204

Koehn, P., & Knowles, R. (2017b, June 13). *Six challenges for neural machine translation*. arXiv. https://doi.org/10.48550/arXiv.1706.03872

Kulkarni, A., Shivananda, A., & Kulkarni, A. (2021). *Natural language processing projects: Build next-generation NLP applications using AI techniques* (1st edition). Apress. https://doi.org/10.1007/978-1-4842-7386-9

Ladefoged, P., & Maddieson, I. (1996). *The sounds of the world's languages*. *Phonological theory*. Blackwell.

Ladefoged, P., & Johnson, K. (2011). *A course in phonetics* (6th ed.). Wadsworth Cengage Learning.

Le, V.-B., & Besacier, L. (2009). Automatic speech recognition for under-resourced languages: Application to Vietnamese language. *IEEE Transactions on Audio, Speech, and Language Processing*, *17*(8), 1471–1482. https://doi.org/10.1109/TASL.2009.2021723

Levelt, W. J. (1999). Models of word production. *Trends in Cognitive Sciences*, *3*(6), 223–232. https://doi.org/10.1016/S1364-6613(99)01319-4

Levelt, W. J., & Wheeldon, L. (1994). Do speakers have access to a mental syllabary? *Cognition*, *50*(1-3), 239–269. https://doi.org/10.1016/0010-0277(94)90030-2

Li, J., Sun, A., Han, J., & Li, C. (2022). A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, *34*(1), 50–70. https://doi.org/10.1109/tkde.2020.2981314

Liu, Y [Yinhan], Ott, M., Goyal, N., Du Jingfei, Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019, July 26). *RoBERTa: A robustly optimized BERT pretraining approach*. In *Proceedings of the 20th Chinese National Conference on Computational Linguistics* (pp. 1218–1227). Chinese Information Processing Society of China. https://aclanthology.org/2021.ccl-1.108

Lo, S. L., Cambria, E., Chiong, R., & Cornforth, D. (2017). Multilingual sentiment analysis: From formal to informal and scarce resource languages. *Artificial Intelligence Review*, *48*(4), 499–527. https://doi.org/10.1007/s10462-016-9508-4

Loginova, E., Varanasi, S., & Neumann, G. (2021). Towards end-to-end multilingual question answering. *Information Systems Frontiers*, *23*(1), 227–241. https://doi.org/10.1007/s10796-020-09996-1

Medeiros, J. (2015). *How Intel gave Stephen Hawking a voice*. Wired. https://www.wired.com/2015/01/intel-gave-stephen-hawking-voice/

Mihalcea, R., Banea, C., & Wiebe, J. (2007). Learning multilingual subjective language via cross-lingual projections. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics* (pp. 976–983). Association for Computational Linguistics. https://aclanthology.org/P07-1123

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013, January 16). *Efficient estimation of word representations in vector space*. https://doi.org/10.48550/arXiv.1301.3781

Nasukawa, T., & Yi, J. (2003). Sentiment analysis. In J. Gennari, B. Porter, & Y. Gil (Eds.), *Proceedings of the International Conference on Knowledge Capture* (pp. 70–77). ACM Press. https://doi.org/10.1145/945645.945658

National Cancer Institute. (2003). *Larynx (Top View)* [image]. Wikimedia. https://commons.wikimedia.org/wiki/File:Larynx_(top_view).jpg

Nichol, A. (2018). *The next generation of AI assistants in enterprise*. O'Reilly. https://www.oreilly.com/radar/the-next-generation-of-ai-assistants-in-enterprise/

Ning, Y., He, S., Wu, Z., Xing, C., & Zhang, L.-J. (2019). A review of deep learning based speech synthesis. *Applied Sciences*, *9*(19), 4050. https://doi.org/10.3390/app9194050

Nooteboom, S. (1997). The prosody of speech: Melody and rhythm. In W. J. Hardcastle & J. Laver (Eds.), *The handbook of phonetic sciences* (pp. 640–673). Blackwell.

OpenStax University Physics. (2016, May 18). *CNX UPhysics 17 05 Larynx* [image]. Wikimedia. https://commons.wikimedia.org/wiki/File:CNX_UPhysics_17_05_Larynx.png

Pang, B., & Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics* (pp. 271–278). Association for Computational Linguistics. http://dx.doi.org/10.3115/1218955.1218990

Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, *2*(1–2), 1–135. https://doi.org/10.1561/1500000011

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2001). BLEU: A method for automatic evaluation of machine translation. In P. Isabelle (Ed.), *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (pp. 311–318). Association for Computational Linguistics. https://doi.org/10.3115/1073083.1073135

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (pp. 1532–1543). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162

Przybocki, M., Peterson, K., Bronsart, S., & Sanders, G. (2009). The NIST 2008 Metrics for machine translation challenge: Overview, methodology, metrics, and results. *Machine Translation*, *23*(2–3), 71–103. https://doi.org/10.1007/s10590-009-9065-6

Ramshaw, L., & Marcus, M. (1995). Text chunking using transformation-based learning. In *Third workshop on very large corpora* (pp. 82–94). Association for Computational Linguistics.https://aclanthology.org/W95-0107

Randles, B. M., Pasquetto, I. V., Golshan, M. S., & Borgman, C. L. (2017). Using the Jupyter Notebook as a tool for open science: An empirical study. In *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)* (pp. 1–2). IEEE. https://doi.org/10.1109/JCDL.2017.7991618

Ratnovsky, A., Elad, D., & Halpern, P. (2008). Mechanics of respiratory muscles. *Respiratory Physiology & Neurobiology*, *163*(1-3), 82–89. https://doi.org/10.1016/j.resp.2008.04.019

Roebuck, K. (2012). *Sentiment analysis: High-impact strategies: What you need to know: Definitions, adoptions, impact, benefits, maturity, vendors*. Emereo Pub.

Ruan, S., Wobbrock, J. O., Liou, K., Ng, A., & Landay, J. A. (2018). Comparing speech and keyboard text entry for short messages in two languages on touchscreen phones. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, *1*(4), 1–23. https://doi.org/10.1145/3161187

Russell, J. A. (1979). Affective space is bipolar. *Journal of Personality and Social Psychology*, *37*(3), 345–356. https://doi.org/10.1037/0022-3514.37.3.345

Russell, S., & Norvig, P. (2021). *Artificial intelligence: A modern approach* (4th ed., Global ed.). Pearson Education.

Saheer, L., Dines, J., & Garner, P. N. (2012). Vocal tract length normalization for statistical parametric speech synthesis. *IEEE Transactions on Audio, Speech, and Language Processing*, *20*(7), 2134–2148. https://doi.org/10.1109/TASL.2012.2198058

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv.* https://doi.org/10.48550/arXiv.1910.01108

Sap, M., Card, D., Gabriel, S., Choi, Y., & Smith, N. A. (2019). The risk of racial bias in hate speech detection. In A. Korhonen, D. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 1668–1678). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1163

Scherrer, Y., Samardžić, T., & Glaser, E. (2019). Digitising Swiss German: How to process and study a polycentric spoken language. *Language Resources and Evaluation*, *53*(4), 735–769. https://doi.org/10.1007/s10579-019-09457-5

Schlippe, T., Quaschningk, W., & Schultz, T. (2014). Combining grapheme-to-phoneme converter outputs for enhanced pronunciation generation in low-resource scenarios. In *4th Workshop on Spoken Language Technologies for Under-Resourced Languages* (pp. 139–145). ISCA. https://www.isca-speech.org/archive/sltu_2014/schlippe14_sltu.html

Schriefers, H., & Vigliocco, G. (2015). Speech Production, Psychology of. In J. D. Wright (Ed.), *International encyclopedia of the social & behavioral sciences* (pp. 255–258). Elsevier. https://doi.org/10.1016/B978-0-08-097086-8.52022-4

Schultz, T., & Waibel, A. (2001). Language-independent and language-adaptive acoustic modeling for speech recognition. *Speech Communication*, *35*(1–2), 31–51. https://doi.org/10.1016/S0167-6393(00)00094-7

Schwartz, O. (2019, November 4). *In the 17th century, Leibniz dreamed of a machine that could calculate ideas*. IEEE Spectrum. https://spectrum.ieee.org/in-the-17th-century-leibniz-dreamed-of-a-machine-that-could-calculate-ideas

Shankar, V., & Parsana, S. (2022). An overview and empirical comparison of natural language processing (NLP) models and an introduction to and empirical application of autoencoder models in marketing. *Journal of the Academy of Marketing Science, 50, 1324*–1350. https://doi.org/10.1007/s11747-022-00840-3

Sharif, W., Samsudin, N. A., Deris, M. M., & Naseem, R. (2016). Effect of negation in sentiment analysis. In E. Ariwa (Ed.), *Sixth International Conference on Innovative Computing Technology* (pp. 718–723). IEEE. https://doi.org/10.1109/INTECH.2016.7845119

Sharma, R., Somani, A., Kumar, L., & Bhattacharyya, P. (2017). Sentiment intensity ranking among adjectives using sentiment bearing word embeddings. In M. Palmer, R. Hwa, & S. Riedel (Eds.), *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (pp. 547–552). Association for Computational Linguistics. https://doi.org/10.18653/v1/D17-1058

Singh, S. (2018). *Natural language processing for information extraction*. arXiv. http://arxiv.org/pdf/1807.02383v1

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., & Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical papers* (pp. 223–231). Association for Machine Translation in the Americas. https://aclanthology.org/2006.amta-papers.25

Snyder, B., & Barzilay, R. (2007). Multiple aspect ranking using the good grief algorithm. In *Proceedings of the Joint Human Language Technology/North American Chapter of the ACL Conference (pp. 300–307).* Association for Computational Linguistics. https://aclanthology.org/N07-1038

Solorio, T., & Liu, Y. (2008). Learning to predict code-switching points. In M. Lapata & H. T. Ng (Eds.), *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 973–*981*). Association for Computational Linguistics. https://doi.org/10.3115/1613715.1613841

Soni, V. D. (2019). Speech recognition: Transcription and transformation of human speech. *International Journal on Integrated Education*, *2*(6), 257–262.

sPacy. (n.d.). *sPacy*. https://spacy.io/

Su, J., Wu, H., Wang, H., Chen, Y., Shi, X., Dong, H., & Liu, Q. (2012). Translation model adaptation for statistical machine translation with monolingual topic information. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Volume 1* (pp. 459–468). Association for Computational Linguistics. https://aclanthology.org/P12-1048

Surachit. (2003). *Ear-anatomy-notext-small* [image]. Wikimedia. https://commons.wikimedia.org/wiki/File:Ear-anatomy-notext-small.svg

Tan, X., Qin, T., Soong, F., & Liu, T.-Y. (2021). *A survey on neural speech synthesis.* arXiv. https://doi.org/10.48550/arXiv.2106.15561

Tealab, A. (2018). Time series forecasting using artificial neural networks methodologies: A systematic review. *Future Computing and Informatics Journal*, *3*(2), 334–340. https://doi.org/10.1016/j.fcij.2018.10.003

TheCPMills. (2018, November 18). *IPA English Vowels and Diphthongs with Sound Examples* [image]. Wikimedia. https://commons.wikimedia.org/wiki/File:IPA_English_Vowels_and_Diphthongs_with_Sound_Examples.svg

Tjong Kim Sang, E. F., & De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task. In W. Daelemans & M. Osborne (Eds.), *Proceedings of the 7th Conference on Natural Language Learning* (pp. 142–147). Association for Computational Linguistics. https://doi.org/10.3115/1119176.1119195

Tomokiyo, L. M., & Waibel, A. (2001). Adaptation methods for non-native speech. In *Proceedings of multilinguality in spoken language processing.* Carnegie Mellon University. https://www.ri.cmu.edu/publications/adaptation-methods-for-non-native-speech/

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, *LIX*(236), 433–460. https://doi.org/10.1093/mind/LIX.236.433

Turney, P. D. (2002). Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (pp. 417–424). https://doi.org/10.48550/arXiv.cs/0212032

Valueva, M. V., Nagornov, N. N., Lyakhov, P. A., Valuev, G. V., & Chervyakov, N. I. (2020). Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation*, *177*, 232–243. https://doi.org/10.1016/j.matcom.2020.04.031

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017, June 12). *Attention is all you need*. arXiv. http://arxiv.org/pdf/1706.03762v5

Wang, S., & Li, G. (2019). Overview of end-to-end speech recognition. *Journal of Physics: Conference Series*, *1187*(5), 52068. https://doi.org/10.1088/1742-6596/1187/5/052068

Waibel, A. (1988). *Prosody and speech* recognition. Pitman.

Wankhade, M., Rao, A. C. S., & Kulkarni, C. (2022). A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review, 55, 5731–5780*. https://doi.org/10.1007/s10462-022-10144-1

Weizenbaum, J. (1966). ELIZA: A computer program for the study of natural language communication between man and machine. *Communications of the ACM*, *9*(1), 36–45. https://doi.org/10.1145/365153.365168

Wu, H., & Wang, H. (2009). Revisiting Pivot Language Approach for Machine Translation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP* (pp. 154–162). Association for Computational Linguistics. https://aclanthology.org/P09-1018

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). *XLNet: Generalized autoregressive pretraining for language understanding.* arXiv. https://doi.org/10.48550/arXiv.1906.08237

Zampieri, M., Nakov, P., & Scherrer, Y. (2020). Natural language processing for similar languages, varieties, and dialects: A survey. *Natural Language Engineering*, *26*(6), 595–612. https://doi.org/10.1017/S1351324920000492

Zhang, D. D. (2000). Speaker Recognition. In K.-Y. Cai & D. D. Zhang (Eds.), *The international series on Asian studies in computer and information science: Automated biometrics* (Vol. 7, pp. 179–201). Springer. https://doi.org/10.1007/978-1-4615-4519-4_9

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). *Aligning books and movies: Towards story-like visual explanations by watching movies and reading books.* arXiv. https://doi.org/10.48550/arXiv.1506.06724

Zimmermann, T., Kotschenreuther, L., & Schmidt, K. (2016, June 17). *Data-driven HR: Résumé analysis based on natural language processing and machine learning*. arXiv. http://arxiv.org/pdf/1606.05611v2

LIST OF TABLES AND FIGURES

Figure 1: NLP in Relation to Computer Science and Linguistics . . . 10
Figure 2: Major Subdomains of NLP . . . 11
Table 1: Typical Conversation With ELIZA . . . 13
Figure 3: Example of Part-of-Speech Tagging . . . 15
Figure 4: Stress in Prosody . . . 18
Figure 5: Example of How Prosody Can Affect the Meaning of a Sentence . . . 19
Table 2: Example Grammar . . . 20
Figure 6: Parse Tree . . . 21
Figure 7: Chomsky’s Types of Grammars . . . 22
Figure 8: The Vocal Tract . . . 27
Figure 9: Larynx (Top View) . . . 28
Figure 10: Levels of Speech Production . . . 30
Figure 11: Example of Conceptual Preparation . . . 31
Figure 12: Subdisciplines of Phonetics . . . 32
Table 3: Examples of Differences Between IPA and ARPAbet . . . 33
Figure 13: IPA English Vowels and Diphthongs with Examples . . . 34
Figure 14: Regions of the Mouth . . . 35
Figure 15: Analog-to-Digital Conversion of a Signal . . . 37
Figure 16: Representation of a Signal in Time and Frequency Domains . . . 38
Figure 17: Anatomy of the Human Ear . . . 39

Figure 18: Different Types of Text Annotations 45
Figure 19: The Confusion Matrix 48
Figure 20: The ROC Curve 50
Figure 21: Different Shapes of the ROC Curve 51
Figure 22: Different Meanings of One Word 53
Figure 23: Dimensions of Language Variations 54
Figure 24: Example of Pivot Machine Translation 56
Figure 25: Example of the Use of Language Models 64
Figure 26: Comparison of CBOW and Skip-Gram 68
Figure 27: Simplified Encoder-Decoder Architecture 74
Figure 28: ASR and Speech Synthesis in a Voice Assistant 81
Figure 29: Connection Between Text-to-text Translation and Speech-to-Speech Translation 83
Figure 30: Example of Named Entity Recognition 84
Table 4: CoNLL Annotation Example 85
Figure 31: Coreference Resolution 86
Figure 32: Multipolarity Example 88
Figure 33: Five Levels of AI Assistants 90
Figure 34: Components of a Chatbot 91
Figure 35: Example of AIML 92
Figure 36: Creating a New Notebook 96
Figure 37: The New Notebook 96
Figure 38: Hello World Example 97
Figure 39: Markdown Example 97

Figure 40: Creating a spaCy Document 98

Figure 41: Tokenization in spaCy 99

Figure 42: Sentence Tokenization with spaCy 99

Figure 43: Part-of-Speech Tagging with spaCy 99

Figure 44: Named Entity Recognition with spaCy 100

IU Internationale Hochschule GmbH
IU International University of Applied Sciences
Juri-Gagarin-Ring 152
D-99084 Erfurt

Mailing Address
Albert-Proeller-Straße 15-19
D-86675 Buchdorf

media@iu.org
www.iu.org

Help & Contacts (FAQ)
On myCampus you can always find answers
to questions concerning your studies.