Inteligencja obliczeniowa - Eksploracja tekstu

Grzegorz Madejski

Definicja

Eksploracja tekstu

Eksploracja tekstu (ang. text mining) to zbiór technik służących do wydobywania informacji z tekstu i ich obróbki. Przez tekst możemy rozumieć: posty, tweety, blogi, artykuły, emaile, książki, recenzje (i wiele innych, zarówno w Internecie, jak i offline).

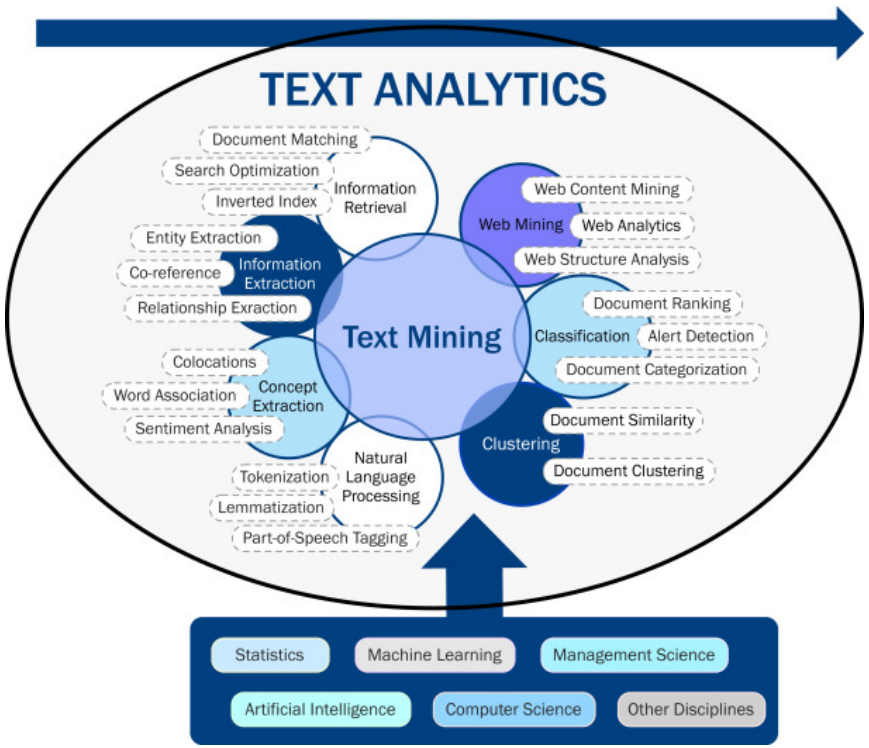
Eksploracja tekstu i inne dyscypliny

Eksploracja tekstu ma związek z dziedzinami pokrewnymi:

- *NLP* (ang. Natural Language Processing), przetwarzanie języka naturalnego. Techniki służące do zamiany nieustrukturyzowanego tekstu na struktury wygodne do dalszej analizy (wektory, tabele, itp).
- *Sztuczna inteligencja*. Ustruktyryzowane teksty można poddawać przeróżnym algorytmom uczenia maszynowego czy sztucznej inteligencji (np. klasyfikacja).
- *Inteligencja obliczeniowa*. Związek z IO istnieje, gdy ustrukturyzowane teksty poddane technikom wykorzystującym logikę rozmytą, aproksymacje i algorytmy inspirowane biologicznie, sieci neuronowe.

Przegląd metod

Inne spojrzenie na eksplorację tekstu (źródło: https://www.sciencedirect.com/topics/mathematics/text-analytics)



Etapy i metody

Metody przetwarzania tekstu		
Etap	Opis	Przykład
Wyszukanie tekstów (Information Retrieval, IR)	Wyszukujemy nieprzetworzone dokumenty wg jakiegoś kryterium lub zapytania	Wyszukanie tweetów wg jakiegoś #hasztaga
Wydobycie informacji (Information Extraction, ER) (Preprocessing)	Przetwarzamy dokumenty wg potrzeb; struktura przetworzonego dokumentu musi umożliwiać pracę nad nim	Rozbicie tekstu na zbiór słów (bag of words); analiza gramatyczna zdań (NLP)
Przetwarzanie i analiza informacji	Używamy algorytmów do przetwarzania informacji	Klasyfikacja/kategoryzacja tekstów; grupowanie tekstów; podsumowanie

Zastosowanie: ChatBot

Chatboty znajdują się w wielu serwisach i pomagają ich użytkownikom w rozwiązywaniu problemów. Chatboty musza przetworzyć tekst wprowadzony przez użytkownika, zrozumieć jego znaczenie i skomponować sensowną odpowiedź.

Zastosowanie: Watson

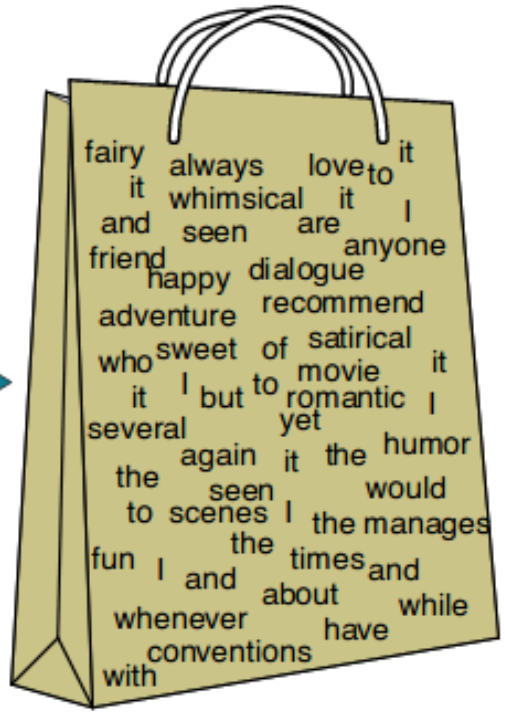
IBM Watson gra w teleturniej Jeopardy:
`https://www.youtube.com/watch?v=YgYSv2KSyWg`
To system do odpowiadania na pytania zadane w jezyku naturalnym (angielskim). Wykorzystuje on połączenie algorytmów do przetwarzania języka naturalnego, wyszukiwania informacji, reprezentacji wiedzy, wnioskowania automatycznego i uczenia maszynowego [Wikipedia].

Bag of Words

- Wszelkie teksty (np. posty, artykuły, itd.) nazywamy *dokumentami*.
- Dokumenty można rozbić na *słowa* (words).
- Słowa czasem określane są jako (ang.) *terms*, choć to określenie może mieć nieco inne znaczenie.
- Model "Bag of Words" rozbija dokumenty na słowa.
- Następnie słowa są zliczane w formie wektora z liczbą występowania słów.
- Jest to jedno z najlatwiejszych podejść do przetwarzania dokumentów (ignoruje kolejność słów, strukturę zdań).

Bag of Words

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

Etapy tworzenia

- *Tokenizacja* - rozbijamy dokument na słowa.
- *Stop words* - usuwanie słów, które nie mają znaczenia w naszej analizie np. przyimki, zaimki.
- *Lematyzacja* - usuwanie odmian słów (np. programował, programowaliśmy $\longrightarrow$ programować)
- *Stemming* - jako alternatywa dla lematyzacji, zachowywanie tylko rdzenia słowa (np. programował, programowaliśmy $\longrightarrow$ program)
- Zapisanie dokumentów jako wektory.

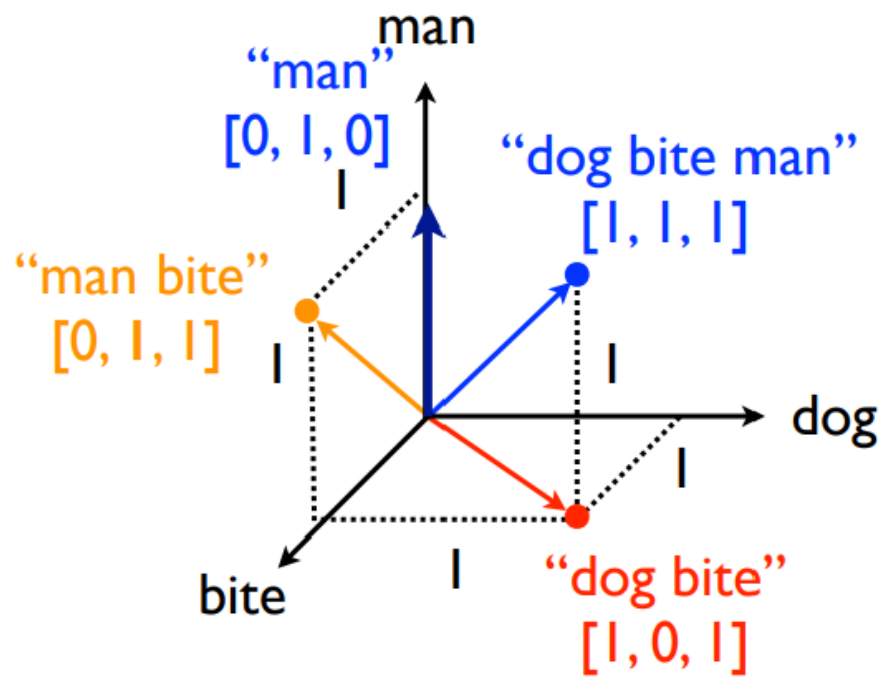
Narzędzia

Najbardziej popularną i rozwiniętą paczką pythonową do przetwarzania tekstów jest *NLTK*, natural language toolkit. Istnieje wiele dobrych samouczków w internecie. Na oficjalnej stronie można znaleźć też zestaw tutoriali w postaci e-książki: `https://www.nltk.org/book/`. Oficjalna strona to `https://www.nltk.org/`. Inne paczki warte sprawdzenia to *spacy* (`https://spacy.io/`) czy *textblob* (`https://textblob.readthedocs.io/en/dev/`).

Dokumenty jako wektory

Jeśli dokumenty są reprezentowane jako wektory z liczbą słów, to można je zestawić w macierzy *DTM* (ang. Document-Term Matrix) oraz zwizualizować w przestrzeni *n*-wymiarowej (gdzie *n* to liczba rozpatrywanych słów).

	dog	*man*	*bite*
doc_1	1	1	1
doc_2	1	0	1
doc_3	0	1	1
doc_4	0	1	0



TF vs IDF vs TFIDF

Gdy mamy bazę dokumentów zapisanych jako wektory w macierzy DTM (document-term matrix), można użyć kilku miar/wag na zliczanie słów:

- Zwyczajnie, jako liczba słów w danym dokumencie.
- *TF, Term Frequency* - występowanie słowa w danym dokumencie, jest podzielone przez liczbę wszystkich słów z danego dokumentu
- *IDF, Inverse Document Frequency* - dla każdego słowa w dokumencie: bierzemy liczbę wszystkich dokumentów, dzielimy przez liczbę dokumentów z tym słowem i całość logarytmujemy (dyskryminowanie zbyt popularnych słów)
- *TFIDF* - pomnożenie dla każdej liczby w tabeli: miara TF razy IDF

Zadanie

Zadanie

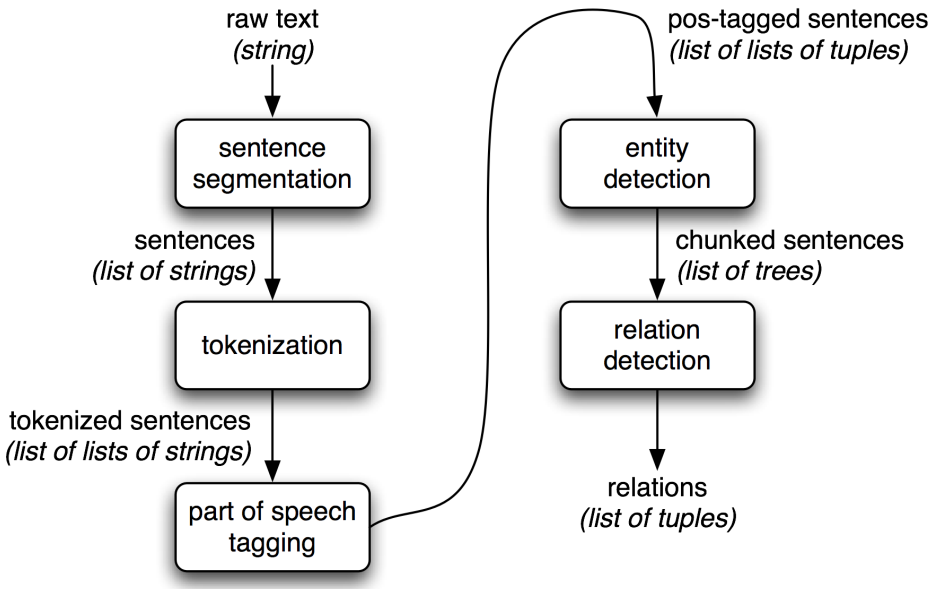
Dana jest macierz DTM:

	Prezent	Mikołaj	Choinka
Dokument 1	4	2	0
Dokument 2	1	1	2
Dokument 3	2	0	1

Podaj macierz TF i TFIDF.

Analizowanie struktury tekstu

Model "Bag of words" rozbijał tekst na słowa, zliczał je, ignorował zdania i strukturę tekstu. Są bardziej zaawansowane techniki NLP, które analizują strukturę tekstu.



Rozpoznawanie części mowy

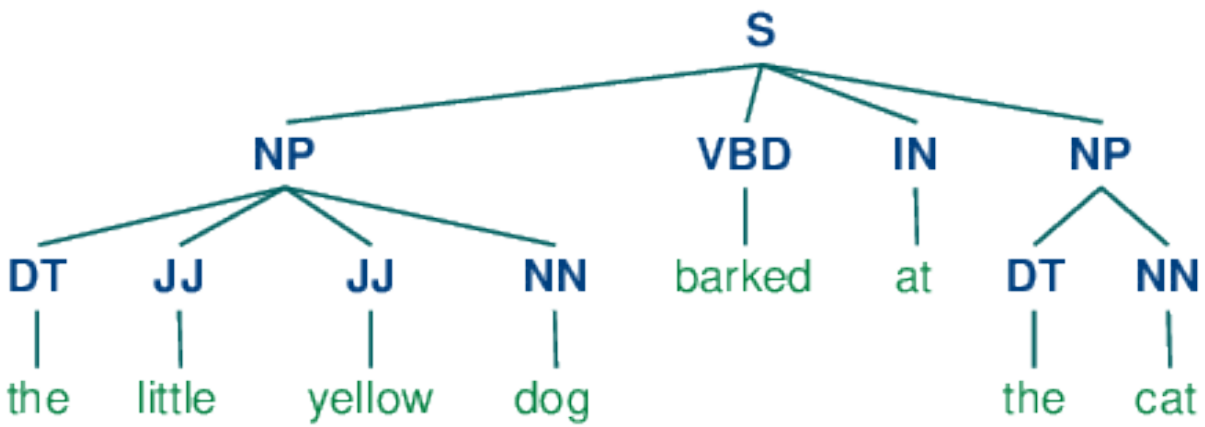
- Oznaczanie części mowy (ang. part of speech tagging, pos tagging) to proces, w którym etykietuje się każde słowo skórtem dla nazwy części mowy.
- W NLTK realizujemy to komendą *nltk.pos_tag(tekst)*.
- Przykłady części mowy: CC (conjunction, spójnik), RB (adverb, przysłówek), itd. Lista w komendzie *nltk.help.upenn_tagset()*

```
>>> text = word_tokenize("And now for something completely different"
>>> nltk.pos_tag(text)
[('And', 'CC'), ('now', 'RB'), ('for', 'IN'), ('something', 'NN'),
('completely', 'RB'), ('different', 'JJ')]
```

https://www.nltk.org/book/ch05.html

Parsowanie zdań

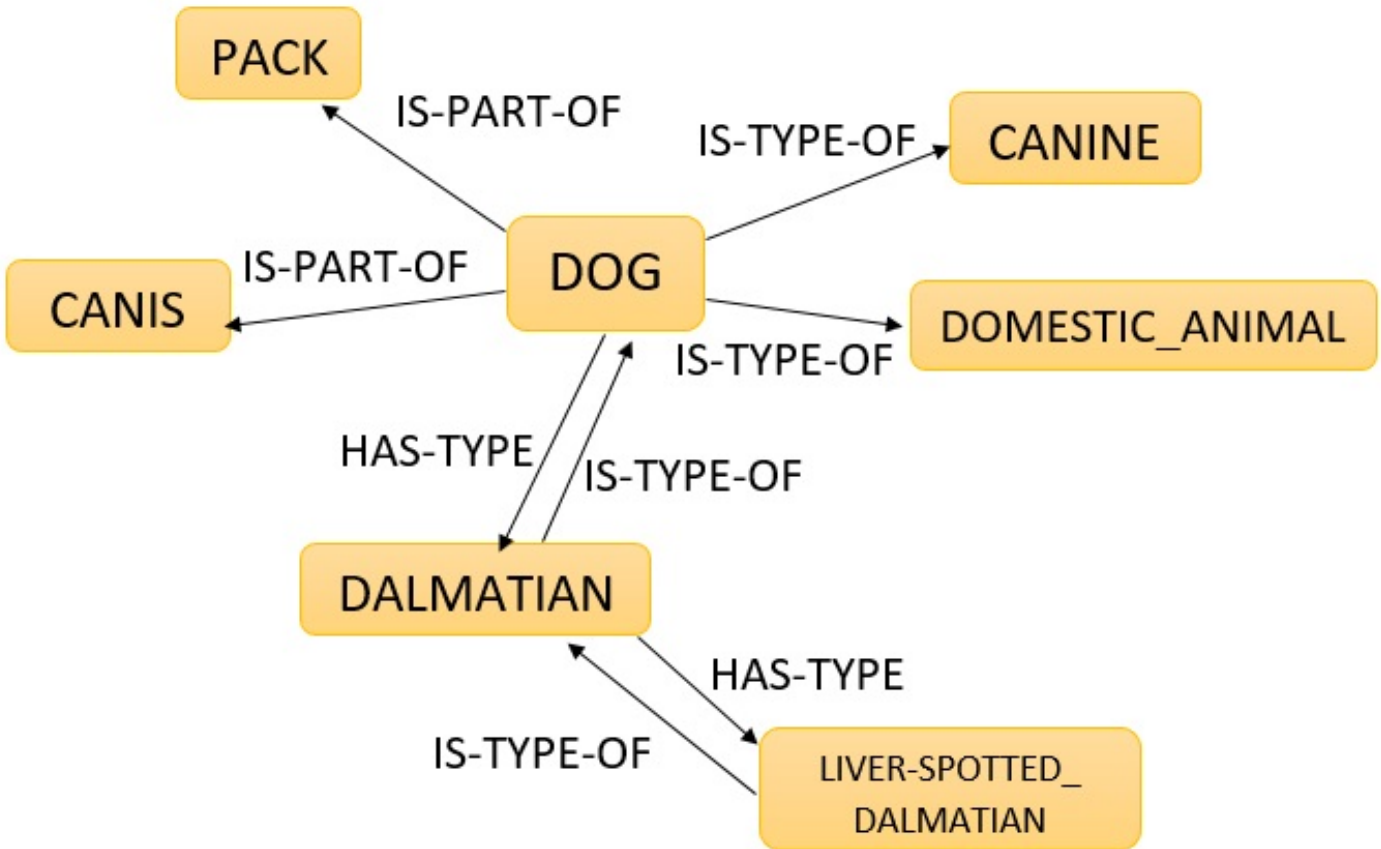
Bardziej złożonym procesem, jest parsowanie zdań. Po przetworzeniu zdanie zamieniane jest w drzewo.



WordNet

- *WordNet* to baza leksykalno-semantyczna tzn. opisuje słowa wraz z ich znaczeniami.
- Pierwsza wersja powstała w latach 1980 na uniwersytecie Princeton, USA. Była rozwijana na przestrzeni lat.
- WordNet wykorzystywana jest w wielu innych bazach wiedzy i systemach. Ich liczba stale rośnie.
- Jednostką bazową bazy jest *synset* = zestaw słów o tym samym znaczeniu (synonimów).
- Synsety mogą być w różnych relacjach (o nich później)
- Synsety połączone relacjami tworzą sieć. Mówimy o takiej sieci: *sieć semantyczna* (ang. semantic network).

Wordnet


PACK
IS-PART-OF
IS-TYPE-OF
CANINE
CANIS
IS-PART-OF
DOG
DOMESTIC_ANIMAL
IS-TYPE-OF
HAS-TYPE
IS-TYPE-OF
DALMATIAN
HAS-TYPE
IS-TYPE-OF
LIVER-SPOTTED_
DALMATIAN

Terminy związane ze słowami

- *Sens wyrazu, Znaczenie*, (ang. *Word Sense, Sense*) - bywa, że jedno słowo ma wiele znaczeń. Ile znaczeń ma słowo "zamek"? O znaczeniu mówimy, że jest wartością semantyczną.
- *Leksem* (ang. *Lexeme*) - słowo wraz z konkretnym znaczeniem, i wszystkimi możliwymi odmianami. Leksem "czytać", "czytam", "przeczytasz", itd obejmuje grupę wyrazów i może być reprezentowany przez wyraz "czytać".
- *Forma słownikowa, Lemat, Lemma* (ang. *Lemma*) - nieodmieniona wersja wyrazu, wystepująca w słownikach. Lemat jest reprezentantem leksemu w słowniku.

Synsety

- *Synset* - zestaw synonimicznych leksemów reprezentowanych przez lematy (formy słownikowe).
- Przykładowo pies jest utożsamiany z synsetem ['dog', 'domestic dog', 'Canis familiaris']. Widzimy zestaw trzech lematów o tym samym znaczeniu (mają tę samą definicję).
- W Wordnecie każdy synset składa się m.in. z:
  - nazwy *name* - składającej się z wyrazu, części mowy, numeru znaczenia dla tego wyrazu
  - definicji *definition* - opis tekstowy znaczenia synsetu
  - cześci mowy *pos* - do wyboru jedna z pięciu: przymiotnik *a*, przymiotnik-satelita (poboczny) *s*, rzeczownik *n*, przysłówek *r*, czasownik *v*.
  - *lemmas*, wszystkie lematy/wyrazy synonimiczne wchodzące w skład synsetu

Synsety w Pythonie

```
from nltk.corpus import wordnet as wn
dog = wn.synset('dog.n.01')
print(dog.name())
print(dog.definition())
print(dog.pos())
print(dog.lemmas())
```

```
dog.n.01
a member of the genus Canis (probably descended from the common wolf) that h
as been domesticated by man since prehistoric times; occurs in many breeds
n
[Lemma('dog.n.01.dog'), Lemma('dog.n.01.domestic_dog'), Lemma('dog.n.01.Can
s_familiaris')]
```

Relacje między synsetami

- *Synonimia* - synonimy to wyrazy równoznaczne lub bliskoznaczne np. "pieniądze", "kasa", "hajs". Lematy połączone relacją synonimii są łączone w jedną jednostkę: synset.
- *Antonimia* - antonimy to wyrazy o przeciwnym znaczeniu np. "suchy" i "mokry".
- *Hiponimia* - hiponim to wyraz podrzędny znaczeniowo np. "krowa" jest hiponimem słowa "zwierzę".
- *Hiperonimia* - hiperonim to wyraz nadrzędny znaczeniowo np. "sztuciec" jest hiperonimem słowa "widelec".
- *Meronimia* - meronim to wyraz, który znaczeniowo wyraża część innego słowa np. "koło" jest meronimem słowa "samochód".
- *Holonimia* - holonim to wyraz, który znaczeniowo wyraża większą całość np. "samochód" jest holonimem "koła".

Relacje między wyrazami

Uwaga! Relacje te właściwie są pomiędzy synsetami (zestaw słów ze wspólnym znaczeniem), a nie samymi słowami.

"Zamek" (budowla) i "Pałac" (budowla) to synonimy.

"Zamek" (zapięcie ubrania) i "Pałac" (budowla) to nie są synonimy.

Relacje między wyrazami

W jakich relacjach są wyrazy?

- "Mickiewicz" jest ... dla "autor"
- "wydział" jest ... dla "profesor"
- "tlen" jest ... dla "woda"
- "pilot" jest ... dla "załoga"
- "kompozytor" jest ... dla "Bach"

Relacje między wyrazami

- Więcej o relacjach: `https://web.stanford.edu/~jurafsky/slp3/old_oct19/C.pdf` (str. 6)
- Relacje w Pythonie: `https://www.nltk.org/api/nltk.corpus.reader.wordnet.html#module-nltk.corpus.reader.wordnet`
- Wypisane wszystkie możliwe relacje.

antonyms, hypernyms, instance_hypernyms, hyponyms, instance_hyponyms, member_holonyms, substance_holonyms, part_holonyms, member_meronyms, substance_meronyms, part_meronyms, topic_domains, region_domains, usage_domains, attributes, derivationally_related_forms, entailments, causes, also_sees, verb_groups, similar_tos, pertainyms

Relacje w Pythonie

```
from nltk.corpus import wordnet as wn
dog = wn.synset('dog.n.01')
print(dog.hypernyms())
print(dog.hyponyms())
print(dog.member_holonyms())
dal = wn.synset('dalmatian.n.02')
print(dal.hypernyms())
print(dal.hyponyms())
```

```
[Synset('canine.n.02'), Synset('domestic_animal.n.01')]
[Synset('basenji.n.01'), Synset('corgi.n.01'), Synset('cur.n.01'), Synset('dalmatian
.02'), Synset('great_pyrenees.n.01'), Synset('griffon.n.02'), Synset('hunting_dog.n.
'), Synset('lapdog.n.01'), Synset('leonberg.n.01'), Synset('mexican_hairless.n.01'),
ynset('newfoundland.n.01'), Synset('pooch.n.01'), Synset('poodle.n.01'), Synset('pug
.01'), Synset('puppy.n.01'), Synset('spitz.n.01'), Synset('toy_dog.n.01'), Synset('w
king_dog.n.01')]
[Synset('canis.n.01'), Synset('pack.n.06')]
[Synset('dog.n.01')]
[Synset('liver-spotted_dalmatian.n.01')]
```

Wordnet - Przydatne linki

Przydatne linki:

- Oficjalna strona projektu: `https://wordnet.princeton.edu/`
- Dokumentacja WordNet w NLTK/Python: `https://www.nltk.org/api/nltk.corpus.reader.wordnet.html`
- Przykłady Wordnet, wersja krótka dokumentacji: `https://www.nltk.org/howto/wordnet.html`
- WordNet online: `http://wordnetweb.princeton.edu/perl/webwn`
- Jakiś małe samouczki: `https://www.geeksforgeeks.org/nlp-synsets-for-a-word-in-wordnet/`, `https://www.geeksforgeeks.org/get-synonymsantonyms-nltk-wordnet-python/?ref=lbp`

Podobieństwo dokumentów

Jeśli dokumenty są reprezentowane jako wektory, to są tym bardziej podobne do siebie im mniejszy kąt między nimi. Im mniejszy kąt, tym większy cosinus kąta (max. 1). Powszechnie stosowaną miarą podobieństwa dokumentów jest miara cosinusowa (cosine similarity).

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Uwaga: można porównywać podbieństwo istniejących dokumentów, lub dokumentów i zapytań (query). Zapytanie to sztucznie stworzony dokument (wektor) np. "dog" lub "man dog" do poprzedniego przykładu.

Zastosowanie? Systemy rekomendujące!

W jakimś serwisie: czytamy artykuły, przeglądamy produkty, wyszukujemy informacji. Serwisy na podstawie naszej aktywności podpowiadają, co nam się może spodobać. Szukamy podobnych dokumentów (cosine similarity?).
Przeczytałem artykuł "Prawdy i mity na temat picia wody".
Wyskoczyło:

Zadanie

Zadanie

Dana jest macierz DTM:

	Prezent	Mikołaj	Choinka
Dokument 1	4	2	0
Dokument 2	1	1	2
Dokument 3	2	0	1

Podaj macierz TF i TFIDF. Korzystając z macierzy TF oblicz podobieństwo dokumentów 1 i 2, oraz 1 i 3.

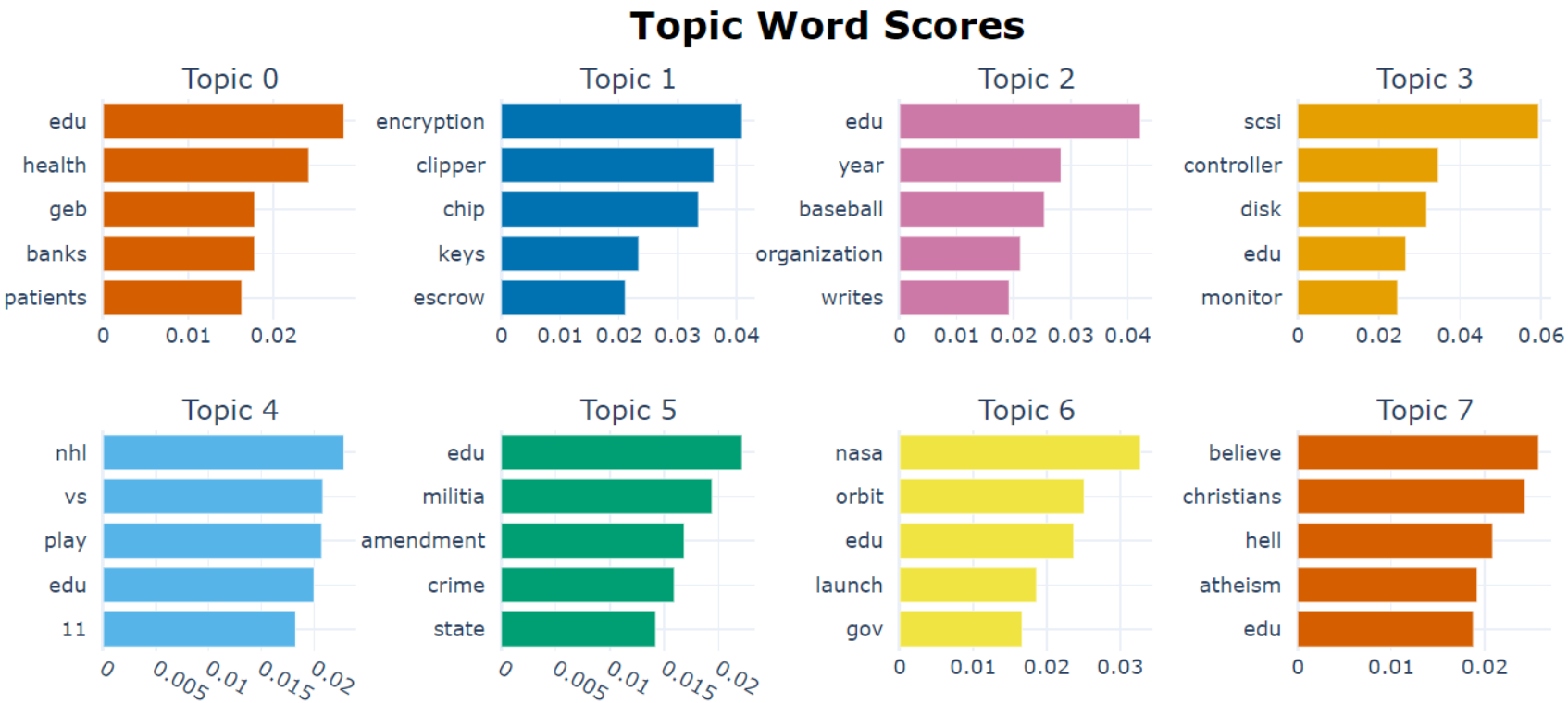
Modelowanie tematów - Definicja

Konceptem podobnym do podobieństwa dokumentów jest modelowanie tematów.

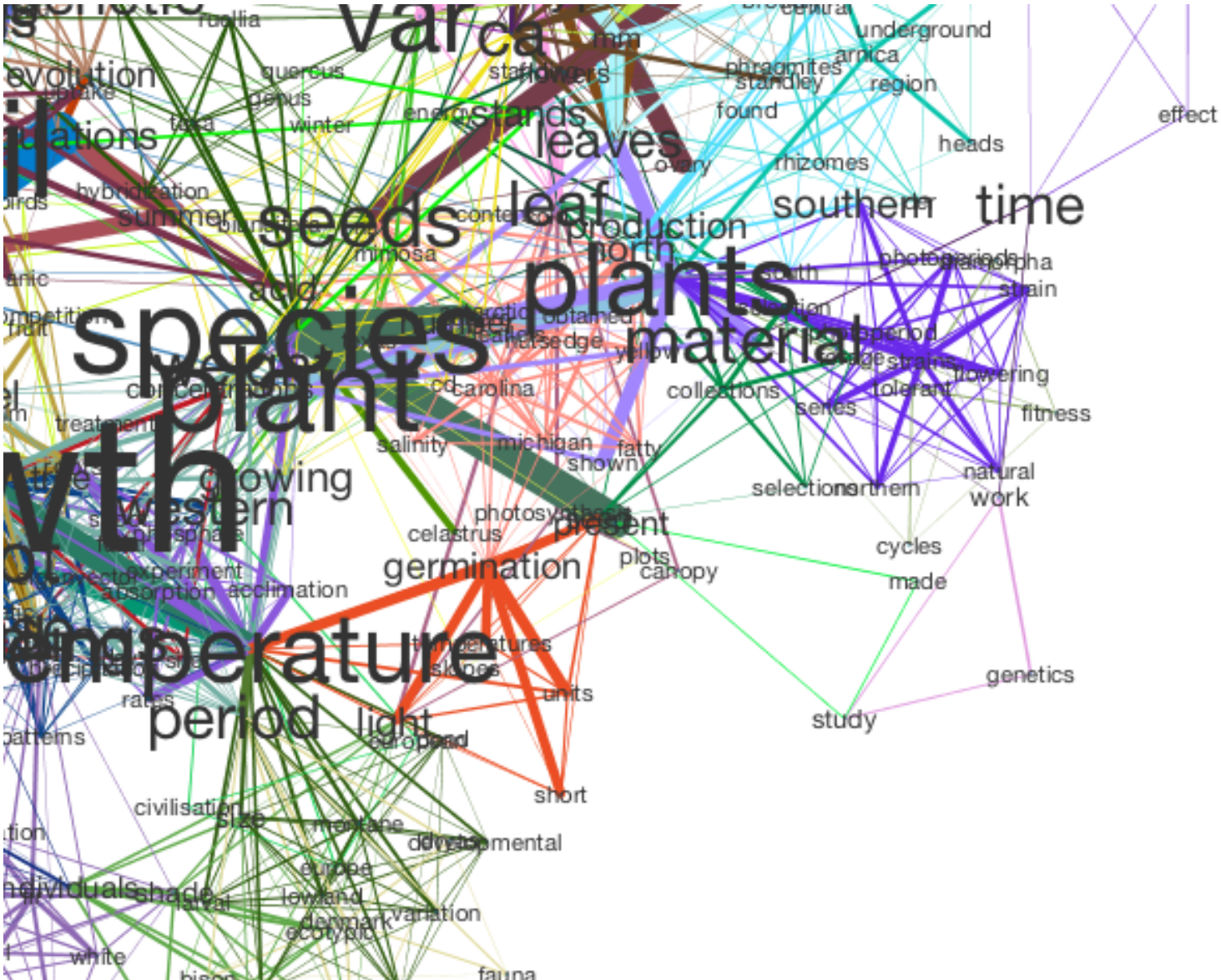
Modelowanie tematów

Modelowanie tematów (ang. topic modelling, topic analysis), to techniki dzielenia dokumentów na grupy/klastry o podobnej tematyce. Grupujemy dokumenty o podobnych zestawach słów-kluczy. Każda grupa ma słowa w innych proporcjach. Sam temat należy rozumieć abstrakcyjnie: nie jako napis (np. "sztuczna inteligencja"), ale zestaw najczęstszych słów występujących w klastrze (np. ["ai", "sztuczna inteligencja", "chatgpt", "robot", ...])

Modelowanie tematów - przykład



Modelowanie tematów - przykład



Modelowanie tematów - przykład

- Modelowanie tematów można wykonać różnymi bibliotekami pythonowymi: NLTK, gensim, scipy.
- Podobnie jak podobieństwo dokumentów - ma zastosowanie choćby w systemach zarządzających lub porządkujących treści, rekomendujących treści.

Definicja

Analiza opinii

Analiza opinii (wydźwięku), ang. sentiment analysis, opinion mining, to zbiór technik służących do identyfikowania, wydobywania i mierzenia emocji, uczuć, subiektywnego nastawienia z tekstu.

Jak działa?

- Podstawowy sposób: weź tekst, przeanalizuj słowa w nim wystepujące. Czy są pozytywne, negatywne czy neutralne (emocjonalnie)? Tekst oznaczamy odpowiednimi poziomami negatywności, pozytywności, neutralności.
- Bardziej zaawansowane techniki mogą otagować tekst wg kilku emocji. Można mierzyć np. poziom szczęścia, strachu, zaskoczenia, smutku, złości.

Narzędzie: NLTK-Vader

- Narzędzie ocenia teksty wg negatywności, pozytywności, neutralności, patrząc na ładunek emocjonalny każdego ze słów w tekście.
- Kod: `https://www.nltk.org/_modules/nltk/sentiment/vader.html`
- Narzędzie bierze pod uwagę wzmacniacze (np. "unbelievably"), idiomy ("the shit"), znaki przestankowe (np. "!!!"), emotki (np. ":-(").
- Narzędzie w istotny sposób korzysta z leksykony słów niosących ładunek emocjonalny. Dostępny np. tutaj: `https://www.kaggle.com/datasets/nltkdata/vader-lexicon`

Narzędzie: Text2Emotion

- Narzędzie ocenia teksty wg poziomów kilku emocji: angry, fear, happy, sad, surprise.
- Oficjalna strona `https://pypi.org/project/text2emotion/`
- Repozytorium `https://github.com/absurd-thought/text2emotion-library`
- Narzędzie wykorzystuje słowa, zwroty czy emotki otagowane jedną z tych 5 emocji.

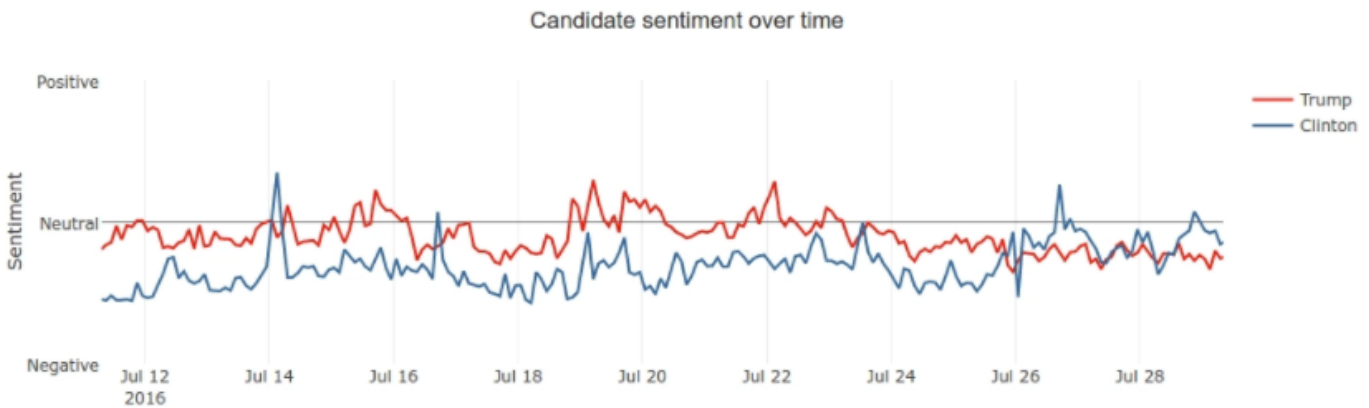
```
text = "Day was pretty amazing😃😃"
te.get_emotion(text)

#Output
{'Angry': 0.0, 'Fear': 0.0, 'Happy': 0.8, 'Sad': 0.0, 'Surprise': 0.2}
```

towardsdatascience.com/text2emotion-python-package-to-detect-emotions-from-textual-data-b2e7b7ce1153

Trendy w opiniach, sondaże

Z bazy tweetów, postów, opinii - wyciągamy informacje o tym, czy dany obiekt się ludziom podoba. Analizujemy jakie emocje wzbudza dany temat.
Wybory w USA (źródło: MonkeyLearn.com):



Finanse

Opinie w social mediach vs kurs akcji firmy BAESystems (źródło: DerwentCapital / BaeSystems.com):