3 Matematyka na słowach (wektory TD-IDF)

Tematyka rozdziału

- Zliczanie słów i określanie *częstości terminów* w celu analizowania znaczenia
- Przewidywanie prawdopodobieństwa występowania słowa za pomocą *prawa Zipfa*
- Co to jest wektorowa reprezentacja słów i jak zacząć jej używać
- Znajdowanie właściwych dokumentów w korpusie za pomocą *odwróconej częstości w dokumentach* (*inverse document frequencies*, IDF)
- Przewidywanie podobieństwa par dokumentów za pomocą *podobieństwa cosinusowego* i *Okapi MB25*

Po zebraniu i policzeniu słów (tokenów) oraz pogrupowaniu ich na rdzenie i lematy nadszedł czas na zrobienie z nimi czegoś ciekawego. Wykrywanie słów jest użyteczne w prostych zadaniach, jak statystyka używania słów czy wyszukiwanie na podstawie słów kluczowych. Ale chcielibyście też wiedzieć, które słowa w określonym dokumencie

i w korpusie jako całości są ważne. Wtedy możecie użyć tej wartości „ważności" do znalezienia właściwych dokumentów w korpusie na podstawie znaczenia słów kluczowych w każdym dokumencie.

To sprawi, że wykrywacz spamu będzie mniej podatny na potknięcie się na jednym przekleństwie lub bliskich spamu słowach w waszym emailu. I chcielibyście też zmierzyć, na ile pozytywny i prospołeczny jest tweet, gdy mamy szeroki zakres słów z różnymi stopniami wyniku „pozytywności" i różnymi etykietami. Jeśli macie pojęcie o częstości, z jaką te słowa pojawiają się w dokumencie *względem* reszty dokumentów, możecie to wykorzystać do dalszego udoskonalenia „pozytywności" dokumentu. W tym rozdziale poznacie bardziej zróżnicowane, mniej binarne miary słów i ich użycia w dokumencie. To podejście od dziesięcioleci stało się filarem wydobywania cech z języka naturalnego dla komercyjnych wyszukiwarek i filtrów spamu.

Kolejnym krokiem waszej przygody jest przekształcenie słów z rozdziału 2 w liczby rzeczywiste zamiast liczb całkowitych reprezentujących liczniki słów lub binarne „wektory bitowe", które wykrywają obecność lub nieobecność określonych słów. Mając reprezentację słów w ciągłej przestrzeni, możecie działać na ich reprezentacji za pomocą znacznie ciekawszej matematyki. Waszym celem jest znalezienie numerycznej reprezentacji słów, która w jakiś sposób uchwyci ważność lub zawartość informacyjną reprezentowanych słów. Będziecie musieli poczekać do rozdziału 4, aby zobaczyć, jak przekształcić tę zawartość informacyjną w liczby reprezentujące *znaczenie* słów.

W tym rozdziale przyglądamy się trzem coraz mocniejszym sposobom reprezentowania słów i ich wagi w dokumencie. Są to:

- *wektory BoW* – wektory liczników częstości słów;
- *wektory pojemników na n-gramy* – liczniki par słów (bigramów), trójek (trigramów) i tak dalej;
- *wektory TF-IDF* – wektory wartości liczbowych przypisywanych słowom w sposób, który lepiej reprezentuje ich znaczenie.

WAŻNE TF-IDF to skrót od *częstość słowa razy odwrotność częstości w dokumentach* (***t**erm **f**requency times **i**nverse **d**ocument **f**requency*). Częstości słów jest określona liczbą wystąpień każdego słowa w dokumencie, które poznaliście w poprzednich rozdziałach. Odwrotna częstość w dokumentach oznacza, że dzielicie każdą z tych liczb słów przez liczbę dokumentów, w których słowo występuje.

Każdą z tych technik można zastosować oddzielnie lub jako część potoku NLP. Są to modele statystyczne, gdyż opierają się na *częstości*. W dalszej części książki poznacie różne sposoby głębszego zajrzenia w związki między słowami, w ich wzorce i nieliniowości.

Ale te „płytkie" systemy NLP są mocne i użyteczne w wielu praktycznych zastosowaniach, jak filtrowanie spamu i analiza wydźwięku.

3.1. Wektor BoW

W poprzednim rozdziale utworzyliście pierwszy model tekstu w przestrzeni wektorowej. Zastosowaliście kodowanie 1 z *n* dla każdego słowa, a następnie połączyliście wszystkie

te wektory za pomocą binarnego działania OR, aby utworzyć wektorową reprezentację tekstu. I ten binarny wektor BoW tworzy wielki indeks do wyszukiwania dokumentów, gdy pobierzemy go do struktury danych jak Pandas DataFrame.

Potem przyjrzeliście się jeszcze bardziej użytecznej reprezentacji wektorowej, która zlicza liczbę wystąpień, czyli częstość, każdego słowa w danym tekście. Jako pierwsze przybliżenie zakładacie, że im więcej razy występuje dane słowo, tym więcej znaczenia musi ono wnosić do dokumentu. Dokument, który odwołuje się często do „wings" i „rudder" (skrzydła i ster) może być bardziej właściwy dla problemu związanego z odrzutowcami czy podróżami powietrznymi niż powiedzmy dokument, w którym często jest mowa o „cats" i „gravity" (kotach i grawitacji). Lub jeśli klasyfikujecie słowa wyrażające pozytywne emocje – słowa jak „good", „best", „joy" i „fantastic" (dobry, najlepszy, radość, fantastyczny) – dokument zawierający więcej tych słów zapewne będzie wyrażał pozytywne nastawienie. Możecie sobie jednak wyobrazić, jak algorytm oparty na tych prostych zasadach może być mylący lub zagubiony.

Popatrzmy na przykład, w którym liczenie wystąpień słowa jest użyteczne:

```
>>> from nltk.tokenize import TreebankWordTokenizer
>>> sentence = """The faster Harry got to the store, the faster Harry,
...     the faster, would get home."""
>>> tokenizer = TreebankWordTokenizer()
>>> tokens = tokenizer.tokenize(sentence.lower())
>>> tokens
['the',
 'faster',
 'harry',
 'got',
 'to',
 'the',
 'store',
 ',',
 'the',
 'faster',
 'harry',
 ',',
 'the',
 'faster',
 ',',
 'would',
 'get',
 'home',
 '.']
```

Mając swoją prostą listę, chcecie pobrać z dokumentu unikatowe słowa i ich liczniki. Słownik Pythona służy dobrze do tego celu, a ponieważ chcecie także liczyć słowa, możecie podobnie jak w poprzednich rozdziałach użyć `Counter`:

```
>>> from collections import Counter
>>> bag_of_words = Counter(tokens)
>>> bag_of_words
Counter({'the': 4,
         'faster': 3,
```

```
'harry': 2,
'got': 1,
'to': 1,
'store': 1,
',': 3,
'would': 1,
'get': 1,
'home': 1,
'.': 1})
```

Jak w każdym dobrym słowniku Pythona kolejność kluczy została wymieszana. Nowa kolejność jest zoptymalizowana pod kątem pamięci, aktualizacji i znajdowania danych, a nie ze względu na wyświetlanie. Zawartość informacyjna w kolejności słów z oryginalnego zdania została odrzucona.

UWAGA Obiekt `collections.Counter` to nieuporządkowana kolekcja, nazywana także wielozbiorem. Zależnie od waszej platformy i wersji Pythona, może okazać się, że Counter jest wyświetlany w całkiem sensownej kolejności, jak kolejność leksykalna lub według tokenów pojawiających się w zdaniu. Ale w standardowym `dict` Pythona nie możecie polegać na kolejności tokenów (kluczy w `Counter`).

Dla krótkich dokumentów jak ten nieuporządkowany wektor BoW nadal zawiera wiele informacji o oryginalnym sensie zdania. A informacje w tym wektorze BoW wystarczają do wykonania takich rzeczy jak wykrywanie spamu, wyznaczenie wydźwięku (pozytywnego nastawienia, szczęścia itp.), a nawet na wykrycie subtelnych intencji jak sarkazm. Może jest to wektor BoW, lecz jest pełen znaczenia i informacji. Wprowadźmy więc ranking tych słów – posortujmy w jakiejś kolejności, o której jest łatwiej myśleć. Obiekt `Counter` ma własnie do tego celu wygodną metodę, *most_common*:

```
>>> bag_of_words.most_common(4)
[('the', 4), (',', 3), ('faster', 3), ('harry', 2)]
```

Domyślnie most_common wypisuje wszyskie tokeny, od najczęstszego do występującego najrzadziej, ale tu ograniczyliście listę do czterech najważniejszych.

Konkretnie, liczba wystąpień słowa w danym dokumencie jest nazywana *częstością słowa* i często używa się skrótu TF (*term frequency*). W niektórych przykładach możecie zobaczyć licznik wystąpień słowa znormalizowany (podzielony) przez liczbę wszystkich słów w dokumencie[1].

Tak więc cztery główne pojęcia, czyli tokeny, to „The”, „,”, „harry” i „faster”. Ale słowo „the” oraz znak interpunkcji „,” nie niosą wiele informacji o zamyśle tego dokumentu. A te nieinformacyjne tokeny będą pojawiać się często podczas waszej szybkiej przygody. Więc na potrzeby tego przykładu zignorujecie je, razem z resztą stop listy standardowych angielskich słów i znaków przestankowych. Nie zawsze tak będzie, ale teraz pomaga w uproszczeniu przykładu. To pozostawia „harry” i „faste” wśród najczęstszych tokenów w waszym wektorze TF (BoW).

[1] Jednak znormalizowana częstość jest naprawdę prawdopodobieństwem, więc zapewne nie powinna być nazywana częstością.