Fakultet organizacionih nauka
Beograd

Obrada prirodnih jezika (**N**atural **L**anguage **P**rocessing - **NLP**)

Dr Marija Đokić Petrović

O sebi …☺

Doktorat - računarske nauke

- Semantički veb
- Rudarenje tekstulanih podataka (engl. Text Data Mining)
- Obrada prirodnih jezika

Radna mesta i pozicije

- Virtual Vehicle Research GmbH, Tehnički univerzitet (TU Graz) - naučni radnik
- Univerzitet primenjenih nauka Joanneum - predavač

Oblasti interesovanja

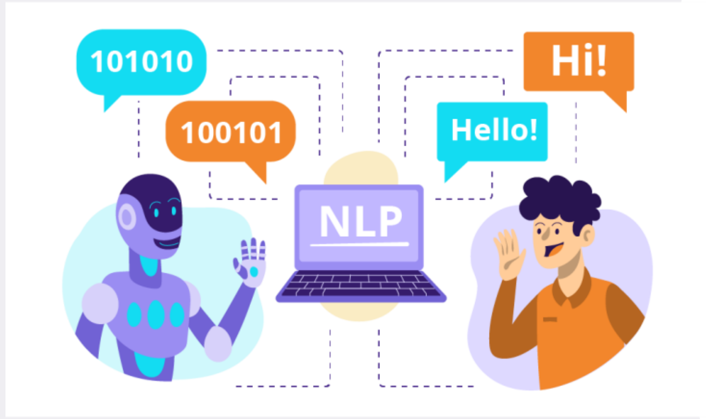
- Otkrivanje znanja (engl. Knowledge Discovery)
- Predstavljanje znanja (engl. Knowledge Representation)
- Zaključivanje
- Mašinsko učenje
- Obrada prirodnih jezika

Inženjer podataka

Aktivnosti - Savetodavno veće za strance u Gracu i srpska zajednica u Austriji

Agenda

NLP



Source: link

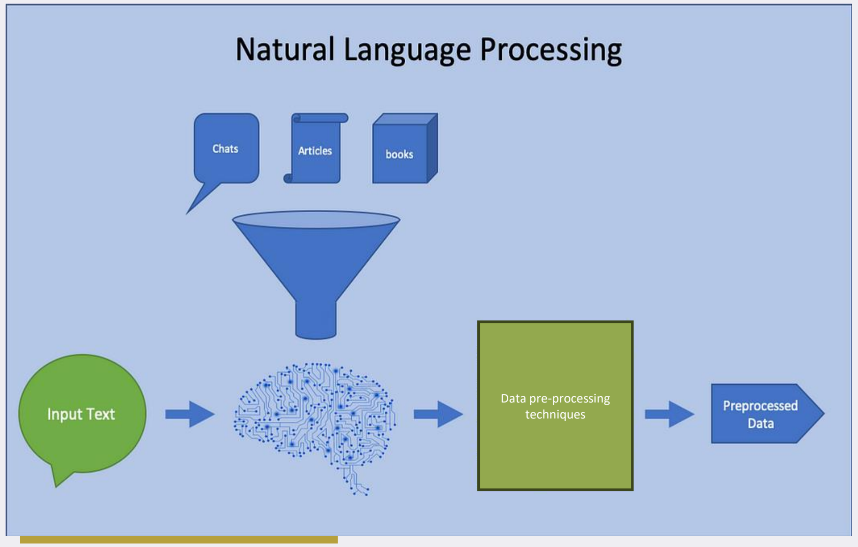
NLP je grana veštačke inteligencije koja ima za cilj da omogući mašinama da čitaju, razumeju, tumače i generišu ljudski jezik.

Razvoj tehnika i algoritama koji omogućavaju mašinama da shvate i odgovore na tekst/govor na način koji je smislen i kontekstualno relevantan.

Cilj NLP-a: premostiti jaz između ljudske komunikacije i mašinskog razumevanja, omogućavajući mašinama da komuniciraju sa ljudima na prirodniji i intuitivniji način.

Kako NLP funkcioniše?

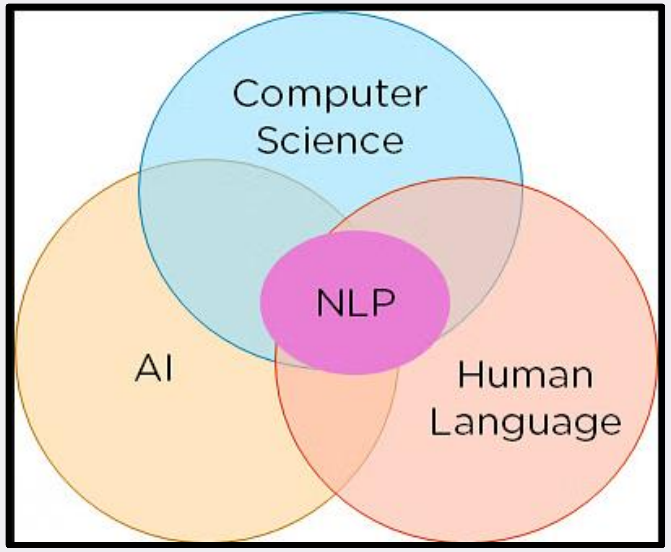
Source: link



Pretprocesiranje podataka

* Priprema, čišćenje i transforamcija sirovih tekstualnih podataka.

Source: link



Razvoj algoritama

* Korišćenje algoritama mašinskog učenja.

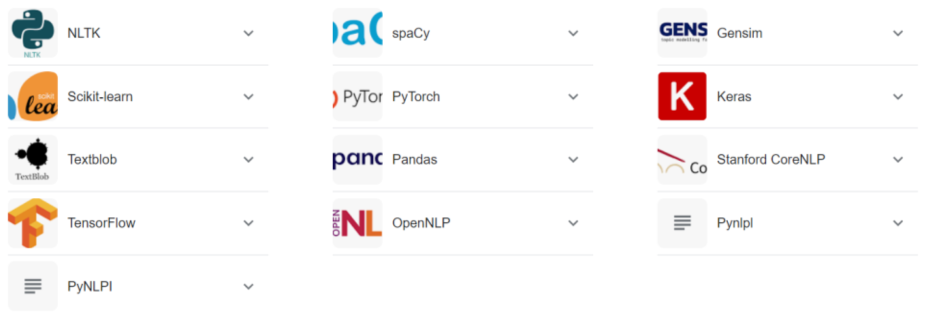
Tehnike NLP-a za pretprocesiranje podataka

Segmentacija (Segmentation)	Podela dokumenata na smislene segmente (**rečenice**)
Čišenje teksta (Text cleaning)	Izvođenje osnovnih radnji za čišćenje teksta
Tokenizacija (Tokenization)	Podela rečenica na pojedinačne reči (**tokene**)
Uklanjanje stop reči (Stop words removal)	Uklanjanje stop reči koje ne daju tekstu puno informacija
Lematizacija (Lemmatization)	Pronalaženje korena (**lemma**) analizom morfologije tokena, korišćenjem reči iz rečnika.
Steming (Stemming)	Konvertovanje tokena u njihove osnovne forme (**trims**) korišćenjem heurističkih pravila
Part-of-speech tagging	Označavanje reči odgovarajućim gramatičkim kategoriajam (imenica, glagol, pridev)

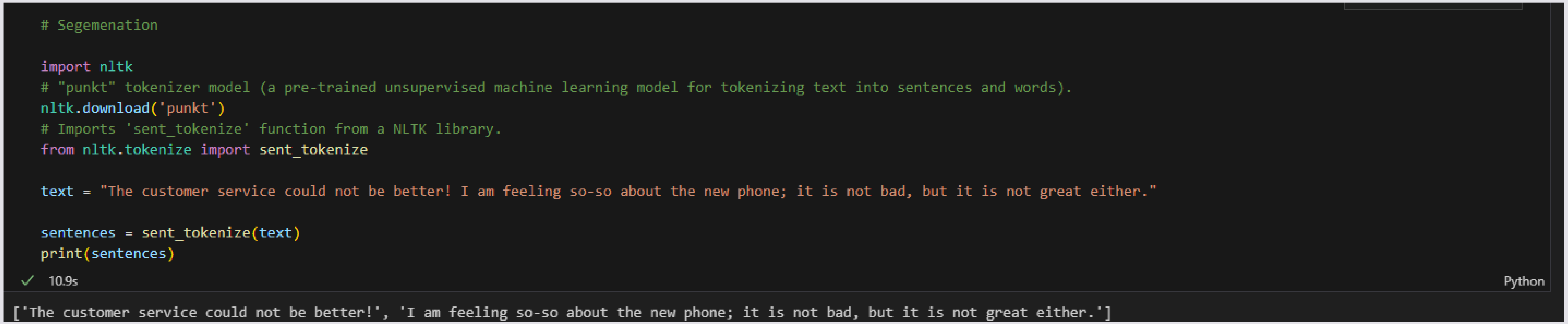
PIPELINE

NLP implementacija u Python-u

- ✓ **Natural Language Toolkit (NLTK)**: Kompletan set alata za NLP tehnike.
- ✓ **Scikit-learn**
- ✓ Pattern
- ✓ TextBlob
- ✓ **spaCy**
- ✓ **Gensim**



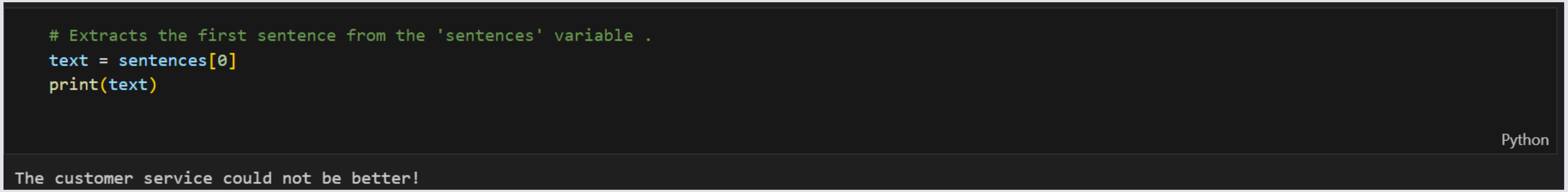
NLTK	spaCy	Gensim
Scikit-learn	PyTorch	Keras
Textblob	Pandas	Stanford CoreNLP
TensorFlow	OpenNLP	Pynlpl
PyNLPl		



```
# Segemenation

import nltk
# "punkt" tokenizer model (a pre-trained unsupervised machine learning model for tokenizing text into sentences and words).
nltk.download('punkt')
# Imports 'sent_tokenize' function from a NLTK library.
from nltk.tokenize import sent_tokenize

text = "The customer service could not be better! I am feeling so-so about the new phone; it is not bad, but it is not great either."

sentences = sent_tokenize(text)
print(sentences)
```

✓ 10.9s Python

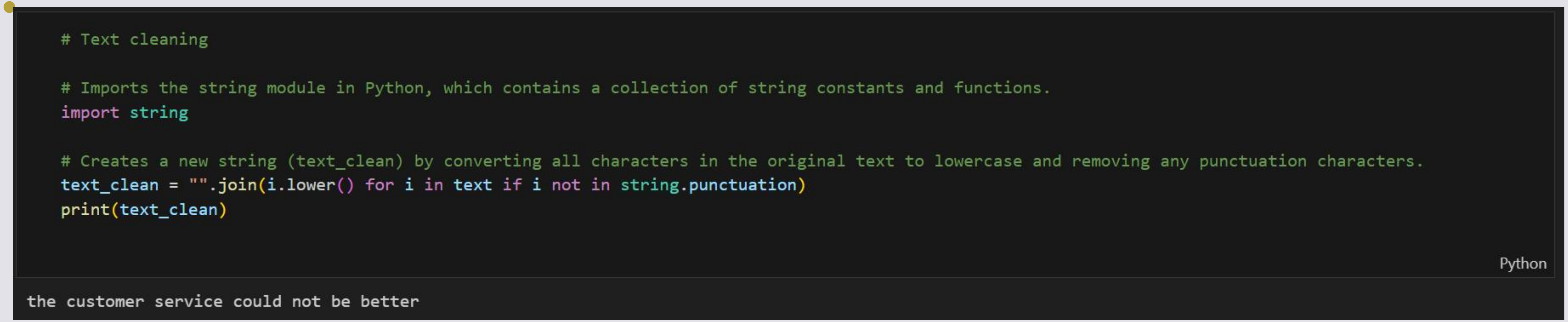
```
['The customer service could not be better!', 'I am feeling so-so about the new phone; it is not bad, but it is not great either.']
```



```
# Extracts the first sentence from the 'sentences' variable .
text = sentences[0]
print(text)
```

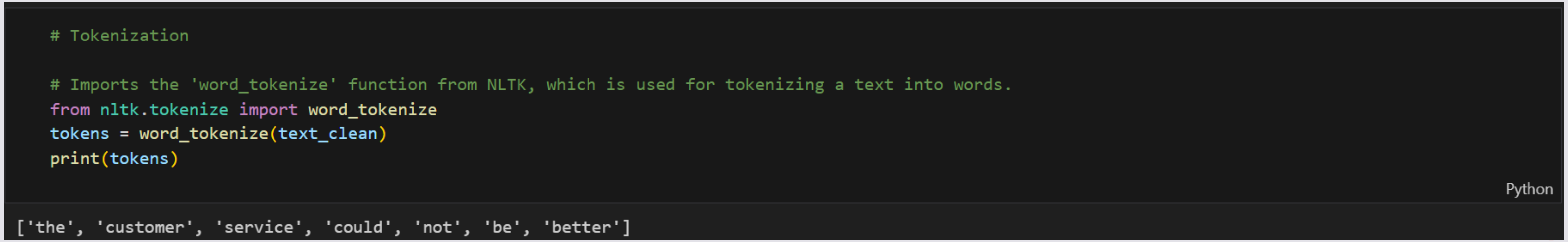
Python

```
The customer service could not be better!
```



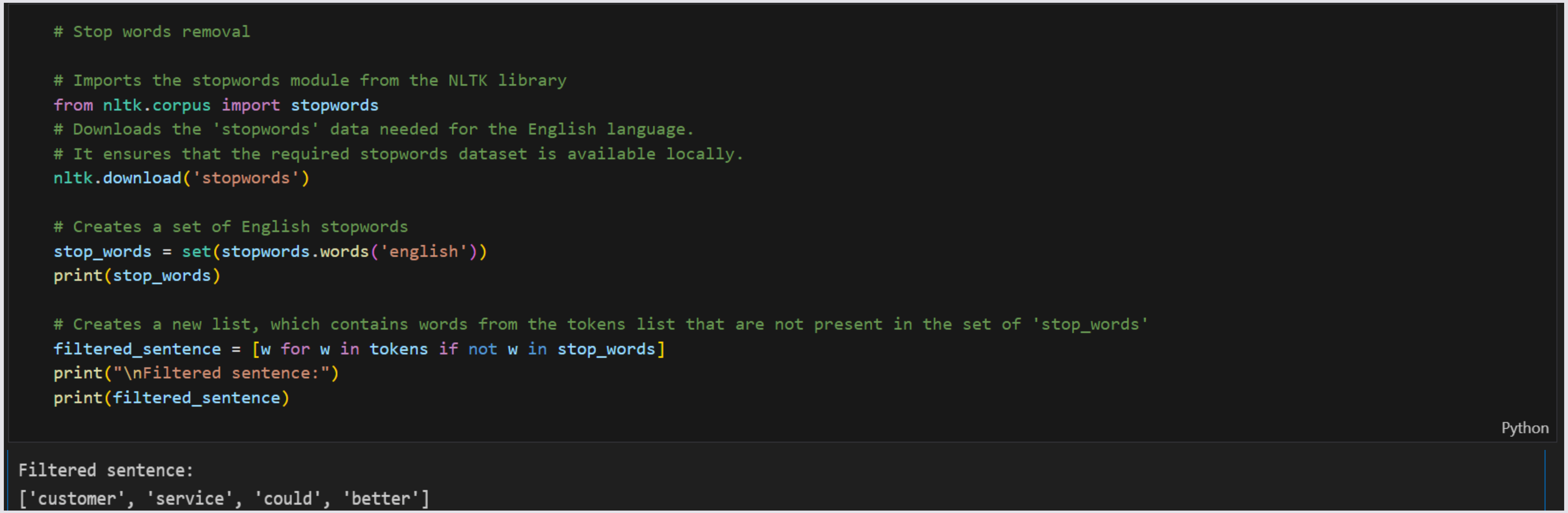
```python
# Text cleaning

# Imports the string module in Python, which contains a collection of string constants and functions.
import string

# Creates a new string (text_clean) by converting all characters in the original text to lowercase and removing any punctuation characters.
text_clean = "".join(i.lower() for i in text if i not in string.punctuation)
print(text_clean)
```

```
the customer service could not be better
```



```python
# Tokenization

# Imports the 'word_tokenize' function from NLTK, which is used for tokenizing a text into words.
from nltk.tokenize import word_tokenize
tokens = word_tokenize(text_clean)
print(tokens)
```

```
['the', 'customer', 'service', 'could', 'not', 'be', 'better']
```

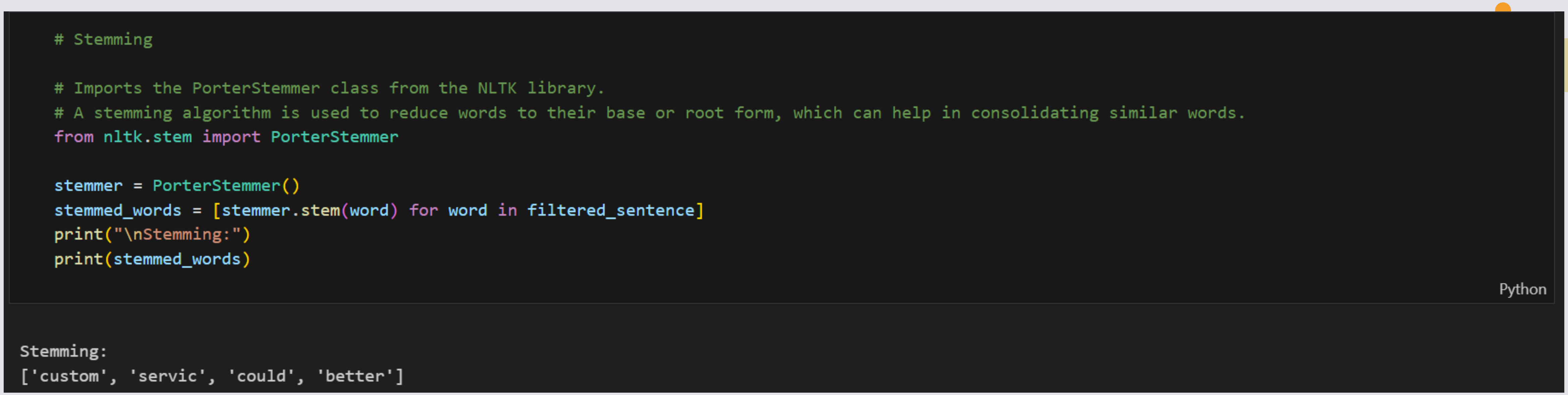


```python
# Stop words removal

# Imports the stopwords module from the NLTK library
from nltk.corpus import stopwords
# Downloads the 'stopwords' data needed for the English language.
# It ensures that the required stopwords dataset is available locally.
nltk.download('stopwords')

# Creates a set of English stopwords
stop_words = set(stopwords.words('english'))
print(stop_words)

# Creates a new list, which contains words from the tokens list that are not present in the set of 'stop_words'
filtered_sentence = [w for w in tokens if not w in stop_words]
print("\nFiltered sentence:")
print(filtered_sentence)
```

Python

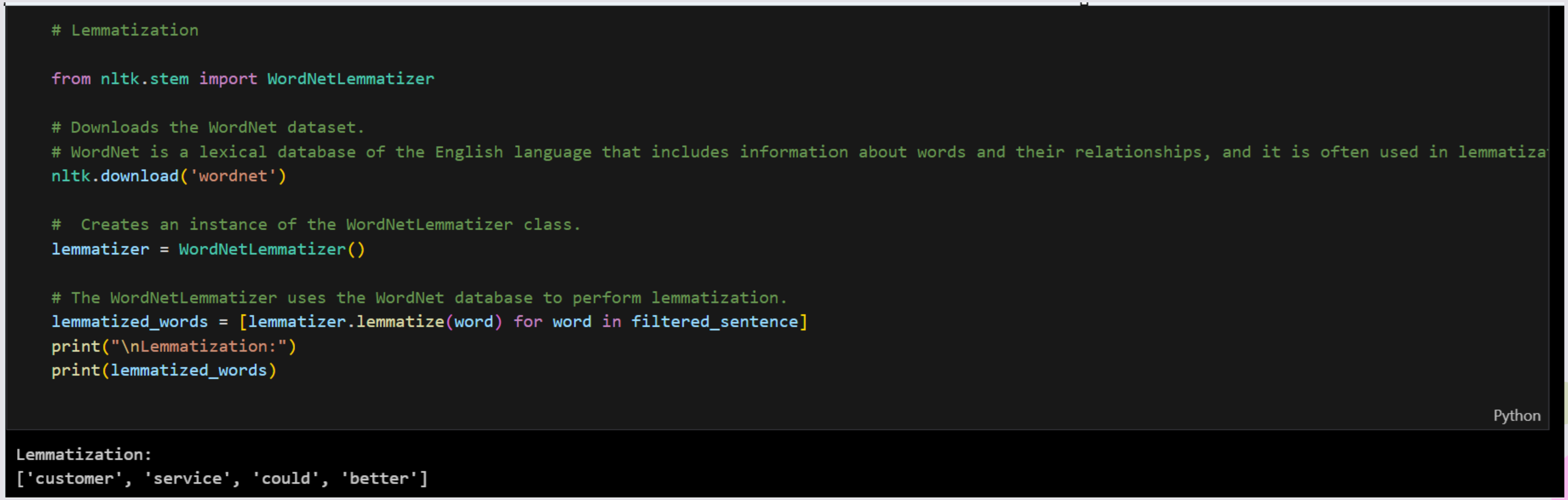
```
Filtered sentence:
['customer', 'service', 'could', 'better']
```

{'too', 'we', 'such', 'a', "wasn't", 'after', 'their', 'you', 'did', 'until', 'shouldn', 'ours', 'hasn', 'same', 'further', 'out', 'what', 'between', 'not', 'under', 'now', 'they', 'mustn', "shouldn't", 'all', 'were', 'why', 're', 'doing', 'about', "hasn't", 'ain', "isn't", 'some', 'm', "you're", 'it', 'both', 'each', "weren't", 'than', 'then', 'your', 'is', 'y', 'most', 'can', "should've", "hadn't", 'having', 'wouldn', 'had', 'at', 'here', 'any', "shan't", 'won', 'd', 'other', 'll', 'wasn', 'so', 'very', "aren't", "won't", 'itself', 'he', 'myself', 'has', 'again', 'up', 'whom', "mightn't", 'with', 'our', 'his', 'couldn', 'that', 'how', "couldn't", 'be', "haven't", 'against', "wouldn't", 'but', 'own', 's', "she's", "didn't", 'into', 'yours', 'nor', 'been', 'yourself', 'theirs', 'through', 'an', 'off', "you'll", 'being', "it's", "you've", 'does', 'shan', 'which', 'who', 'as', 've', 'don', 'mightn', 'these', "don't", 'there', 'just', 'am', 'more', 'i', 'by', "doesn't", 'and', 'she', 'for', 'from', 'my', 'yourselves', 't', 'hers', 'should', 'while', 'once', 'above', 'him', 'during', 'weren', 'those', 'themselves', 'o', 'didn', 'needn', 'only', 'was', 'haven', 'me', 'isn', "you'd", "needn't", 'ma', 'to', 'no', 'this', 'where', 'do', 'if', 'have', 'of', 'on', 'are', 'few', 'down', 'in', "that'll", 'will', 'doesn', 'her', 'or', 'ourselves', 'when', 'herself', 'its', 'the', 'below', 'over', 'himself', 'before', 'them', 'hadn', "mustn't", 'because', 'aren'}



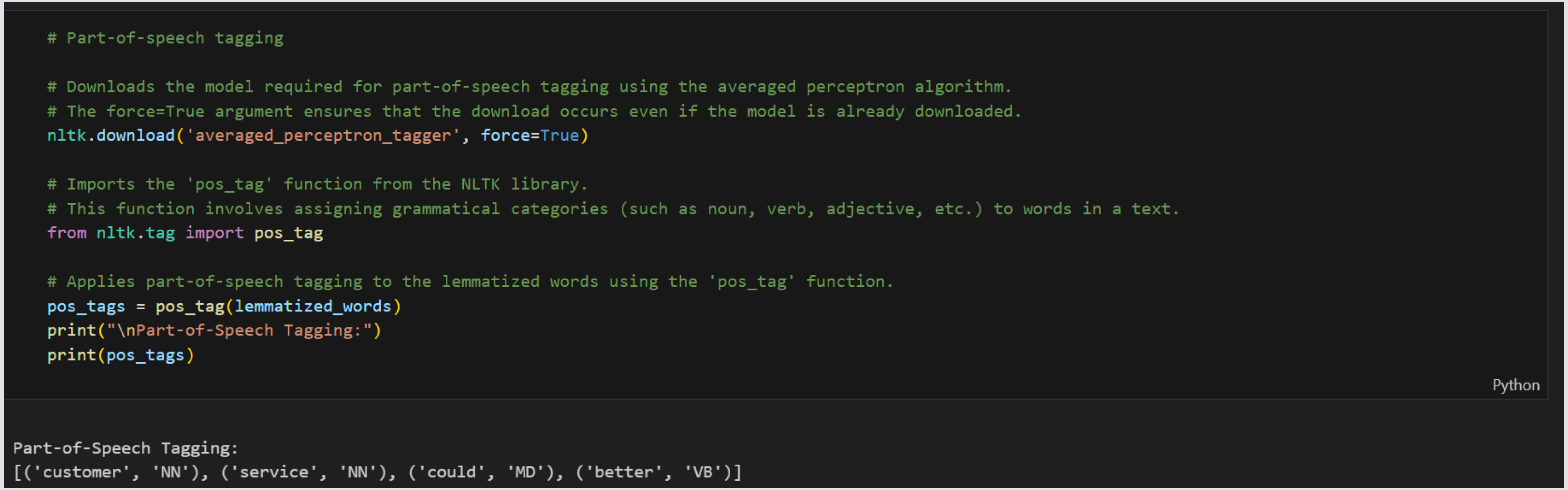
```python
# Stemming

# Imports the PorterStemmer class from the NLTK library.
# A stemming algorithm is used to reduce words to their base or root form, which can help in consolidating similar words.
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()
stemmed_words = [stemmer.stem(word) for word in filtered_sentence]
print("\nStemming:")
print(stemmed_words)
```

```
Stemming:
['custom', 'servic', 'could', 'better']
```



```python
# Lemmatization

from nltk.stem import WordNetLemmatizer

# Downloads the WordNet dataset.
# WordNet is a lexical database of the English language that includes information about words and their relationships, and it is often used in lemmatiza
nltk.download('wordnet')

#  Creates an instance of the WordNetLemmatizer class.
lemmatizer = WordNetLemmatizer()

# The WordNetLemmatizer uses the WordNet database to perform lemmatization.
lemmatized_words = [lemmatizer.lemmatize(word) for word in filtered_sentence]
print("\nLemmatization:")
print(lemmatized_words)
```

```
Lemmatization:
['customer', 'service', 'could', 'better']
```



```python
# Part-of-speech tagging

# Downloads the model required for part-of-speech tagging using the averaged perceptron algorithm.
# The force=True argument ensures that the download occurs even if the model is already downloaded.
nltk.download('averaged_perceptron_tagger', force=True)

# Imports the 'pos_tag' function from the NLTK library.
# This function involves assigning grammatical categories (such as noun, verb, adjective, etc.) to words in a text.
from nltk.tag import pos_tag

# Applies part-of-speech tagging to the lemmatized words using the 'pos_tag' function.
pos_tags = pos_tag(lemmatized_words)
print("\nPart-of-Speech Tagging:")
print(pos_tags)
```

Python

```
Part-of-Speech Tagging:
[('customer', 'NN'), ('service', 'NN'), ('could', 'MD'), ('better', 'VB')]
```

Korpus

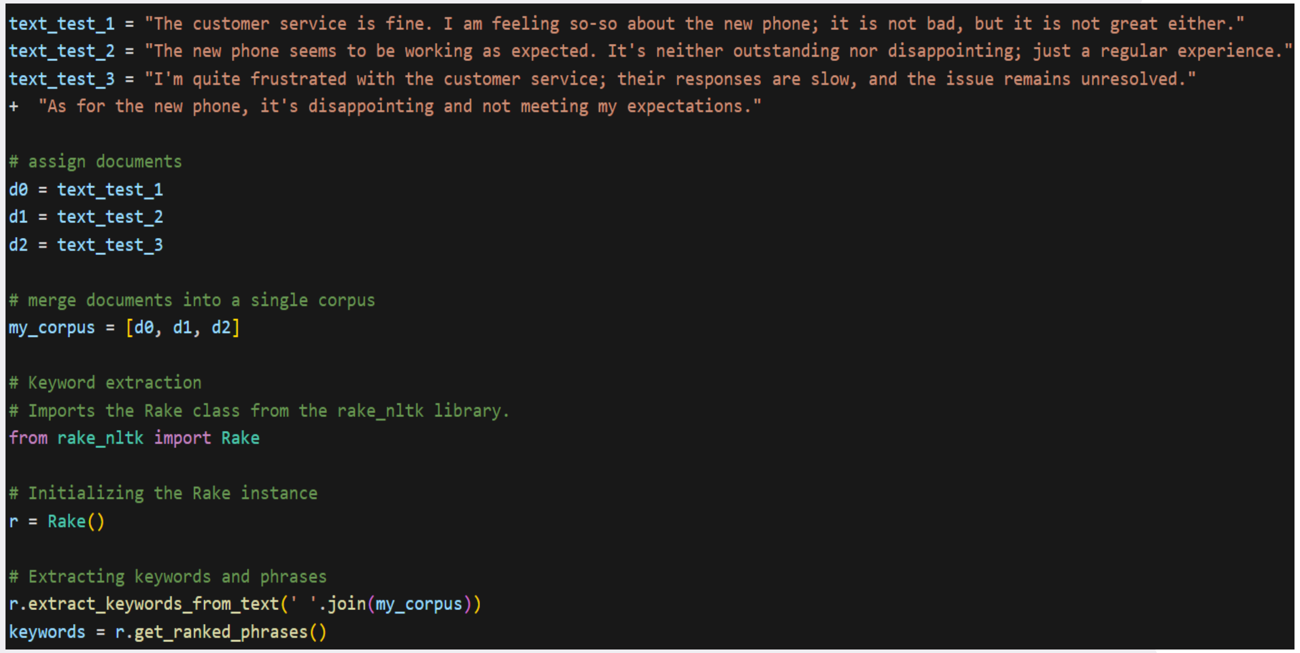
- ***Input 1*** = “The customer service is fine. I am feeling so-so about the new phone; it is not bad, but it is not great either.”

- ***Input 2*** = “The new phone seems to be working as expected. It's neither outstanding nor disappointing; just a regular experience.”

- ***Input 3 =*** “I'm quite frustrated with the customer service; their responses are slow, and the issue remains unresolved. As for the new phone, it's disappointing and not meeting my expectations.”

NLP tehnika: Ekstrakcija ključnih reči

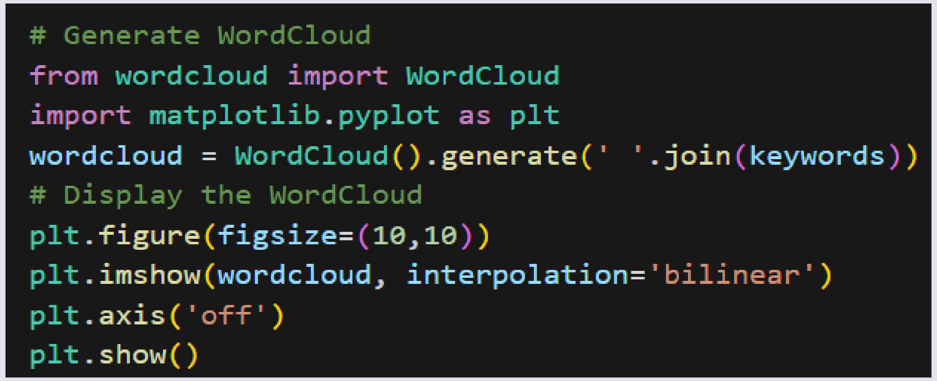
- Eng. Keyword Extraction
- Izdvaja ključne reči i fraze iz ulaznog teksta.
- **RAKE** (Rapid Automatic Keyword Extraction)



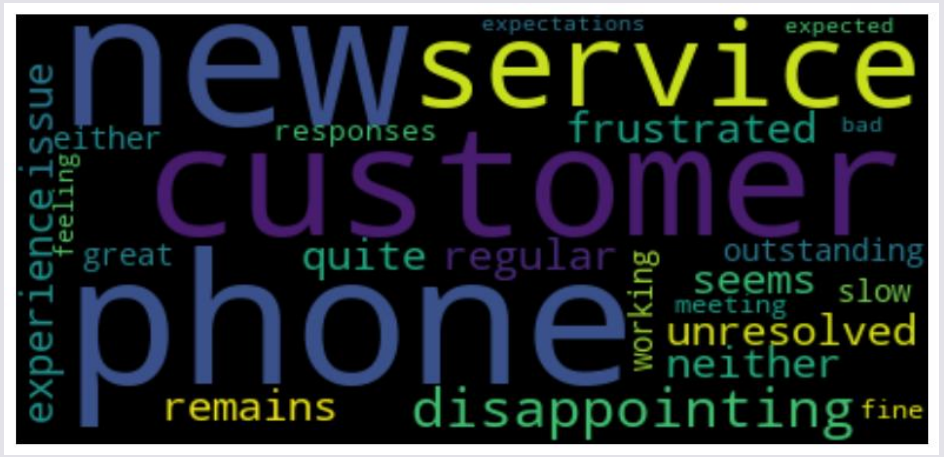
```
text_test_1 = "The customer service is fine. I am feeling so-so about the new phone; it is not bad, but it is not great either."
text_test_2 = "The new phone seems to be working as expected. It's neither outstanding nor disappointing; just a regular experience."
text_test_3 = "I'm quite frustrated with the customer service; their responses are slow, and the issue remains unresolved."
+  "As for the new phone, it's disappointing and not meeting my expectations."

# assign documents
d0 = text_test_1
d1 = text_test_2
d2 = text_test_3

# merge documents into a single corpus
my_corpus = [d0, d1, d2]

# Keyword extraction
# Imports the Rake class from the rake_nltk library.
from rake_nltk import Rake

# Initializing the Rake instance
r = Rake()

# Extracting keywords and phrases
r.extract_keywords_from_text(' '.join(my_corpus))
keywords = r.get_ranked_phrases()
```

*['issue remains unresolved', **'new phone seems'**, **'new phone'**, **'new phone'**, 'regular experience', 'quite frustrated', 'neither outstanding', 'great either', **'customer service'**, **'customer service'**, 'working', 'slow', 'responses', 'meeting', 'fine', 'feeling', 'expected', 'expectations', **'disappointing'**, **'disappointing'**, 'bad']*

NLP tehnika: Ekstrakcija ključnih reči



```python
# Generate WordCloud
from wordcloud import WordCloud
import matplotlib.pyplot as plt
wordcloud = WordCloud().generate(' '.join(keywords))
# Display the WordCloud
plt.figure(figsize=(10,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

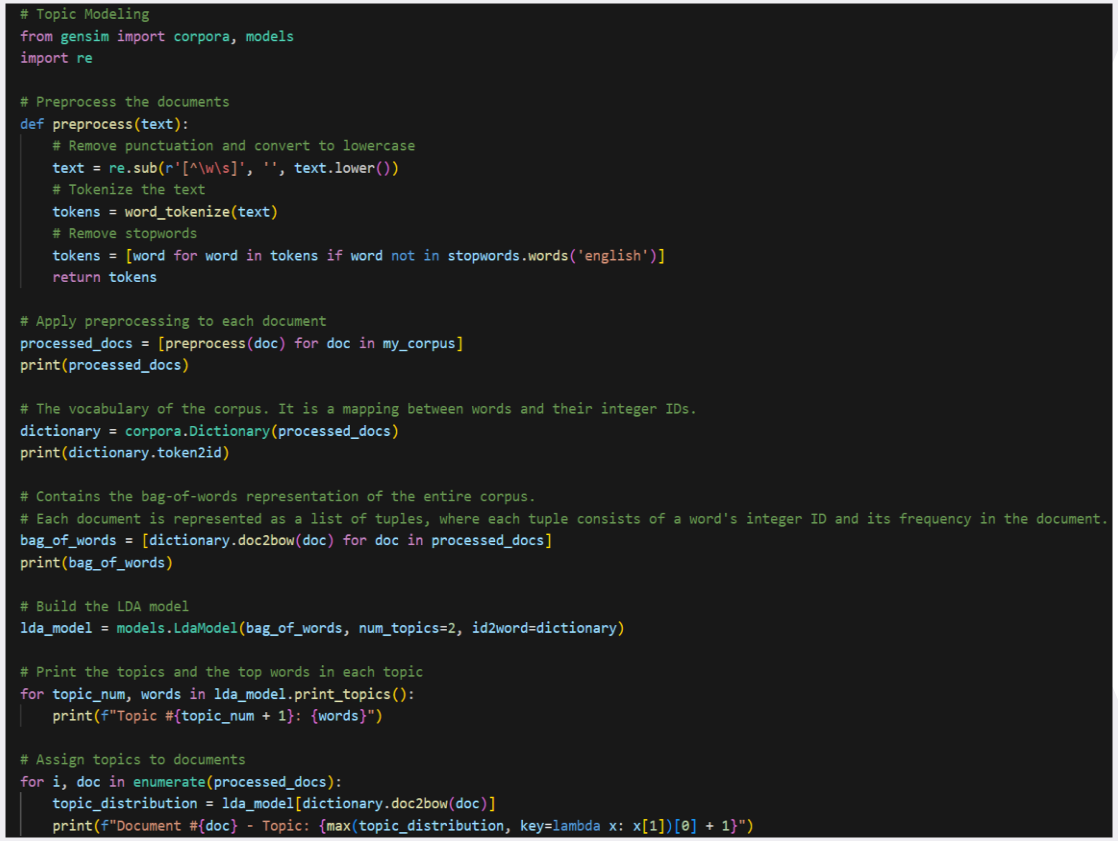


WordCloud

*['issue remains unresolved', **'new phone seems'**, **'new phone'**, **'new phone'**, 'regular experience', 'quite frustrated', 'neither outstanding', 'great either', **'customer service'**, **'customer service'**, 'working', 'slow', 'responses', 'meeting', 'fine', 'feeling', 'expected', 'expectations', **'disappointing'**, **'disappointing'**, 'bad']*

NLP tehnika: Modelovanje teme

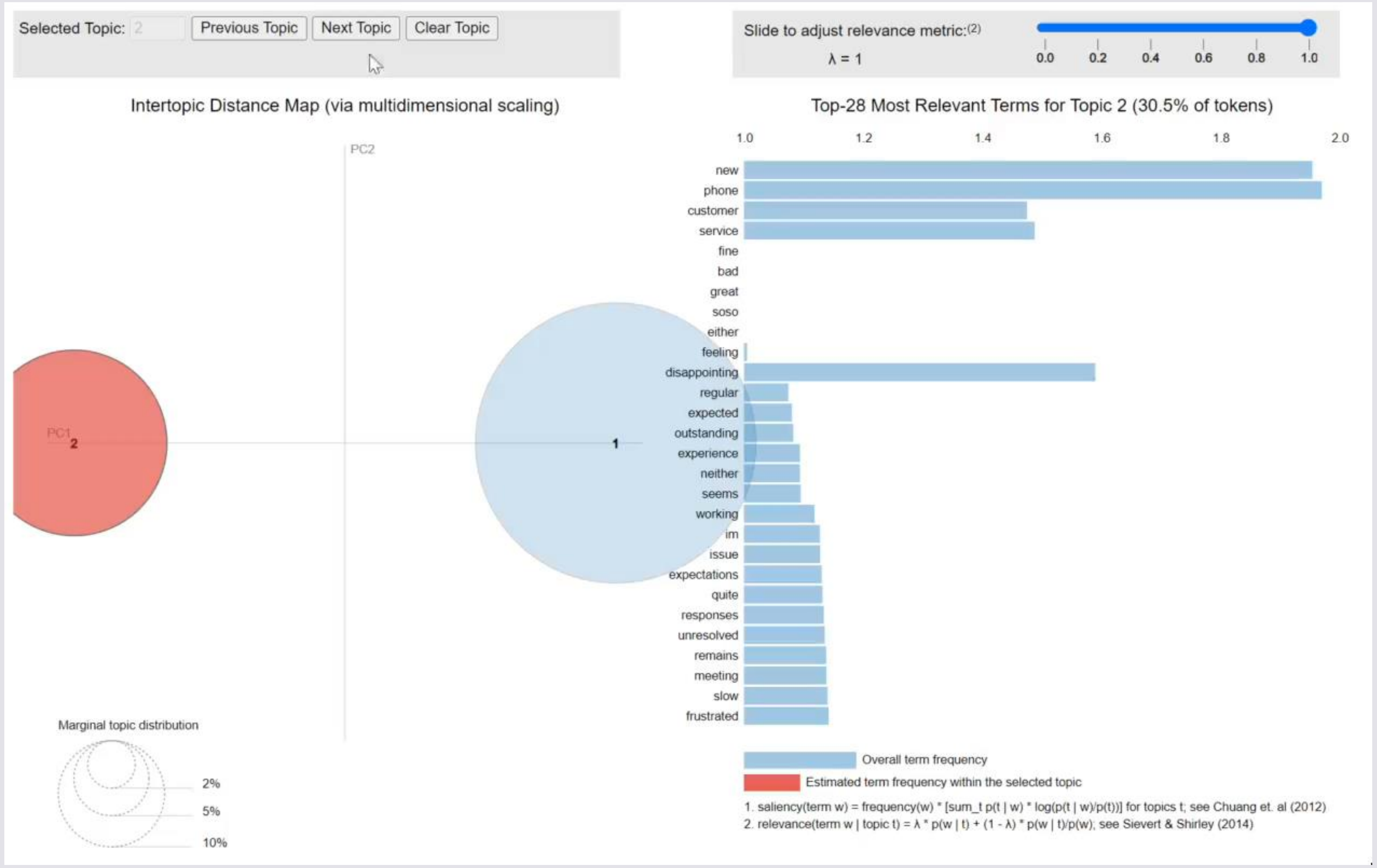
- Engl. Topic Modelling
- Otkriva (skrivene) teme u kolekciji dokumenata.
- **Latent Dirichlet Allocation** (**LDA**): svaki dokument je skup tema, a svaka tema je skup reči.



```python
# Topic Modeling
from gensim import corpora, models
import re

# Preprocess the documents
def preprocess(text):
    # Remove punctuation and convert to lowercase
    text = re.sub(r'[^\w\s]', '', text.lower())
    # Tokenize the text
    tokens = word_tokenize(text)
    # Remove stopwords
    tokens = [word for word in tokens if word not in stopwords.words('english')]
    return tokens

# Apply preprocessing to each document
processed_docs = [preprocess(doc) for doc in my_corpus]
print(processed_docs)

# The vocabulary of the corpus. It is a mapping between words and their integer IDs.
dictionary = corpora.Dictionary(processed_docs)
print(dictionary.token2id)

# Contains the bag-of-words representation of the entire corpus.
# Each document is represented as a list of tuples, where each tuple consists of a word's integer ID and its frequency in the document.
bag_of_words = [dictionary.doc2bow(doc) for doc in processed_docs]
print(bag_of_words)

# Build the LDA model
lda_model = models.LdaModel(bag_of_words, num_topics=2, id2word=dictionary)

# Print the topics and the top words in each topic
for topic_num, words in lda_model.print_topics():
    print(f"Topic #{topic_num + 1}: {words}")

# Assign topics to documents
for i, doc in enumerate(processed_docs):
    topic_distribution = lda_model[dictionary.doc2bow(doc)]
    print(f"Document #{doc} - Topic: {max(topic_distribution, key=lambda x: x[1])[0] + 1}")
```

```
[['customer', 'service', 'fine', 'feeling', 'soso', 'new', 'phone', 'bad', 'great', 'either'], ['new',
'phone', 'seems', 'working', 'expected', 'neither', 'outstanding', 'disappointing', 'regular',
'experience'], ['im', 'quite', 'frustrated', 'customer', 'service', 'responses', 'slow', 'issue',
'remains', 'unresolved', 'new', 'phone', 'disappointing', 'meeting', 'expectations’]]

{'bad': 0, 'customer': 1, 'either': 2, 'feeling': 3, 'fine': 4, 'great': 5, 'new': 6, 'phone': 7,
'service': 8, 'soso': 9, 'disappointing': 10, 'expected': 11, 'experience': 12, 'neither': 13,
'outstanding': 14, 'regular': 15, 'seems': 16, 'working': 17, 'expectations': 18, 'frustrated': 19, 'im':
20, 'issue': 21, 'meeting': 22, 'quite': 23, 'remains': 24, 'responses': 25, 'slow': 26, 'unresolved': 27}

[[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1)], [(6, 1), (7, 1), (10,
1), (11, 1), (12, 1), (13, 1), (14, 1), (15, 1), (16, 1), (17, 1)], [(1, 1), (6, 1), (7, 1), (8, 1), (10,
1), (18, 1), (19, 1), (20, 1), (21, 1), (22, 1), (23, 1), (24, 1), (25, 1), (26, 1), (27, 1)]]

Topic #1: 0.061*"new" + 0.059*"disappointing" + 0.047*"customer" + 0.042*"phone" +
0.041*"responses" + 0.039*"slow" + 0.039*"frustrated" + 0.039*"service" + 0.038*"im"
+ 0.038*"expectations"
Topic #2: 0.078*"phone" + 0.065*"new" + 0.053*"service" + 0.048*"customer" +
0.040*"disappointing" + 0.038*"soso" + 0.038*"bad" + 0.038*"great" + 0.038*"fine" +
0.037*"either"

Document #['customer', 'service', 'fine', 'feeling', 'soso', 'new', 'phone', 'bad',
'great', 'either'] - Topic: 2
Document #['new', 'phone', 'seems', 'working', 'expected', 'neither', 'outstanding',
'disappointing', 'regular', 'experience'] - Topic: 2
Document #['im', 'quite', 'frustrated', 'customer', 'service', 'responses', 'slow',
'issue', 'remains', 'unresolved', 'new', 'phone', 'disappointing', 'meeting',
'expectations'] - Topic: 1
```

NLP tehnika: Modelovanje teme



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

NLP tehnika: TF-IDF

Term Frequency-Inverse Document Frequency (TF-IDF) koristi se za određivanje značaja reči u dokumentu ili korpusu.

TF (učestalost termina) - koliko puta se neka reč pojavila u datom dokumentu

IDF (inverzna učestalost) - koliko informacija pruža reč, odnosno koliko je uobičajena ili retka u svim dokumentima.

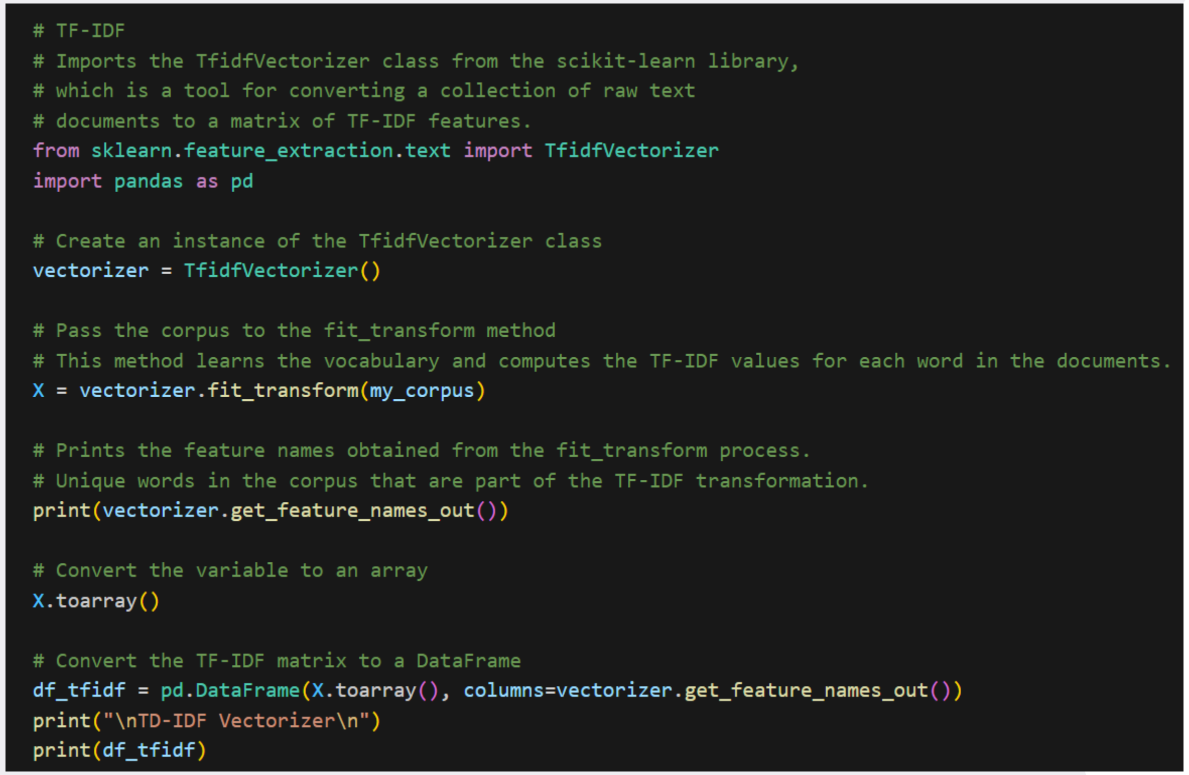
$$\mathrm{tf}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}},$$

	the	customer	service	is	fine	new	phone	not	disappointing	expect
$D1$	1	1	1	1	1	0	0	0	0	0
$D2$	2	0	0	1	0	1	1	2	0	0
$D3$	1	0	0	0	0	1	1	0	0	1
$D4$	1	0	0	0	0	0	0	0	1	0
$D5$	1	1	1	0	0	0	0	0	0	0
$D6$	1	0	0	0	0	1	1	1	1	0

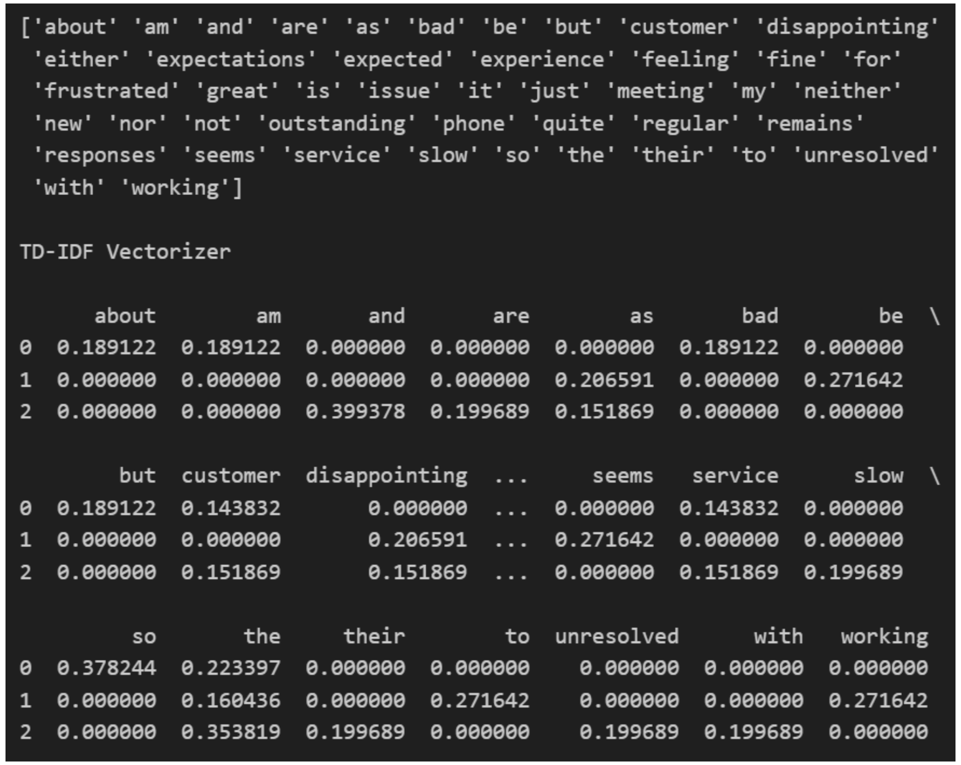
$$\mathrm{idf}(t,D) = \log \frac{N}{|\{d : d \in D \text{ and } t \in d\}|}$$

Term	IDF
the	0
customer	$\log\left(\frac{6}{2}\right) = \log(3) \approx 1.0986$
service	$\log\left(\frac{6}{2}\right) = \log(3) \approx 1.0986$
is	$\log\left(\frac{6}{2}\right) = \log(3) \approx 1.0986$
fine	$\log\left(\frac{6}{1}\right) = \log(6) \approx 1.7918$
new	$\log\left(\frac{6}{4}\right) = \log(1.5) \approx 0.4055$
phone	$\log\left(\frac{6}{4}\right) = \log(1.5) \approx 0.4055$
not	$\log\left(\frac{6}{3}\right) = \log(2) \approx 0.6931$
disappointing	$\log\left(\frac{6}{2}\right) = \log(3) \approx 1.0986$
expected	$\log\left(\frac{6}{1}\right) = \log(6) \approx 1.7918$
and	$\log\left(\frac{6}{2}\right) = \log(3) \approx 1.0986$
with	$\log\left(\frac{6}{1}\right) = \log(6) \approx 1.7918$
responses	$\log\left(\frac{6}{1}\right) = \log(6) \approx 1.7918$
slow	$\log\left(\frac{6}{1}\right) = \log(6) \approx 1.7918$
issue	$\log\left(\frac{6}{1}\right) = \log(6) \approx 1.7918$
frustrated	$\log\left(\frac{6}{1}\right) = \log(6) \approx 1.7918$
meeting	$\log\left(\frac{6}{1}\right) = \log(6) \approx 1.7918$
expectations	$\log\left(\frac{6}{1}\right) = \log(6) \approx 1.7918$

NLP tehnika: TF-IDF

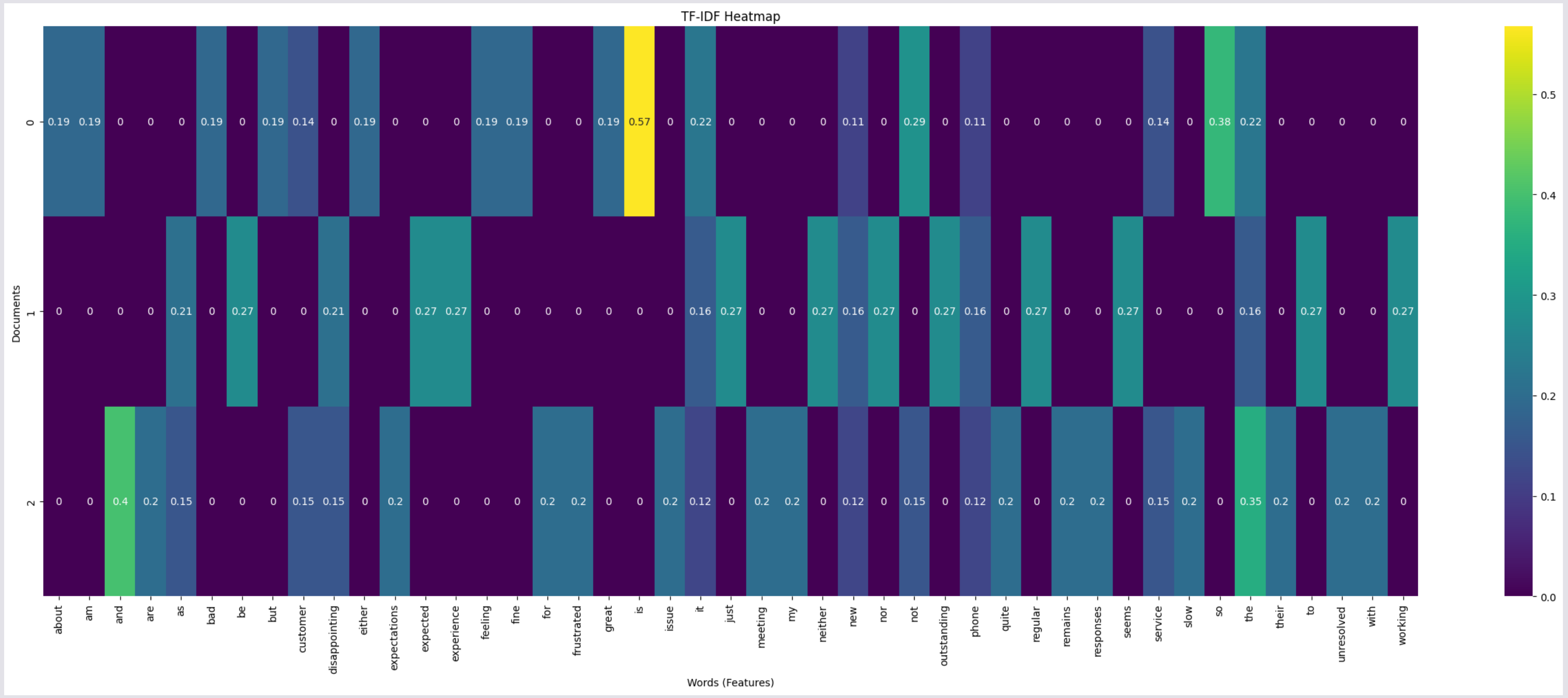


```python
# TF-IDF
# Imports the TfidfVectorizer class from the scikit-learn library,
# which is a tool for converting a collection of raw text
# documents to a matrix of TF-IDF features.
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd

# Create an instance of the TfidfVectorizer class
vectorizer = TfidfVectorizer()

# Pass the corpus to the fit_transform method
# This method learns the vocabulary and computes the TF-IDF values for each word in the documents.
X = vectorizer.fit_transform(my_corpus)

# Prints the feature names obtained from the fit_transform process.
# Unique words in the corpus that are part of the TF-IDF transformation.
print(vectorizer.get_feature_names_out())

# Convert the variable to an array
X.toarray()

# Convert the TF-IDF matrix to a DataFrame
df_tfidf = pd.DataFrame(X.toarray(), columns=vectorizer.get_feature_names_out())
print("\nTD-IDF Vectorizer\n")
print(df_tfidf)
```



```
['about' 'am' 'and' 'are' 'as' 'bad' 'be' 'but' 'customer' 'disappointing'
 'either' 'expectations' 'expected' 'experience' 'feeling' 'fine' 'for'
 'frustrated' 'great' 'is' 'issue' 'it' 'just' 'meeting' 'my' 'neither'
 'new' 'nor' 'not' 'outstanding' 'phone' 'quite' 'regular' 'remains'
 'responses' 'seems' 'service' 'slow' 'so' 'the' 'their' 'to' 'unresolved'
 'with' 'working']

TD-IDF Vectorizer

      about        am       and       are        as       bad        be  \
0  0.189122  0.189122  0.000000  0.000000  0.000000  0.189122  0.000000
1  0.000000  0.000000  0.000000  0.000000  0.206591  0.000000  0.271642
2  0.000000  0.000000  0.399378  0.199689  0.151869  0.000000  0.000000

        but  customer  disappointing  ...     seems   service      slow  \
0  0.189122  0.143832       0.000000  ...  0.000000  0.143832  0.000000
1  0.000000  0.000000       0.206591  ...  0.271642  0.000000  0.000000
2  0.000000  0.151869       0.151869  ...  0.000000  0.151869  0.199689

         so       the     their        to  unresolved      with   working
0  0.378244  0.223397  0.000000  0.000000    0.000000  0.000000  0.000000
1  0.000000  0.160436  0.000000  0.271642    0.000000  0.000000  0.271642
2  0.000000  0.353819  0.199689  0.000000    0.199689  0.199689  0.000000
```

NLP tehnika: TF-IDF



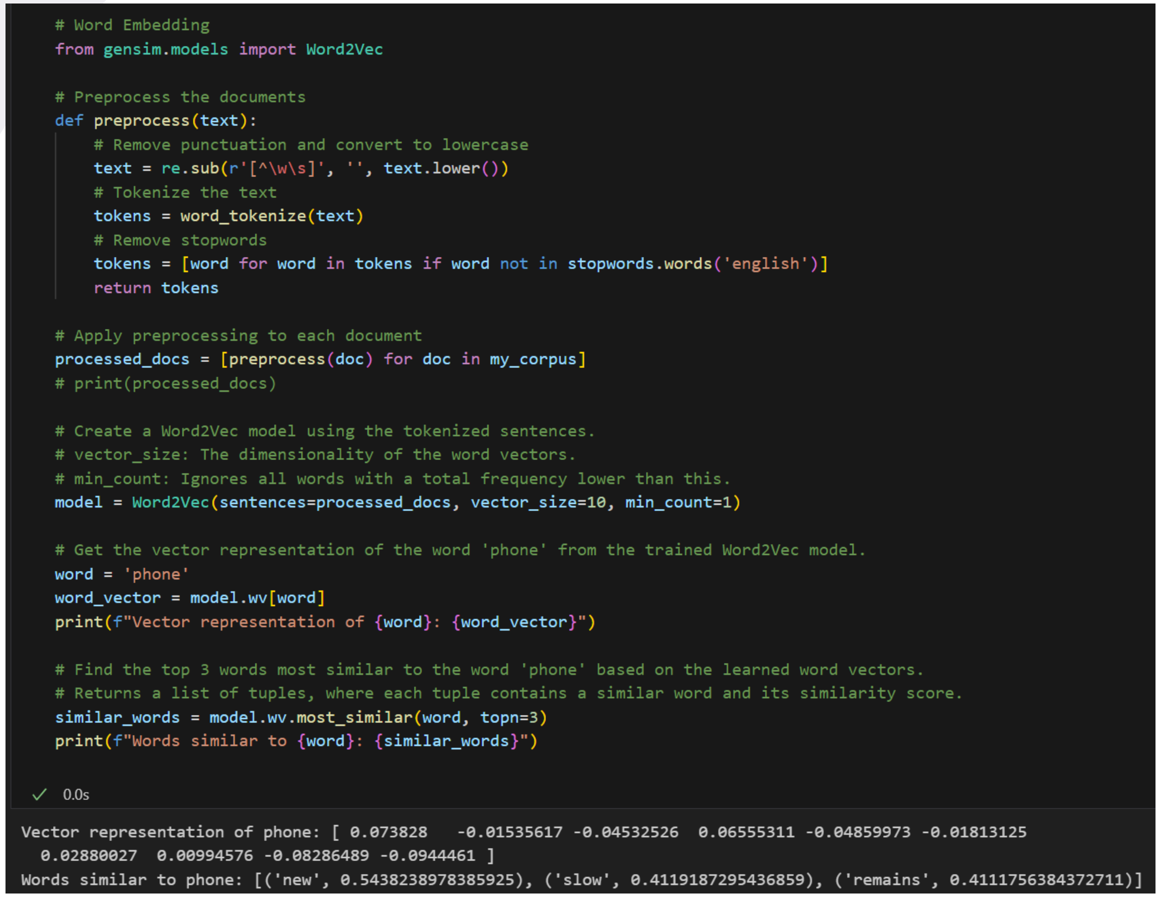
Documents	about	am	and	are	as	bad	be	but	customer	disappointing	either	expectations	expected	experience	feeling
0	0.19	0.19	0	0	0	0.19	0	0.19	0.14	0	0.19	0	0	0	0.19
1	0	0	0	0	0.21	0	0.27	0	0	0.21	0	0	0.27	0.27	0
2	0	0	0.4	0.2	0.15	0	0	0	0.15	0.15	0	0.2	0	0	0

Documents	fine	for	frustrated	great	is	issue	it	just	meeting	my	neither	new	nor	not	outstanding
0	0.19	0	0	0.19	0.57	0	0.22	0	0	0	0	0.11	0	0.29	0
1	0	0	0	0	0	0	0.16	0.27	0	0	0.27	0.16	0.27	0	0.27
2	0	0.2	0.2	0	0	0.2	0.12	0	0.2	0.2	0	0.12	0	0.15	0

Documents	phone	quite	regular	remains	responses	seems	service	slow	so	the	their	to	unresolved	with	working
0	0.11	0	0	0	0	0	0.14	0	0.38	0.22	0	0	0	0	0
1	0.16	0	0.27	0	0	0.27	0	0	0	0.16	0	0.27	0	0	0.27
2	0.12	0.2	0	0.2	0.2	0	0.15	0.2	0	0.35	0.2	0	0.2	0.2	0

NLP tehnika: Ugrađivanje reči

- Engl. Word Embeddings/Word vectorization
- Preslikava reči ili fraze iz rečnika u odgovarajući vektor realnih brojeva koji se koristi za predviđanje reči ili utvrđivanje sličnosti.
- **Kosinusna sličnost (engl. Cosine similarity)**: meri kosinus ugla između dva vektora u N-dimenzionalnom vektorskom prostoru.
- **Word2Vec**

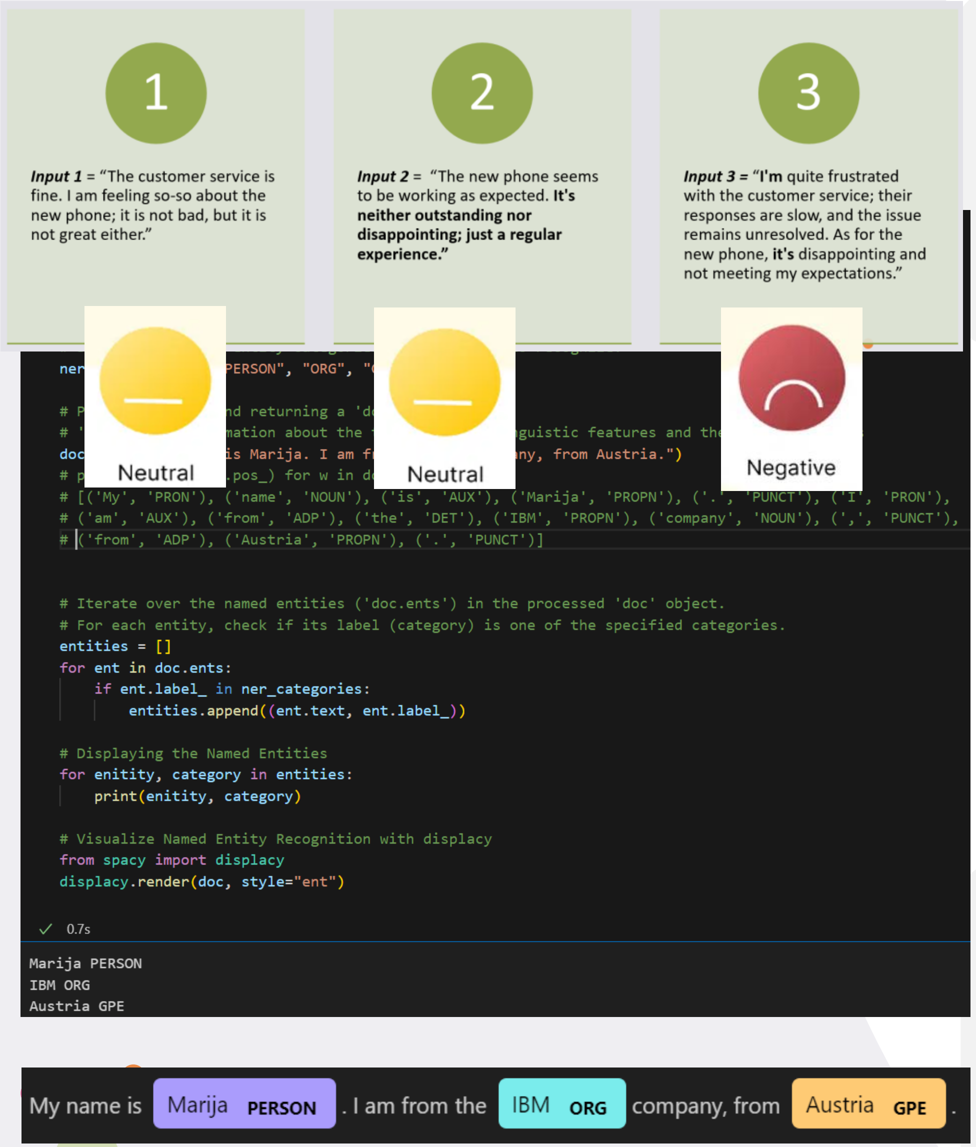


```python
# Word Embedding
from gensim.models import Word2Vec

# Preprocess the documents
def preprocess(text):
    # Remove punctuation and convert to lowercase
    text = re.sub(r'[^\w\s]', '', text.lower())
    # Tokenize the text
    tokens = word_tokenize(text)
    # Remove stopwords
    tokens = [word for word in tokens if word not in stopwords.words('english')]
    return tokens

# Apply preprocessing to each document
processed_docs = [preprocess(doc) for doc in my_corpus]
# print(processed_docs)

# Create a Word2Vec model using the tokenized sentences.
# vector_size: The dimensionality of the word vectors.
# min_count: Ignores all words with a total frequency lower than this.
model = Word2Vec(sentences=processed_docs, vector_size=10, min_count=1)

# Get the vector representation of the word 'phone' from the trained Word2Vec model.
word = 'phone'
word_vector = model.wv[word]
print(f"Vector representation of {word}: {word_vector}")

# Find the top 3 words most similar to the word 'phone' based on the learned word vectors.
# Returns a list of tuples, where each tuple contains a similar word and its similarity score.
similar_words = model.wv.most_similar(word, topn=3)
print(f"Words similar to {word}: {similar_words}")
```

✓ 0.0s

```
Vector representation of phone: [ 0.073828   -0.01535617 -0.04532526  0.06555311 -0.04859973 -0.01813125
  0.02880027  0.00994576 -0.08286489 -0.0944461 ]
Words similar to phone: [('new', 0.5438238978385925), ('slow', 0.4119187295436859), ('remains', 0.4111756384372711)]
```

NLP zadaci



Input 1 = "The customer service is fine. I am feeling so-so about the new phone; it is not bad, but it is not great either."

Input 2 = "The new phone seems to be working as expected. **It's neither outstanding nor disappointing; just a regular experience.**"

Input 3 = "**I'm** quite frustrated with the customer service; their responses are slow, and the issue remains unresolved. As for the new phone, **it's** disappointing and not meeting my expectations."

```
ner                PERSON", "ORG", "

# P                nd returning a 'd
# '                mation about the                 guistic features and the
doc                is Marija. I am fr               ny, from Austria.")
# p                .pos_) for w in d
# [('My', 'PRON'), ('name', 'NOUN'), ('is', 'AUX'), ('Marija', 'PROPN'), ('.', 'PUNCT'), ('I', 'PRON'),
# ('am', 'AUX'), ('from', 'ADP'), ('the', 'DET'), ('IBM', 'PROPN'), ('company', 'NOUN'), (',', 'PUNCT'),
# ('from', 'ADP'), ('Austria', 'PROPN'), ('.', 'PUNCT')]


# Iterate over the named entities ('doc.ents') in the processed 'doc' object.
# For each entity, check if its label (category) is one of the specified categories.
entities = []
for ent in doc.ents:
    if ent.label_ in ner_categories:
        entities.append((ent.text, ent.label_))

# Displaying the Named Entities
for enitity, category in entities:
    print(enitity, category)

# Visualize Named Entity Recognition with displacy
from spacy import displacy
displacy.render(doc, style="ent")
```

✓ 0.7s

```
Marija PERSON
IBM ORG
Austria GPE
```

My name is Marija PERSON . I am from the IBM ORG company, from Austria GPE .

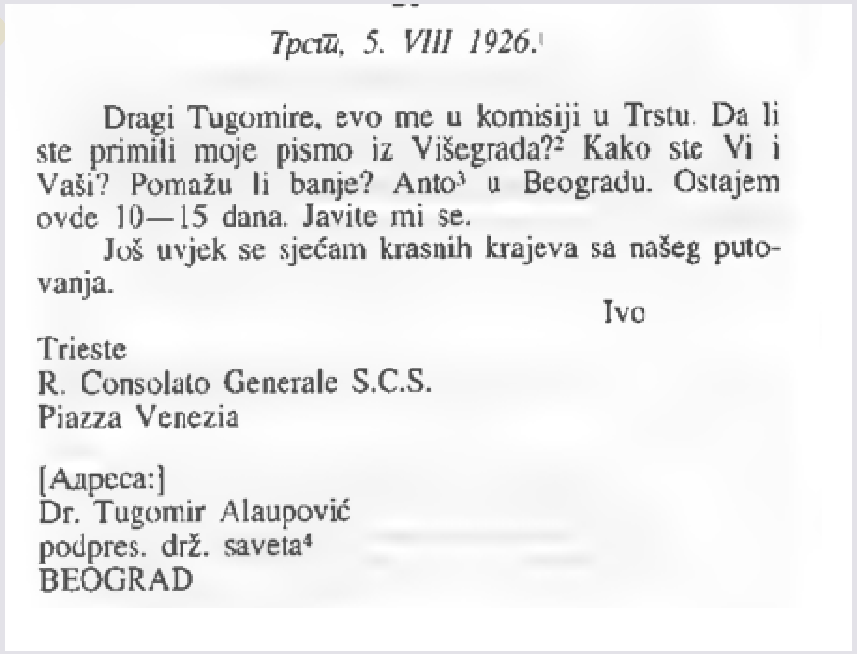
Analiza sentimenta (engl. Sentiment Analysis): klasifikovanje teksta na osnovu raspoloženja ili sentimenta.

Primene: nadgledanje društvenih medija i korisnička podrška (šta klijenti misle o nekom proizvodu, brendu ili usluzi), tržišni sektor (praćenje javnog raspoloženja o finansijskim sredstvima, tržišnim trendovima i ekonomskim pokazateljima)

Klasifikacija teksta (engl. Text classification): označavanje i razvrstavanje teksta u različite kategorije na osnovu njihovog sadržaja.

Primene: filtriranje e-maila (spam e-mails), **identifikacija jezika** (Google Translate), **identifikacija starosti/pola anonimnih korinsika**, **označavanje online sadržaja**, **prepoznavanje govora koji koriste virtuelni asistenti** (Siri and Alexa)

Prepoznavanje imenovanih entiteta (engl. Named Entity Recognition): klasifikovanje imenovanih entiteta u unapred definisane kategorije kao što su imena osoba, organizacije, lokacije, vremenski izrazi, količine,...

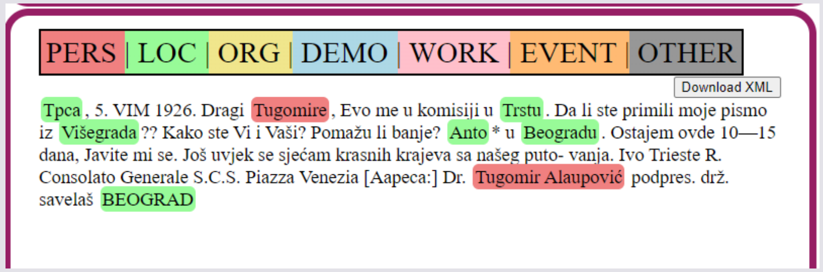

Трсш, 5. VIII 1926.[1]

Dragi Tugomire, evo me u komisiji u Trstu. Da li ste primili moje pismo iz Višegrada?[2] Kako ste Vi i Vaši? Pomažu li banje? Anto[3] u Beogradu. Ostajem ovde 10—15 dana. Javite mi se.
Još uvjek se sjećam krasnih krajeva sa našeg puto-vanja.

Ivo

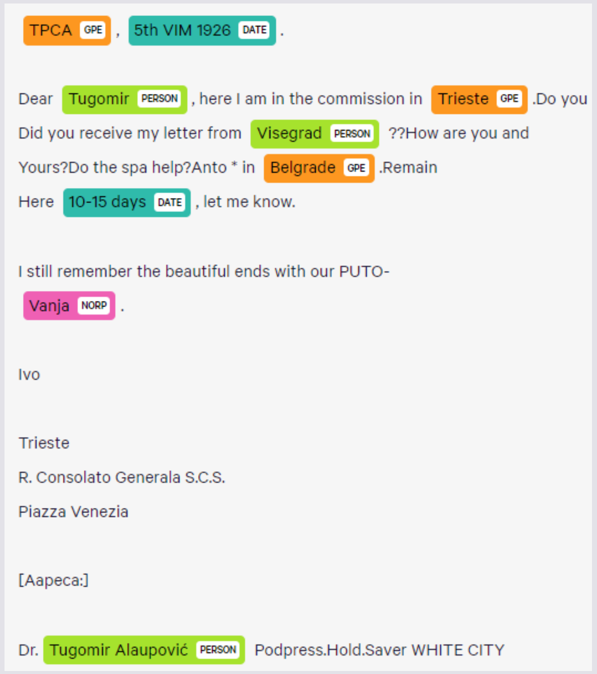
Trieste
R. Consolato Generale S.C.S.
Piazza Venezia

[Адреса:]
Dr. Tugomir Alaupović
podpres. drž. saveta[4]
BEOGRAD


PERS | LOC | ORG | DEMO | WORK | EVENT | OTHER

Download XML

Tpca , 5. VIM 1926. Dragi Tugomire , Evo me u komisiji u Trstu . Da li ste primili moje pismo iz Višegrada ?? Kako ste Vi i Vaši? Pomažu li banje? Anto * u Beogradu . Ostajem ovde 10—15 dana, Javite mi se. Još uvjek se sjećam krasnih krajeva sa našeg puto- vanja. Ivo Trieste R. Consolato Generale S.C.S. Piazza Venezia [Aapeca:] Dr. Tugomir Alaupović podpres. drž. savelaš BEOGRAD


TPCA GPE , 5th VIM 1926 DATE .

Dear Tugomir PERSON , here I am in the commission in Trieste GPE .Do you
Did you receive my letter from Visegrad PERSON ??How are you and
Yours?Do the spa help?Anto * in Belgrade GPE .Remain
Here 10-15 days DATE , let me know.

I still remember the beautiful ends with our PUTO-
Vanja NORP .

Ivo

Trieste
R. Consolato Generala S.C.S.
Piazza Venezia

[Aapeca:]

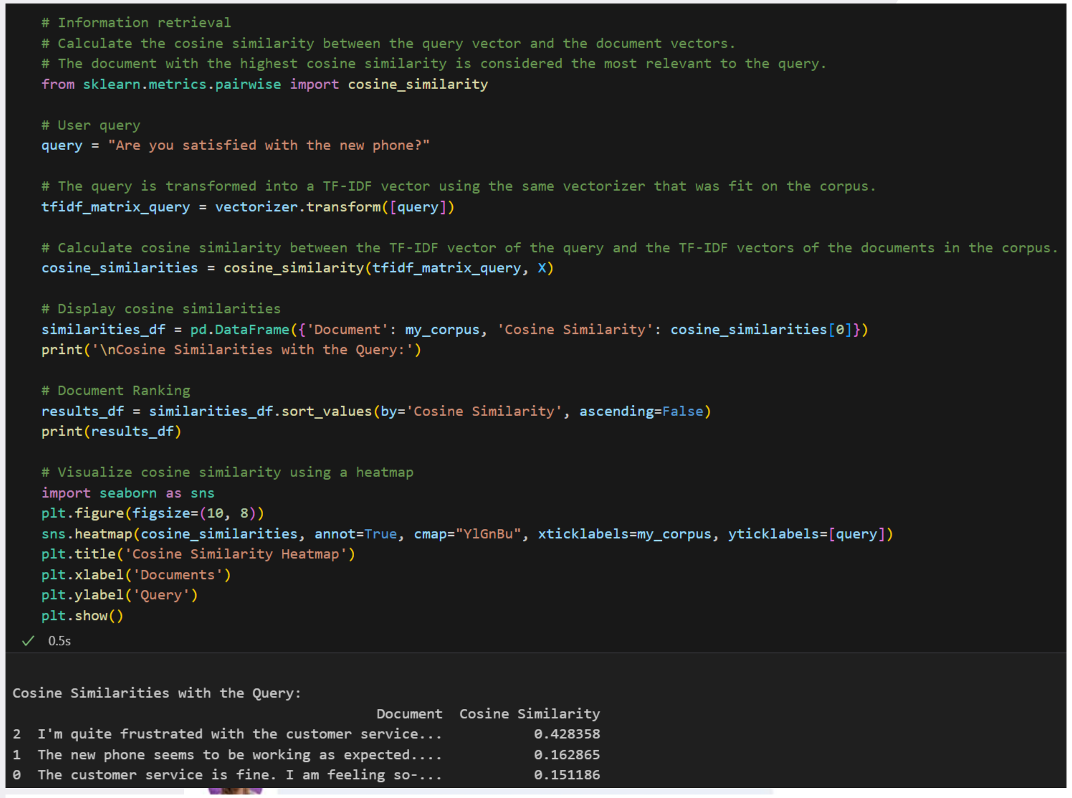
Dr. Tugomir Alaupović PERSON Podpress.Hold.Saver WHITE CITY

comment

"Tugomir Marko Alaupović (Dolac, Travnik, 18 August 1870 – Zagreb, 9 April 1958) was a Yugoslav professor at "First Grammar School ,Sarajevo, poet, storyteller and politician. In addition to his rich political biography, he was also minister of religion in the government of the Kingdom of Serbs, Croats and Slovenes. He has written several literary works that have been translated into French, German, Czech and Italian. He was one of the initiators of the Croatian Society for the "Setting up of Children in Crafts and Trade" in Sarajevo and later initiated the change of the society name to Napredak. He was a member of the Main Board of the Serbian St. Sava Society in Belgrade. On 16 January 1934, after a serious operation, in a letter to Tihomir Djordjevic, a prominent Serbian ethnologist, he "@en

Unveiling Literary Legacies: Integrating Named Entity Recognition and SPARQL for Analyzing Andrić's Letters, Marija Đokić Petrović, Dragana Bečejski Vujaklija, Jasmina Đorđević, Jelena Mitrović, 16. Symposiym „Andrić's letters", October 2024, Graz

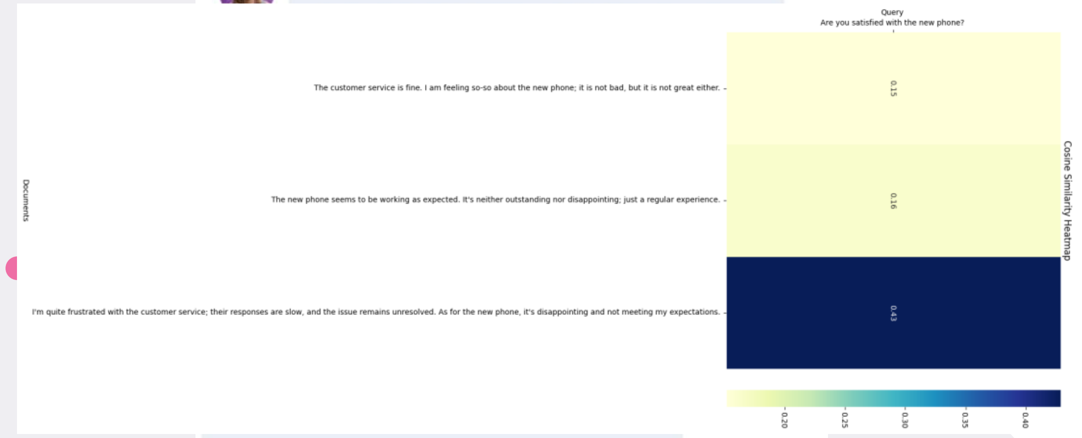
NLP zadaci



```
# Information retrieval
# Calculate the cosine similarity between the query vector and the document vectors.
# The document with the highest cosine similarity is considered the most relevant to the query.
from sklearn.metrics.pairwise import cosine_similarity

# User query
query = "Are you satisfied with the new phone?"

# The query is transformed into a TF-IDF vector using the same vectorizer that was fit on the corpus.
tfidf_matrix_query = vectorizer.transform([query])

# Calculate cosine similarity between the TF-IDF vector of the query and the TF-IDF vectors of the documents in the corpus.
cosine_similarities = cosine_similarity(tfidf_matrix_query, X)

# Display cosine similarities
similarities_df = pd.DataFrame({'Document': my_corpus, 'Cosine Similarity': cosine_similarities[0]})
print('\nCosine Similarities with the Query:')

# Document Ranking
results_df = similarities_df.sort_values(by='Cosine Similarity', ascending=False)
print(results_df)

# Visualize cosine similarity using a heatmap
import seaborn as sns
plt.figure(figsize=(10, 8))
sns.heatmap(cosine_similarities, annot=True, cmap="YlGnBu", xticklabels=my_corpus, yticklabels=[query])
plt.title('Cosine Similarity Heatmap')
plt.xlabel('Documents')
plt.ylabel('Query')
plt.show()
```

✓ 0.5s

```
Cosine Similarities with the Query:
                                         Document  Cosine Similarity
2  I'm quite frustrated with the customer service...           0.428358
1  The new phone seems to be working as expected....           0.162865
0  The customer service is fine. I am feeling so-...           0.151186
```



Sumiranje teksta (engl. Text Summarization): sintetizuje velika korpuse tekstova u njihove najvažnije delove.

Primena: **sumiranje akademskih radova, sumiranje blogova, sumiranje online vesti, Chatbot odgovori, ….**

Preuzimanje informacija (engl. Information Retrieval): preuzimanje i analize informacija iz tekstualnih dokumenata na osnovu određenog upita koji je dao korisnik.

Primene: Veb pretraživač (engl. **Web search engine)**, sistemi za odgovore na pitanja (engl. **Question answering systems)**, lični asistenti, **Chatbots, digitalne biblioteke**

Rangiranje dokumenta (engl. Document Ranking): dodeljivanje numeričke ocene (rang) dokumentima na osnovu njihove relevantnosti za određeni upit. Dokumenti se rangiraju na osnovu njihove kosinusne sličnosti sa upitom.

Veliki jezički modeli (engl. Large Language Models)

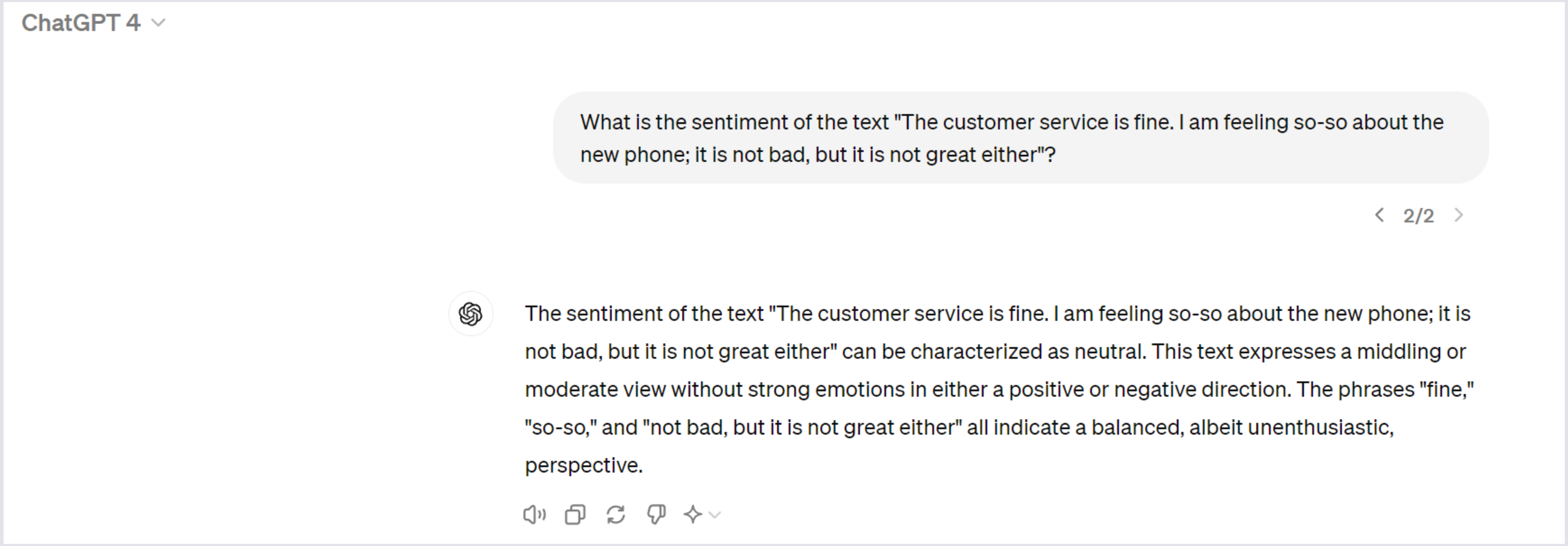
Veliki jezički model (LLM) je algoritam dubokog učenja (engl. deep learning algorithm) koji može da obavlja različite NLP zadatke.

Treniranje (engl. Training): koristeći velike tekstualne skupove podataka sa sajtova kao što su Wikipedia, GitHub; **Fino podešavanje (engl. Fine-tuning):** parametri prethodno obučenog modela se obučavaju na novim podacima; **Zaključivanje (engl. Inference):** model uzima input (prompt) koje definiše korinsik i generiše odgovarajući izlaz.

Primene: Generisanje koda/tekstualnih sadržaja, moderiranje sadržaja, odgovaranje na pitanja (engl. Question Answering), Chatbots i konverzacijski AI, obrazovne alate, Preuzimanje informacija (engl. Information retrieval), sumiranje teksta (engl. Text summarization), Analiza sentimenta (engl. Sentiment analysis)

GPT(Generative Pre-trained Transformer): OpenAI, sastoji se od 175 milijardi parametara i pokazao je snažne performanse u različitim NLP zadacima. **BERT (Bidirectional Encoder Representations from Transformers):** Google, dvosmerni trening, razume kontekst sa obe strane, koristi se za odgovaranje na pitanja i razumevanje jezika. **RoBERTa (Robustly optimized BERT approach):** Facebook AI, optimizovana verzija BERT-a.

Veliki jezički modeli - ChatGPT



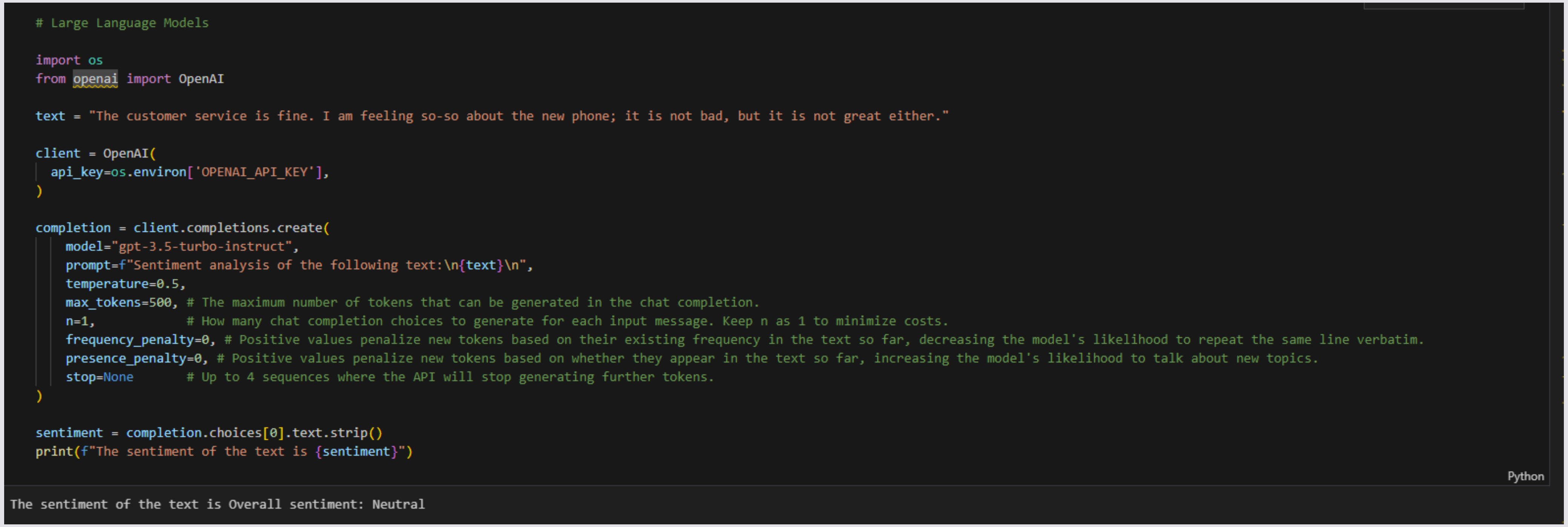
ChatGPT 4

What is the sentiment of the text "The customer service is fine. I am feeling so-so about the new phone; it is not bad, but it is not great either"?

2/2

The sentiment of the text "The customer service is fine. I am feeling so-so about the new phone; it is not bad, but it is not great either" can be characterized as neutral. This text expresses a middling or moderate view without strong emotions in either a positive or negative direction. The phrases "fine," "so-so," and "not bad, but it is not great either" all indicate a balanced, albeit unenthusiastic, perspective.

Veliki jezički modeli - GPT OpenAI



```python
# Large Language Models

import os
from openai import OpenAI

text = "The customer service is fine. I am feeling so-so about the new phone; it is not bad, but it is not great either."

client = OpenAI(
  api_key=os.environ['OPENAI_API_KEY'],
)

completion = client.completions.create(
    model="gpt-3.5-turbo-instruct",
    prompt=f"Sentiment analysis of the following text:\n{text}\n",
    temperature=0.5,
    max_tokens=500, # The maximum number of tokens that can be generated in the chat completion.
    n=1,            # How many chat completion choices to generate for each input message. Keep n as 1 to minimize costs.
    frequency_penalty=0, # Positive values penalize new tokens based on their existing frequency in the text so far, decreasing the model's likelihood to repeat the same line verbatim.
    presence_penalty=0, # Positive values penalize new tokens based on whether they appear in the text so far, increasing the model's likelihood to talk about new topics.
    stop=None       # Up to 4 sequences where the API will stop generating further tokens.
)

sentiment = completion.choices[0].text.strip()
print(f"The sentiment of the text is {sentiment}")
```

Python

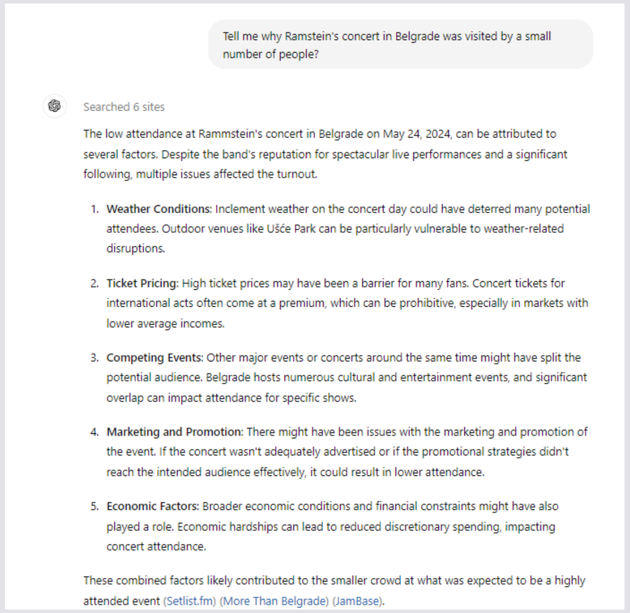
```
The sentiment of the text is Overall sentiment: Neutral
```

Veliki jezički modeli - izazovi

- Nedostaci: halicinacije

- **Halucinacije**: rezultati koji su sintaksički i semantički ispravni, ali odvojeni od stvarnosti i zasnovani na lažnim pretpostavkama

- **Retrieval-Augmented Generation (RAG)**



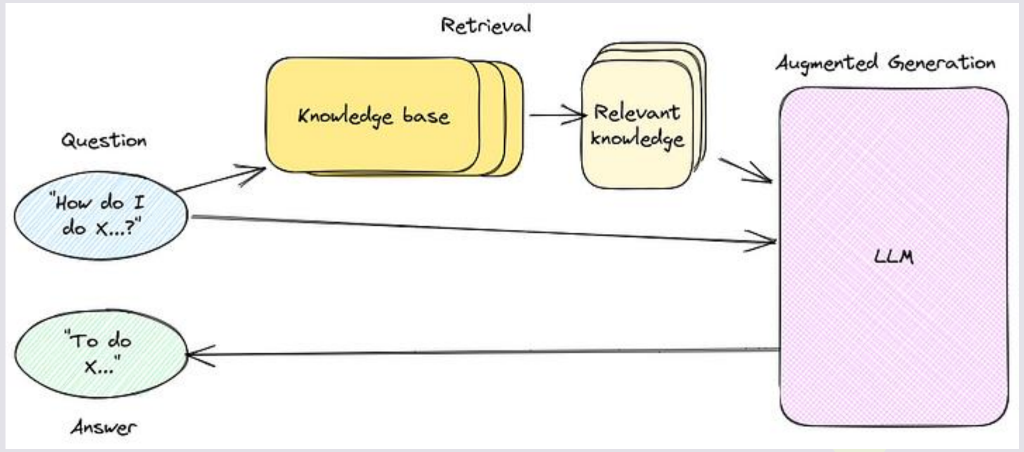
Tell me why Ramstein's concert in Belgrade was visited by a small number of people?

Searched 6 sites

The low attendance at Rammstein's concert in Belgrade on May 24, 2024, can be attributed to several factors. Despite the band's reputation for spectacular live performances and a significant following, multiple issues affected the turnout.

1. **Weather Conditions**: Inclement weather on the concert day could have deterred many potential attendees. Outdoor venues like Ušće Park can be particularly vulnerable to weather-related disruptions.
2. **Ticket Pricing**: High ticket prices may have been a barrier for many fans. Concert tickets for international acts often come at a premium, which can be prohibitive, especially in markets with lower average incomes.
3. **Competing Events**: Other major events or concerts around the same time might have split the potential audience. Belgrade hosts numerous cultural and entertainment events, and significant overlap can impact attendance for specific shows.
4. **Marketing and Promotion**: There might have been issues with the marketing and promotion of the event. If the concert wasn't adequately advertised or if the promotional strategies didn't reach the intended audience effectively, it could result in lower attendance.
5. **Economic Factors**: Broader economic conditions and financial constraints might have also played a role. Economic hardships can lead to reduced discretionary spending, impacting concert attendance.

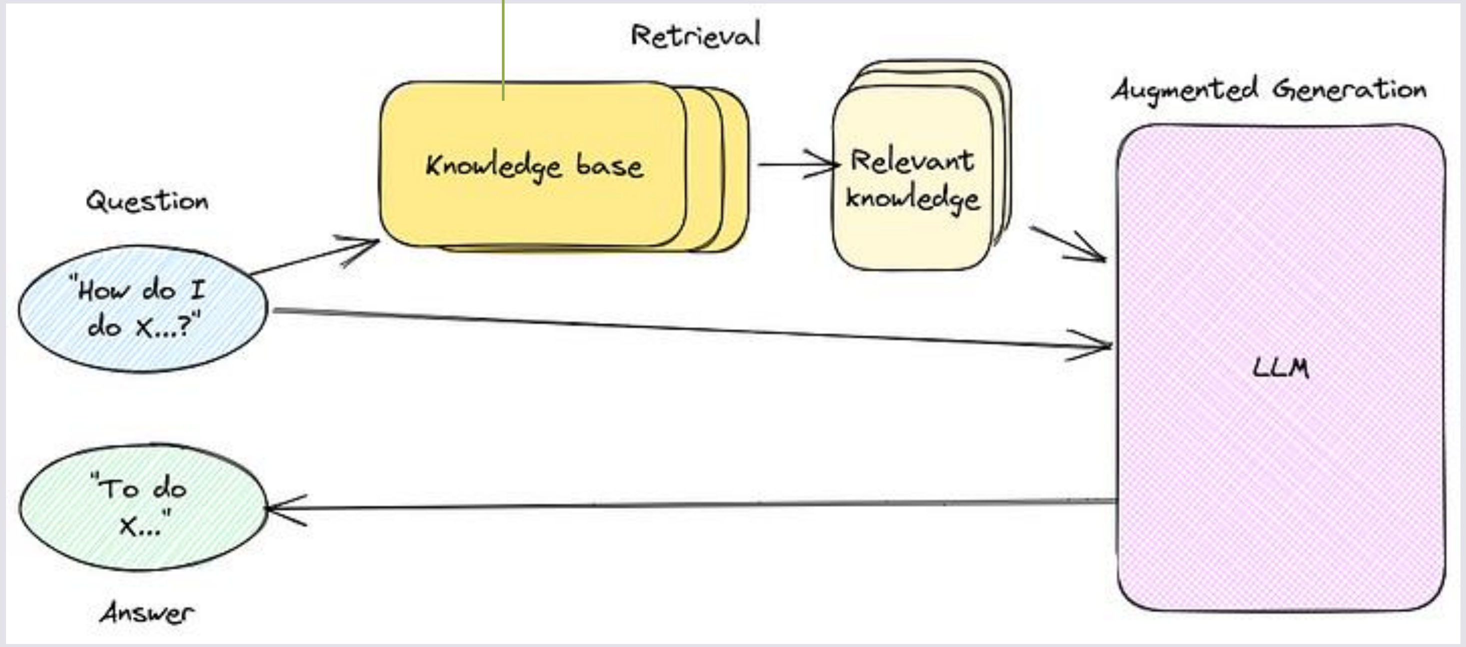
These combined factors likely contributed to the smaller crowd at what was expected to be a highly attended event (Setlist.fm) (More Than Belgrade) (JamBase).



Source: link

RAGQL

Abstract određenog naučnog rada i njemu sličnih radova (RDF baze znanja i SPARQL upite)



Enhancing Scientific Paper Comprehension through Retrieval-Augmented Generation (RAG) technique and SPARQL, Marija Đokić Petrović, Jasmina Đorđević, International conference “Stylistics and AI”, Belgrade, Jun 2024

NLP benefiti

Objektivnija i tačnija analiza

Analiza velikih razmera

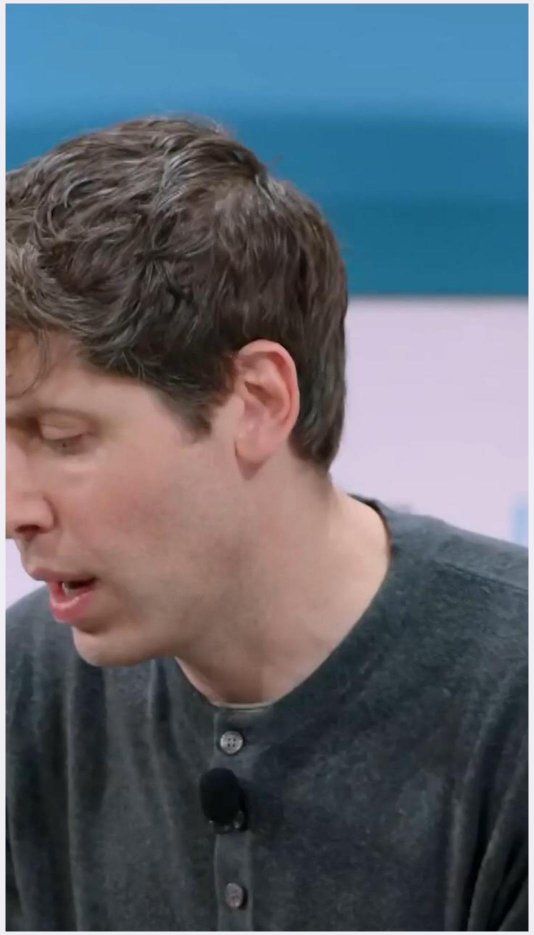
Jednostavniji procesi i manji troškovi

Bolje zadovoljstvo korisnika u poslovnom okruženju

NLP izazovi

Pravopisne greške	*cake/take* *suziti/suditi*
Jezičke razlike	*I’m going to work tomorrow morning.* *Idem sutra na posao.*
Reči sa višestrukim značenjem	*bark ->* *kosa*
Sarkazam	*If I had a dollar for every smart thing you say, I’d be poor. / Neću da te vređam nego se radujem tvojoj kosi koja je pobegla od nepristojne glave.*
Treniranje podataka	*Više podataka za treniranje -> bolji rezultati*

Zaključak



Lično, duboko sumnjam u ovu viziju budućnosti u kojoj su svi super bliski sa AI prijateljima, a ne više sa svojim ljudskim prijateljima.

- Sam Altman, CEO OpenAI -

Source: link

Grac

Source: *link*

Srbi u Gracu

- Nikola Tesla
- Ivo Andrić
- Milena Pavlović Barili
- Karađorđe

Source: link

Ivo Andrić (Travnik 1892, Belgrad 1975),
Schriftsteller und Literaturnobelpreisträger,
der am 13. Juni 1924 an der
Karl-Franzens-Universität Graz promovierte.

www.marijadjokicpetrovic.com
office@marijadjokicpetrovic.com

Source: link