DIGITAL CULTURE
ITMO UNIVERSITY

Introduction to Automatic Text Processing

High School of Digital Culture
ITMO University
dc@itmo.ru

Contents

1 Introduction

Welcome to the introductory course in Natural Language Processing. NLP (Natural Language Processing) is a knowledge domain and an entire science that has become especially popular in the last ten years.

Think, for example, about predictions for the next word you type when browsing or messaging, or language recognition and automatic translation into another language, or smart replies with ready- made messages, or automatic spelling and grammar checkers. These convenient features have become a part of daily life for many of us. Natural language processing is what makes all of this possible.

What is "crazy" for hockey fans and for those who discuss gender equality? How has the meaning of qualitative adjectives changed from the nineteenth century to nowadays? What figures of speech and writing manner are characteristic of an author? NLP often deals with these problems of applied linguistics.

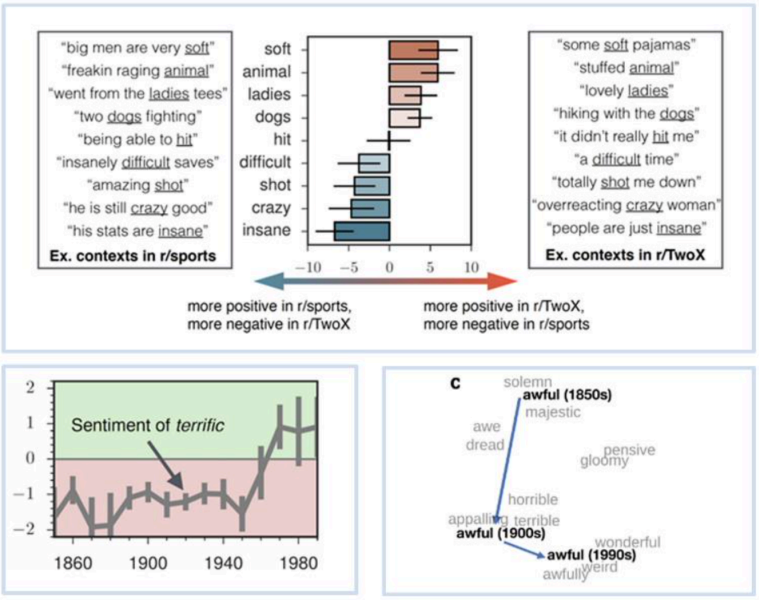

Figure 1: The word "crazy" sentiment: `https://nlp.stanford.edu/projects/socialsent/`

What are art critics writing about, and whom do they approve or disapprove of? What can we learn about side effects of a drug using patient reviews on social media? Digital texts are one of the main ways of presenting information on the Internet. That's why NLP is often used as the main tool in data-based applied interdisciplinary research.

The science of automatic language processing is well developed. Some problems are easy to solve, but other require difficult mathematics. Fortunately, available tools are becoming more advanced. Who knows, maybe it is you who

will make a breakthrough in your scientific field by using NLP for analysis of big data arrays and processing automatization. We hope this course will help you to achieve this.

1.1 Who is This Course for?

This course will cover a small but probably the most important part of the problems that can be solved with modern methods of natural language processing.

You'll find this course useful if you are going to work with texts in scientific interdisciplinary or quantitative research or if you will deal with data analysis, for example, in business.

The course is for those who are ready to study new models, approaches, and ideas. It is also available for students who do not have strong mathematical backgrounds. The basics of programming experience will also help, but they are not essential to understand and pass the course. The main prerequisites are courage, openness, and interest in machine language processing.

Every week you will attend online lectures, complete the quiz, and solve a problem (usually, using ready-made or almost ready-made software).

1.2 What is NLP, Overview of Problems

There are different definitions of NLP, but none of them is common.

[Daniel Jurafsky et al, Stanford University][1] The goal of this new field is to get computers to perform useful tasks involving human language, tasks like enabling human-machine communication, improving human-human communication, or simply doing useful processing of text or speech.

[Wikipedia][2] Natural language processing (NLP) is a subfield of linguistics, computer science, information engineering, and artificial intelligence concerned with the interactions between computers and human (natural) languages, in particular how to program computers to process and analyze large amounts of natural language data.

The main point of the definitions is that NLP is a domain at the intersection of linguistics, computer science, and artificial intelligence. Some definitions include speech or sound processing. However, despite the many intersections with NLP, speech processing is a different science that we will not discuss in this course.

1.3 NLP vs Computational Linguistics

It is important to distinguish natural language processing (NLP) from computer processing or, more precisely, computational linguistics. Obviously, they

[1] https://www.pearsonhighered.com/assets/samplechapter/0/1/3/1/0131873210.pdf

[2] https://en.wikipedia.org/wiki/Natural_language_processing

have a lot of in common. While linguistics is the scientific study of language including abstract theories and modeling, NLP focuses on applications for machine language processing, for example, human-machine interaction. It is often indicated as the goal of natural language processing. The definitions provided earlier are controversial, but we will follow them in this course.

In the era of optimism, even the greatest linguists believed that a language could be described by sets of simple laws and rules. Indeed, we were taught about rules and exceptions. Digital dictionaries of many languages have been available for a while. You will only need to work very hard to set all the rules for one large system, and machines will easily solve all the standard problems of language processing just as we do. Think about highly specialized rule-based systems that are doing great.

At the same time, the language is much more complicated than it might seem. A deep understanding of the meaning is often referred to as AI-hard problems. In terms of complexity, they can be compared to creating artificial intelligence.

1.4 Challenges of Language Processing

Let's consider some well-known examples.

Example 1. Drinking water should be moderate. What is this sentence about? Should drinking of water be moderate? Or should the amount of drinking water be moderate? We do not know who is thirsty, but we hope

Figure 2: Example 1

they'll find enough water. Both interpretations are acceptable, and only the context can give us an idea of what the author had in mind.

Example 2. Can you can a can as a canner can can a can? The first 'can' is a verb that means 'to be able to'; the second verb 'can' has the meaning 'to preserve something by sealing'. The third 'can' is a noun that stands for a 'tin used for preserving'. And finally, 'canner' is 'someone or something which cans'.

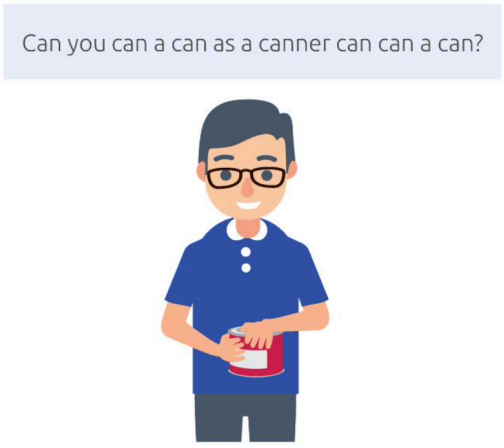

Figure 3: Example 2

The whole phrase means: 'Are you able to preserve a tin the way a person who cans food is able to preserve a tin?'

To make machines understand the meanings behind each word, we will have to do our best. Let's consider the following sentences.

Example 3. There were dresses of several different colors and styles. They were all pretty, labeled with price tags. Sally chose a blue one. Mary chose a stylish one. At the first glance, it is not evident what the problem is, as for

Figure 4: Example 3

us it's very easy. But the machine has no common sense, therefore, it's not evident whether the girls chose 'dresses' or 'price tags'? Since both dresses and price tags in general (like any other physical object) have colors, both answers would be equally right without knowing further about the real world (such as women's need and desire for pretty clothes compared to price tags). However, the context describes color and style as attributes of dresses, and not tags. If we take it into account, it should in principle increase the likelihood of 'blue' and 'stylish' being applied to dresses rather than tags. This important linguistic

task is called anaphora resolution.

A similar example, often used to demonstrate cases of parsing ambiguity, is the famous joke from the Animal Crackers movie, 1930: One morning I shot an elephant in my pajamas. How he got into my pajamas, I'll never know. If you discard the second sentence, it will be completely unclear who was in pajamas, the man or the elephant. In this case, it is important that each of the possible interpretations has its own parse tree within the framework of the immediate constituent analysis.

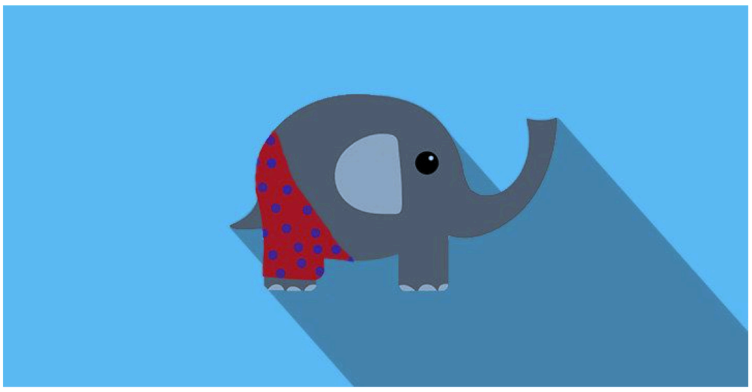

Figure 5: One morning I shot an elephant in my pajamas

Here is an incomplete list of language processing problems.

- A spelling checker is a feature of many advanced text editors, and it's quite simple.
- Hyphenation is the automated process of breaking words between lines.
- Keyword extraction is a task of finding the most important words and expressions in a text. Extraction purposes differ drastically. For example, it can be about finding complex terms or getting the idea of a text.
- Language modeling involves predicting the next word if the sequence of preceding words is known. It has a variety of applications in modern NLP.
- Part-of-speech tagging is the process of marking up a word as corresponding to a part of speech, as we all did at school. It is a task of morphological (word-category) disambiguation.
- Stemming and lemmatization bring texts to normal vocabulary form. The examples have shown that it is not a trivial task when a part of speech cannot be determined without the context.

- Named entity recognition locates named entities in a text. Sometimes it is not easy to find toponyms, names of companies or persons, and so on. It is widely used in many applications.

- Text (topic) classification is one of the fundamental language processing problems. For example, it is the task of assigning tags to unannotated texts given some tagged texts (a tag refers to positive or negative tone of voice). It includes tonality analysis, topic and sentiment classification in news, toxic comment detection, and much more.

- Syntax parsing is grammatical arrangement of words in a sentence and their relationship with each other. It is also a well-known task.

- Machine translation is the task of automatically converting source text in one language to text in another language. This field advanced a lot in recent years compared to systems distributed in the 2000s.

- Text clustering is a task of grouping similar texts in one cluster and separating dissimilar texts in other clusters without any human markup. Examples include news aggregators that surprisingly well merge news articles on events.

- Topic modeling is like clustering. It is a topic selecting from texts and matching each document with topics distribution. It is often used in the social sciences for unlabeled texts analysis as well as for exploratory search and data analysis. Thematic modeling allows us to quickly give an answer to the question, what all these texts are about, since each topic has a distribution of the most probable words associated with it.

- QA (question-answer systems) and dialogue systems are, for example, smart assistants, which are widely implemented by mobile operators, banks, and corporations to quickly answer frequent questions. It also includes popular virtual assistants built in mobile devices.

- Text summarization is used to condense long documents into short summaries.

- Plagiarism detection. Well, students are well aware of this feature, since most of papers are checked for plagiarism. This task is also referred to as NLP.

This list is far from complete. There are other important applied and esoteric tasks. Besides, some tasks only make sense in some languages. For example, dividing a text into words in some hieroglyphic systems is already a problem.

Of course, we will not consider all these tasks in this course, so let's turn to the basics. There are still many problems in natural language processing, which have no satisfactory solution as of 2020. NLP is a highly competitive field now, and there are plenty of things to do there, so welcome!

2 A Brief Overview of Computational Linguistics History

Let's start with the pre-computer era. Apart from various experiments with text encryption, the first computational approach to the language that we would like to mention is the model developed by the great Russian mathematician. Andrei Markov applied the model (later called the Markov chain) to the text of the poem Eugene Onegin by Alexander Pushkin.

2.1 Pre-computer age

A simple Markov chain is a model describing a sequence of events in which the probability of each event depends only on the state attained in the previous event (even one preceding event). Markov considered the distribution of vowels and consonants in the poem and assumed that the probability that the next letter will be a vowel or consonant depends only on the class of the current letter, i.e. whether it is a vowel or consonant. Usually, a vowel follows a consonant and vice versa. In this task, the preceding context is disregarded.

Having examined the first 200 thousand characters of "Eugene Onegin", the great mathematician manually calculated that a vowel follows a vowel in 12.8% of cases, and a vowel follows a consonant in 66.3% of cases.

How to evaluate the probability that, having stumbled on a vowel, we will see that the next two letters are consonants? Within a simple Markov chain, we just need to calculate the product of probabilities of the events "a consonant follows a vowel" and "a consonant follows a consonant". The same applies to longer sequences. Using the basic properties of probabilities, it is possible to evaluate the probability that any letter at the position N in the text is a vowel or a consonant.

The most important property of a Markov chain is that for large values of N, the probability that a letter at position N is a vowel or a consonant are almost the same and do not depend on the initial state. This property is called ergodic; and the probabilities given the infinite large N are called limiting.

Using this property, Andrey Markov calculated the probability that a randomly taken letter would be a vowel. It matched the vowel ratio in the text.

2.2 Computer Age

However, the true history of computational linguistics began much later. It is difficult to even list the main milestones in the right order so that nothing is accidentally left out or no one is offended because their merits were not mentioned. With the development of digital computing, tasks of human language processing emerged immediately, and several countries began to study different human-machine interactions using natural languages. These studies primarily started in the Soviet Union and the United States.

Throughout the history of computational linguistics, machine translation has been (and remains) one of the most important and complex tasks. Nowadays, most of successful machine translation systems use machine learning algorithms that require large memories of sentence pairs in two languages. Back in 1954, many believed that the rules only were enough. That was the time when the automatic translation of 60 sentences from Russian into English was demonstrated in the United States. The translation system used only 6 grammatical rules and was a result of the collaboration between IBM and Georgetown University. Demonstration of power, wow effect, and the illusion of success helped to attract investments to the development of machine translation. However, soon they realized that the problem couldn't be solved using rules only. 12 years of studies passed, and funding was significantly cut.

Although the Georgetown experiment was sensational, it wasn't the first try. At that time in the Soviet Union, several university research groups were also working on machine translation.

Another experiment of the era of optimism, which is worth mentioning, is Eliza, one of the first chatbots or virtual assistants, developed by Joseph Weizenbaum at the MIT Artificial Intelligence Laboratory.

Eliza simulated a psychotherapist parroting back at patients what they'd just said. Let's consider a dialogue typical of Eliza:

HUMAN: Well, my boyfriend made me come here.
ELIZA: Your boyfriend made you come here?
HUMAN: He says I'm depressed much of the time.
ELIZA: I am sorry to hear you are depressed.
HUMAN: It's true. I'm unhappy.
ELIZA: Do you think coming here will help you not to be unhappy?

As you might guess, Eliza is a rule-based system. Its answers were chosen in accordance with the template that fits the user's phrase, and it did not learn anything or did not keep the context. Despite that, many people thought that Eliza was emotionally involved in the conversation and showed genuine interest. That is why anthropomorphism of computer technology, when the user knows

about interacting with a computer, is called the Eliza effect. Things have changed since then. Modern sophisticated chatbots use neural networks, but more about that in a moment.

Unfortunately, we don't have enough time to discuss all the important milestones, but here's a brief history.

2.3 1950s

The 50s of the twentieth century were marked by the first attempts to solve the problems that are now considered computational linguistics tasks. In 1948, Claude Shannon published an article entitled "A Mathematical Theory of Communication," where the basics of information theory were described. In 1957, the publication "Syntactic Structures" by Noam Chomsky was published, which revolutionized linguistics.

2.4 1960-1970s

The 60s and 70s can be called the era of the boom-and-bust cycle of symbolic artificial intelligence, when researchers believed that intelligent machines can be created using inference algorithms and expert systems. In the 60s, soviet linguists began to develop the Meaning-Text Theory first put forward by Igor Melchuk. This theory had a significant influence on Russian linguistics, and its validity has been passionately disputed for many years. In the 60s, the English language corpus of Brown University was developed (the Brown Corpus). It was one of the most important linguistic resources of many subsequent years. Its creation marked the beginning of corpus linguistics. At that time, one of the greatest scientists of the twentieth century Andrei Kolmogorov used mathematical methods in Russian verse research.

2.5 1980-1990s

In the 80s and 90s, machine learning broke into natural language processing. Numeric evaluation of the quality of text analysis based on datasets became a standard. In the Soviet Union in 1985, academician Andrei Ershov initiated the creation of the Russian Language Corpora. The most important components of text processing neural networks were invented at that time. Structured machine learning models were used for speech recognition, for example. Quantitative approaches to distributional semantics were developed.

2.6 2000s

In the 2000s the Internet era began, and every year more and more text data became available on the network. The access to the Russian National Corpus was opened. Machine learning was widely applied in natural language processing. Data storage and computing power became cheaper. Text process-

ing tasks required larger and larger datasets, some of which changed from time to time. The interest in unsupervised machine learning was growing because it revealed hidden patterns in unlabelled data with no expert opinion. Typical problems included grouping of similar texts, finding topics in a set of texts, etc.

2.7 2010s

The 2010s was the era of the dominance of one of the most important machine learning models called neural networks. The era of deep learning began. Due to the relative cheapness of data storages and computing, as well as the development of new learning methods and tools for researchers, the use of neural networks has become an effective technique and broke all records in prediction quality in computer vision and natural language processing. However, there are many nuances that one should consider when applying neural networks, such as prediction speed and performance. In practice, neural networks are not the first choice that guarantees instant success. Since this course is an introduction, we will not discuss neural networks, but we highly recommend the materials of the Stanford University course called Natural Language Processing with Deep Learning.

3 Text Preparation for Automatic Processing

Depending on the chosen task and method, there are different ways to prepare texts for automatic processing. For example, to analyze news topics, it is enough to make a set of words using each article. On the contrary, the word order is important to correctly mark up parts of speech or to parse the text. Therefore, we consider a text as a chain of words, etc.

Let's start with simple and the most common tasks of natural language preprocessing.

So, we are dealing with texts, for example, in Russian. And let's say our first task is to prepare the texts so that we can find all the documents that contain some word.

For a machine, a text is just a string, a finite sequence of characters, such as letters, spaces, punctuation, and so on. For effective word search, the string should be split into substrings, i.e. separate words and punctuation marks. This process of chopping a string into tokens (small pieces) is called tokenization. Usually, there is no need to code it yourself, as many tools can do this for you. They are often based on heuristic rules of the following form "a token is a subsequence of letters, separated from the rest of the text by spaces and commas or periods".

Note that if you're using a linguistic tool that is fed with ready-made tokens

as an input, it is important to use the tokenizer supported by your tool. For example, some tokenizers omit punctuation, and some allocate it to a special token. An algorithm that does or does not process punctuation may lose in quality if the texts are tokenized incorrectly. The same applies not only to tokenization but also to other stages of text preprocessing. Let's go back to them.

Not always, but often the meaning of individual words is more important to us than their number, tense, and so on. For example, if we want to find all the documents containing the word "mole", we will obviously need those with the word "moles". If we want to locate all the documents with the word "close", then we also need all the documents with the words "closed", "closely", "closing", and so on. To achieve this, the original words are processed in such a way that all various word forms are transformed into a single token. There are two ways of doing it.

3.1 Stemming

The first method is called stemming. There are different interpretations of stemming, but usually stemming is a process of finding the main part of a word (called a stem) that stays the same when endings are added to it. In the example of "mole", such an ideal fragment common to all forms is "mole". Note that a stem is not necessary a word. A hypothetical stemmer will select 'tradit' as a stem from the words 'traditional', 'tradition', 'traditionally'.

Besides, some letters in stemming are not truncated but replaced to avoid cases when words with completely different roots and meanings have the same stem.

There are many stemming algorithms. Let's have a look at a classic version.

Porter Stemmer The first truly successful stemmer was written by Martin Porter. It is still used sometimes and often called Porter Stemmer. This algorithm applies several processing rules step by step. The algorithm is described in detail on the author's website. We will not dig into the details but describe the original algorithm very briefly.

Step 1a.
SSES $\rightarrow$ SS
IES $\rightarrow$ I
SS $\rightarrow$ SS
S $\rightarrow$

These rules transform postfixes (i.e. endings) of words. For example, at this step we get rid of the plural forms of nouns and the third-person singular forms of verbs. If the word ends with sses, then this postfix will be replaced by ss. For example, caresses $\rightarrow$ caress.

In this example, the word remains a word. However, the second rule will clearly turn many words into something different from words. For example, ties → ti, ponies → poni written with i at the end and so on.

Step 1b.

(m>0) EED → EE

(*v*) ED →

(*v*) ING →

M is greater than zero is an estimate of instances of a vowel followed by a consonant before the suffix of the EED string. Thus, the words feed and breed are left as is, but disagreed is going to be truncated.

The rule *v* (V in asterisks) requires at least some vowels to be in the stem. So, bed cannot be truncated, but we truncate the ending of carved. The same works for gerunds ending with -ing. The idea is clear. We also get rid of the endings if it doesn't change the meaning of the word.

If the second and third rules are applied at this step, we reconstruct the initial form of the verb by truncating (if necessary) double consonants at the end and adding the letter E if necessary.

AT → ATE

BL → BLE

IZ → IZE

(*d and not (*L or *S or *Z)) → single letter (m=1 and *o) → E

Step 1c changes -y to -i at the end of the words when appropriate.

(*v*) Y → I

Well, that's the first and the most difficult step of the Porter Stemmer, which deals with tenses and plural forms. Now the idea is clear. The steps from two to five are much simpler. To see the description of the whole algorithm, visit the author's website[3]

According to Google Scholar, the article "An algorithm for suffix stripping" by Martin Porter has been cited more than ten and a half thousand times by 2019. You can learn more about the Porter stemmer in practice using the NLTK library, which we will use in this course. The author of the stemmer has developed a special framework for defining stemmer transforms. It is called the Snowball.

3.2 Lemmatization

The second and preferred way of truncating different forms of a word to bring them to a single entity is lemmatization, which derives the base form (lemma) of words. For example, the lemmatization of the word "growled" gives us "growl", and the words "bites" and "walking" become "bite" and "walk"

[3] http://snowball.tartarus.org/algorithms/porter/stemmer.html

respectively. Sometimes lemmatization is also considered one of the stemming options.

3.3 Morphological Transformation

The most common lemmatization tools usually perform dictionary-based processing by searching for the given words in some list. However, there are no complete dictionaries (nor can there be one) because languages develop and change over time. Therefore, there should be some rules to lemmatize an unknown word when only the word itself and the text surrounding it are available. Humans can do this. Recall, for example, the poem Jabberwocky by

LEWIS CARROLL: Beware the Jabberwock, my son!
The jaws that bite, the claws that catch!
Beware the Jubjub bird, and shun
The frumious Bandersnatch!

Henry A. Gleason has also invented one of the famous examples "The iggle squiggs trazed wombly in the harlish hoop".

In such phrases, the word endings are very helpful. We can determine the part of speech, gender, number, and time. Not all languages allow us to construct a similar phrase. Anyway, this course will not cover the morphological classification of languages. Let's just note that, in Russian, endings can give us a lot of information about what is going on even if the meaning of the words is unknown.

For ease of comparison, let's consider an example from literature. Vladimir Sorokin used an unusual technique in his novel The Norm. The author has redacted some words by replacing them with non-existent equivalents.

THE ASSISTANT EDITOR SAID: Well, it is a too difficult plovkrnrae.
ALEXANDER LAUGHED: Oh boy, but trafksd poktl, saretknlorfg por!

The word plovkrnrae in the first sentence is most likely a noun, probably, something like a question. However, it is impossible to make this assumption without the context (without the surrounding words and their position in the sentence). The second sentence gives no information on the bon mot that Alexander made. Moreover, it's not clear how to analyze the sentence, and there is too little contextual information to determine even parts of speech used in this sentence.

3.4 Syntactic Ambiguity

You already know that most algorithms use context to determine a dictionary form of the word. But why is it necessary? It's especially important for such highly inflectional languages as Russian.

Let's try to determine the dictionary form of the word 'flies' from the phrase 'time flies like an arrow'. It's clearly a trick. Without context, we cannot determine what part of speech it is, for example, a verb or plural noun, the lemma of both is "fly". However, when this word appears in the sentence "the time is flying like an arrow", all the doubts disappear. The same problem arises with the interpretation of 'like'. Depending on the context, it can be a verb that means 'to enjoy' or a conjunction with the meaning 'in the same way as'?

This phenomenon is called syntactic ambiguity. Therefore, even though spelling is helpful, we still need context to label parts of speech. To obtain the dictionary form of a token, it is better to use tools that can handle syntactic ambiguity. However, it doesn't matter when the incorrect lemma does not affect the result, or when the processing speed is more important.

3.5 Tools for English Text Normalization

In practice, text normalization is performed with different tools. Let's consider how to use them for morphological analysis.

Natural Language Toolkit (NLTK)[4] is a Python library providing easy-to-use interfaces to language resources, such as corpora, lexical resources, pre-trained or trainable models for NLP, etc. Although sometimes this library doesn't catch up with the industry-scale performance, it still offers a large variety of models and methods to try and choose from. NLTK can handle many language-related problems, such as splitting the text into sentences, syntactic parsing, and language modeling.

Porter Stemmer Let's look first at the text stemming performed using the Porter stemmer. It's not the only stemmer available within NLTK.

```
import nltk

stemmer = nltk.PorterStemmer()
sample_text = "the traditional dress of the mongols".split()
words = [stemmer.stem(word) for word in sample_text]
print(words)
```

```
['the', 'tradit', 'dress', 'of', 'the', 'mongol']
```

As you can see, the example is processed as expected: traditional has turned into tradit. Try it yourself by applying it to a phrase. Will the stemmer recognize irregular verbs? How will it handle the plural forms of Latin words, for example, corpora?

[4] https://www.nltk.org/

To test it in your browser, you can use a free cloud service, for example, Google Colab[5].

Since English is the most widely spoken language, it has many tools for various processing tasks; and lemmatization is no exception. We will consider only two practical examples. However, please bear in mind that there are many more out there.

As we have already touched upon NLTK, let's see what it has to offer for lemmatization.

WordNet Lemmatizer. NLTK provides WordNet Lemmatizer that uses the WordNet[6] Database to lookup lemmas of words. NLTK is often used as an interface to WordNet, which is an important, widely used, and popular language resource worth of a separate discussion. Let's look at the rocks, papers, scissors example:

```
from nltk.stem import WordNetLemmatizer

nltk.download("wordnet")
lemmatizer = WordNetLemmatizer()
print(lemmatizer.lemmatize("rocks"))
print(lemmatizer.lemmatize("papers"))
print(lemmatizer.lemmatize("scissors"))

rock
paper
scissors
```

The words rocks and papers lost their endings, while scissors, as expected, preserved the last letter. It is something a stemmer doesn't do.

Here's another task for the lemmatizer, which requires dictionary-related capabilities.

```
print(lemmatizer.lemmatize("men"))
print(lemmatizer.lemmatize("women"))
print(lemmatizer.lemmatize("corpora"))

men
woman
corpus
```

[5] https://colab.research.google.com/

[6] https://wordnet.princeton.edu/

As you can see, tools for such unformalized domains as a natural language are imperfect. The words corpora and women were processed correctly, unlike the word men.

Well, what if we do the same with the so-to-say difficult word 'better'? What will we get?

```
print(lemmatizer.lemmatize("better"))
```

```
better
```

Is this wrong? Well, it depends. If better is an adjective, the output will be good. However, if better is a thing that is better than something else, the lemmatizer is right.

WordNet is annotated with parts of speech, so the lemmatizer can handle ambiguous cases:

```
print(lemmatizer.lemmatize("better", pos="a"))
```

```
good
```

The `pos` parameter means 'part-of-speech', as you probably guessed. We set it to `a` that means adjective. This time, the algorithm looked up for the right word.

spaCy[7] is a suite of natural language processing tools. SpaCy is claimed to help users to do real work — to build real products, or gather real insights. It is a time-efficient solution and a popular NLP Python library. Apart from part-of-speech tagging, syntax parsing, and other important tasks, SpaCy provides several convenient ways to prepare texts for further processing with other instruments, for example, with some sophisticated neural networks implemented using your favorite framework, such as TensorFlow or PyTorch. Let's lemma-

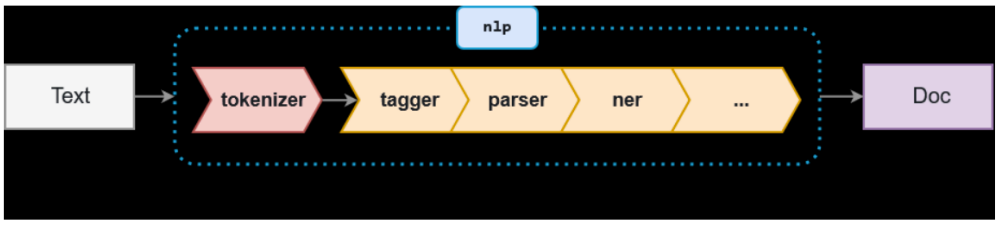

Figure 6: spaCy tokenizer

tize a short text. spaCy can process a text in a processing pipeline. When you call nlp on a text, it is processed in several different steps and yields an object

[7] `https://spacy.io/`

with the processing results. What are these steps? A typical pipeline starts with a tokenizer followed by a tagger, syntax parser, text classifier, and so on.

By the way, spaCy can process texts in small portions, or batches, which helps speed everything up. To find out more about pipelines, please refer to the user-friendly spaCy documentation.

Here's one more thing to consider. Applying all the available pipeline components to a text takes time, and it is very inefficient at scale. For example, lemmatizing doesn't require named entity recognition or text classification. That's why you should only apply the pipeline components you need. Use the disable keyword argument to get rid of unnecessary components.

First, we install the English models and print their pipeline components.

```
import spacy
nlp = spacy.load("en_core_web_sm")
print(nlp.pipe_names)
```

```
['tagger', 'parser', 'ner']
```

To extract lemmas, we only need a tagger and nothing else from the list. Let's print the original tokens, lemma and part of speech, having applied spaCy to the familiar example phrase:

```
doc = nlp("Time flies like an arrow!", disable=["ner", "parser"])
for token in doc:
    print(token.text, token.lemma_, token.pos_)
```

```
 Time Time PROPN
 flies fly VERB
 like like SCONJ
 an an DET
 arrow arrow NOUN
 ! ! PUNCT
```

Well, the results look good. A human would do the same. But what if we add the word really before like?

```
doc = nlp("Time flies really like an arrow!", disable=["ner", "parser"])
for token in doc:
    print(token.text, token.lemma_, token.pos_)
```

```
 Time Time PROPN
 flies fly VERB
 really really ADV
```

```
like like VERB
an an DET
arrow arrow NOUN
! ! PUNCT
```

Well, the word really had no disambiguating effect. As earlier, it is unclear whether certain magical insects love sharp weapons of war or whether the phrase describes the way the time moves. A broad range of tools can preprocess texts

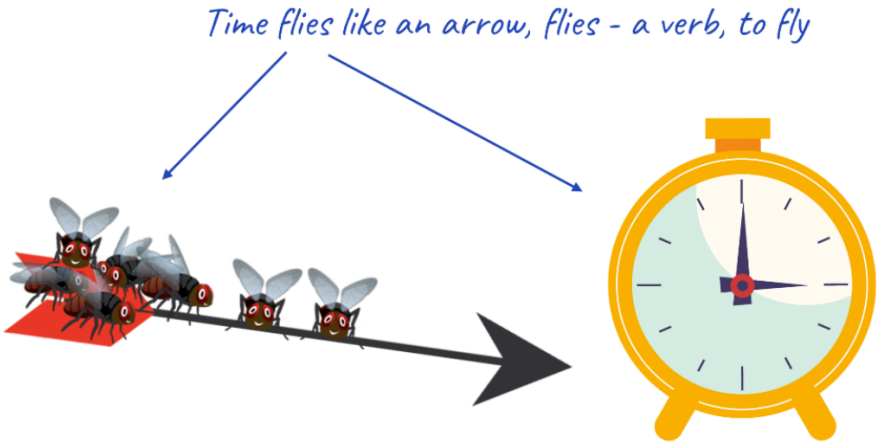

Figure 7: Time flies...

in English, and many of them have online tutorials. So, feel free to google if you are dealing with text normalization or morphological analysis.

The choice of a tool for morphological analysis and word normalization depends on the requirements. A researcher should get the priorities right and decide what is most important— quality, performance, or something else.