

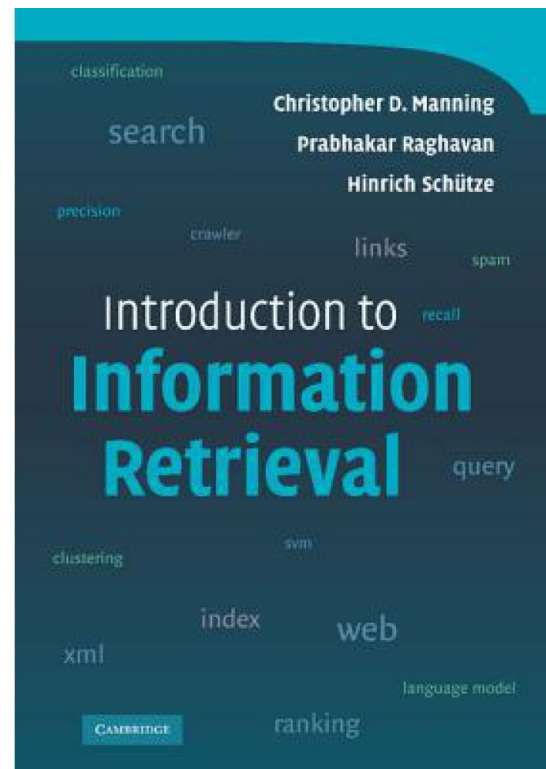
Text Mining

Andrzej Dudziec 24.11.2019

am.dudziec@gmail.com

Na podstawie:

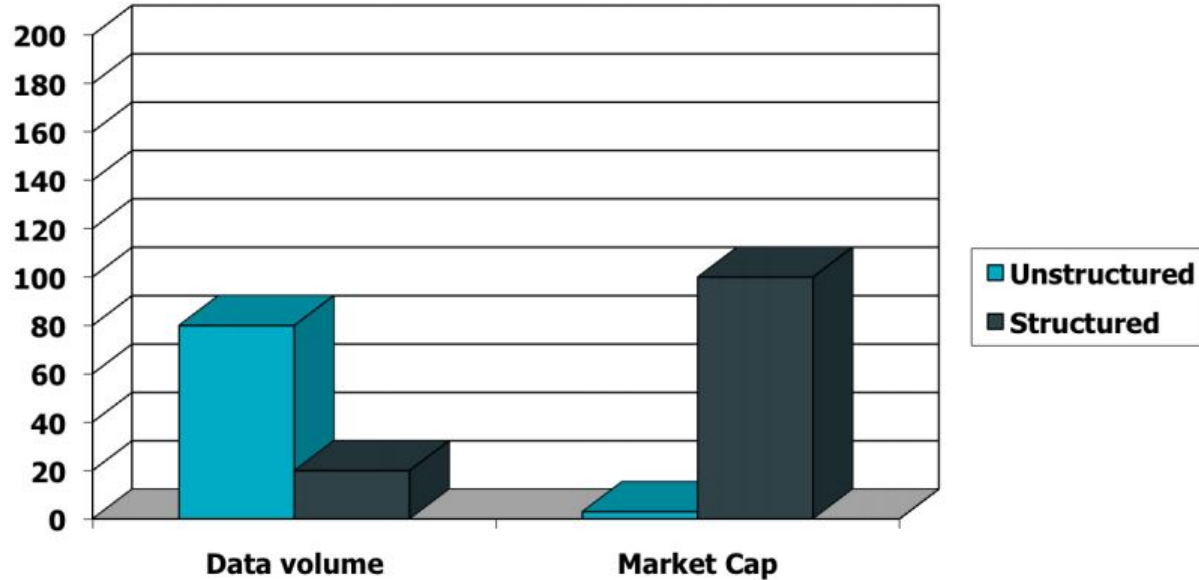
- Text Mining, Liliana Pięta, Rafał K. Wojdan
- Information Retrieval, Donald Kossmann
- Manning, Raghavan, Schütze: Introduction to Information Retrieval, Cambridge Press
<https://nlp.stanford.edu/IR-book/information-retrieval-book.html>



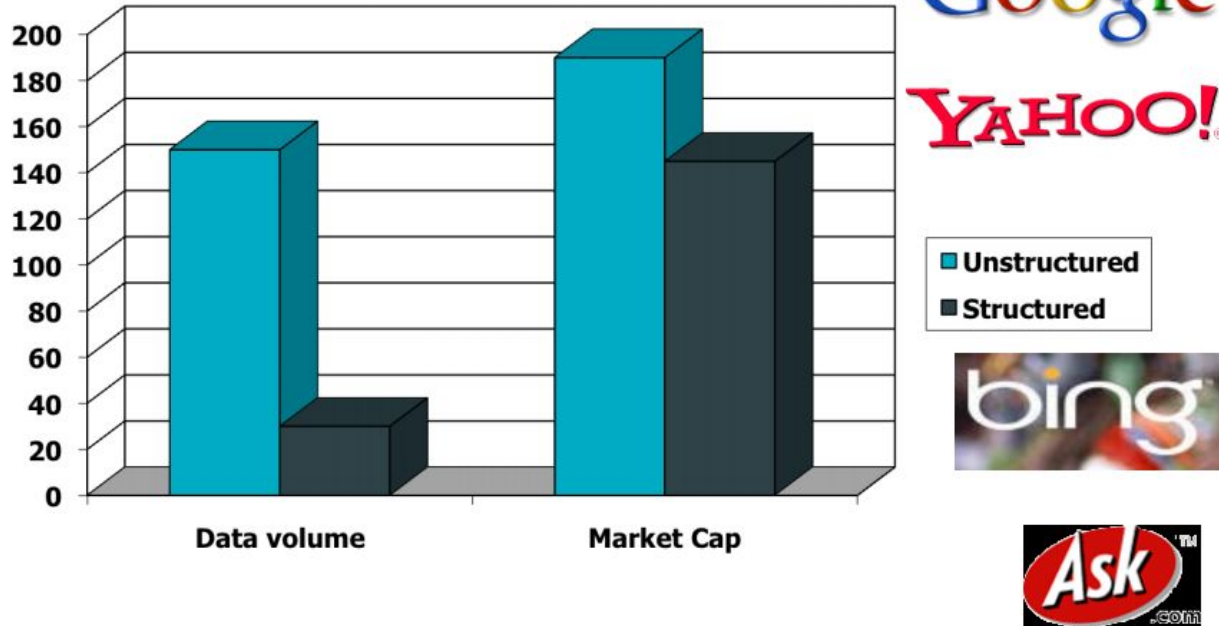
Dane, dane, dane...

- **Structured data** - bazy danych (tabele, kolumny), ustrukturyzowane dane numeryczne
- **Semi-structured data** - json, XML
- **Unstructured data** - tekst (strony, maile, tweety, książki itp.)

Unstructured vs Structured (1996)



Unstructured vs Structured (2009)



Zastosowania analizy tekstu

- Klasyfikacja: newsów, tweetów, stron itp.
- Kategoryzacja e-maili (np. wykrywanie spamu)
- Klastrowanie dokumentów lub stron internetowych
- Systemy rekomendacyjne
- Analiza sentymentu
- Podpowiedzi kolejnego słowa
- Predykcja churn'u
- I wiele innych :)

Unstructured data - wyszukiwanie binarne

Które sztuki Szekspira zawierają słowa **Brutus** AND **Caesar** but NOT **Calpurnia**?

Może 'grep'?

- wolne (dla dużych zbiorów danych)
- 'NOT **Calpurnia**' jest nietrywialne
- zaawansowane wyszukiwania niemożliwe
- brak jakości dopasowania

Incidence vectors

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

***Brutus AND Caesar BUT NOT
Calpurnia***

1 if play contains word, 0
otherwise

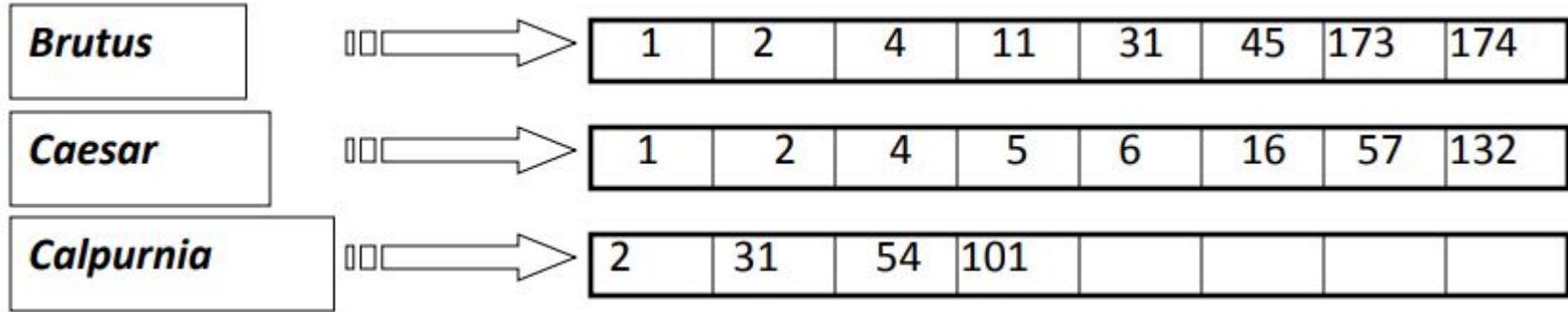
Duże zbiory

- Załóżmy, że mamy 1 milion dokumentów, każdy po ok 1000 słów
- Średnio 6 bitów/słowo daje ok 6GB danych
- Załóżmy, że mamy 500 unikalnych słów

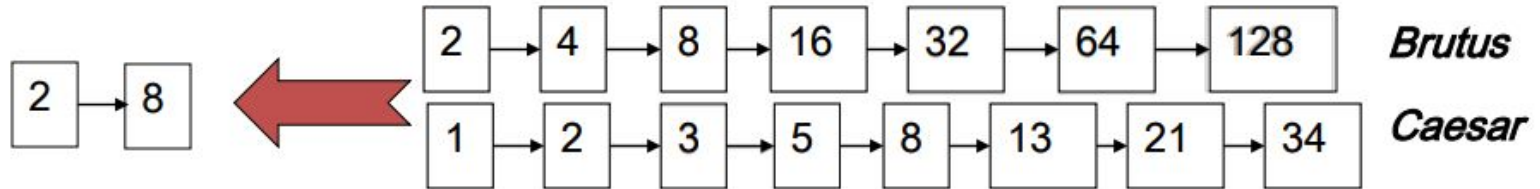
Taka macierz miałaby pół tryliona “zer” lub “jedynek”
(głównie “zera” - dlaczego?)

Lepszym rozwiązaniem jest przechowywać tylko “jedyнки”

Inverted index



Brutus AND Caesar



Wyszukiwanie binarne

- używa operatorów AND, OR, NOT
- traktuje dokumenty wyłącznie jako zbiór słów
- dokument pasuje do zapytania albo nie (nie ma opcji pośredniej)
- było podstawowym podejściem do wyszukiwania przez 3 dekady!
- wiele systemów wciąż z niego korzysta (np. Mac OS Spotlight)

Ograniczenia:

- trzeba dokładnie wiedzieć czego się szuka
- co z frazami? np. “Stanford University”
- lokalizacja w tekście: find **Gates** NEAR **Microsoft**
- potrzeba sortowania po trafności

Preprocessing

Trzeba sobie odpowiedzieć na kilka pytań:

- w jakim formacie są dokumenty? (pdf/word/excel/html/e-mail)
- w jakim są języku?
- jakim alfabetem są spisane?

Tokenizacja

W większości wypadków tokenem jest każde ze słów, które (po odpowiednim przetworzeniu) może stać się częścią indeksu

- Finland's capital -> Finland/Finlands/Finland's?
- Hewlett-Packard -> jeden token czy dwa?
- San Francisco -> jeden token czy dwa?
- Liczby: 3/20/91 B-52 55 B.C. Mar. 12, 1991

Tokenizacja - problemy językowe

- francuski: **L'ensemble** (jakie tokeny? czy jest tym samym co **un ensemble**?)
- niemiecki: **Lebensversicherungsgesellschaftsangestellter** (pracownik firmy ubezpieczeń na życie)
- chiński: 莎拉波娃现在居住在美国东南部的佛罗里达 (brak spacji)
- arabski: استقلت الجزائر في سنة 1962 بعد 132 عاما من الاحتلال الفرنسي.
← → ← → ← start

Stopwords

Nie ma sensu indeksować bardzo częstych słów, które niewiele wnoszą: a, an, the, to, from, do etc. Dlatego często nie uwzględnia się ich w indeksie

Ale mogą być potrzebne:

- wyszukiwanie fraz: “King of Denmark”
- Piosenki, cytaty etc.: “Let it be”, “To be or not to be”
- Relacje: “flights to London”

Normalizacja

- usuwamy kropki, przecinki itp
 - U.S.A., USA
 - anti-discriminatory, antidiscriminatory
 - résumé vs. resume
 - polskie znaki diakrytyczne
 - umlauty
- zamieniamy wszystkie litery na małe
- stosujemy słowniki (**car** = **automobile**, **color** = **colour**)
- próbujemy poprawić literówki (słowniki/soundex)

Lematyzacja

- am, are, is -> be
- car, cars, car's, cars' -> car
- “the boy's cars are different colors” -> “the boy car be different color”

Stemming

Sprowadzenie do podstawowej formy fleksyjnej

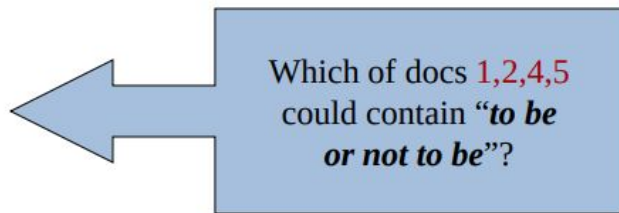
- Ujednolica słowa (czasowniki sprowadza do formy bezokolicznika)
- Zwija formy mnogie do pojedynczej
- Usuwa afiksy (przedrostki, przyrostki)

np: walking -> walk

Wyszukiwanie fraz

1. tokenem może być para słów: np: "stanford university"
 - zapytanie "stanford university palo alto" może być rozbite na **stanford university AND university palo AND palo alto**
 - może powodować false positive
 - indeks rośnie bardzo szybko
2. w indeksie oprócz id dokumentów można zapisywać miejsca wystąpień

<**be**: 993427;
1: 7, 18, 33, 72, 86, 231;
2: 3, 149;
4: 17, 191, 291, 430, 434;
5: 363, 367, ...>



Ranked retrieval

- Cały czas ogranicza nas wyszukiwanie binarne (dokument pasuje albo nie)
- Łączenie tokenów AND daje za mało (często zero) wyników, a OR daje za dużo.
- Czas dołożyć trafność - przypisanie parze query-document liczy od 0 do 1
- W ten sposób ograniczamy wyniki do 10 najlepszych, albo jeśli nic nie znajdziemy, pokażemy dokumenty najbardziej zbliżone

Trafność - indeks Jaccarda

Jeżeli w jednym dokumencie słowo występuje częściej niż w innym, to zapewne ten dokument lepiej odpowiada na zapytanie użytkownika

- $\text{jaccard}(A,B) = |A \cap B| / |A \cup B|$
- $\text{jaccard}(A,A) = 1$
- $\text{jaccard}(A,B) = 0$ if $A \cap B = 0$

Indeks Jaccarda - przykład

Ile wyniesie indeks Jaccarda dla każdego dokumentu w poniższym przykładzie?

- Query: **ides of march**
- Document 1: **caesar died in march**
- Document 2: **the long march**

Indeks Jaccarda - problemy

- Nie bierze pod uwagę **term frequency**
 - dokumenty, które dużo mówią o danym zagadnieniu powinny mieć wyższy score
- Nie bierze pod uwagę **rzadkości**
 - Rzadko występujące słowa niosą więcej informacji niż stopwords
- Potrzeba normalizować długość
 - np: $|A \cap B| / \sqrt{|A \cup B|}$

tf score

term frequency - liczba wystąpień słowa w dokumencie

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	1
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

tf score

Jest wiele różnych podejść do wyliczania tf score, np:

weighting scheme	tf weight
binary	0, 1
raw count	$f_{t,d}$
term frequency	$f_{t,d} / \sum_{t' \in d} f_{t',d}$
log normalization	$\log(1 + f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K) \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

Score dla pary query-document to suma po wszystkich tf score dla słów w query

idf weight

Rzadkie słowa niosą więcej informacji niż “pospolite”

Przykład: w zapytaniu pojawia się słowo **arachnogenic**

Jeśli w dokumencie pojawiło się słowo **arachnogenic**, to zapewne odpowiada on na zapytanie bardziej niż dokument, w którym są wszystkie pozostałe słowa zapytania

Chcemy podnieść wagę takich słów

idf weight

- df_t (document frequency)
 - liczba dokumentów, w których występuje słowo t
 - jest odwrotną miarą informatywności słowa t
 - $df_t \leq N$ (N - liczba dokumentów w zbiorze)
- idf_t (inverse document frequency)

$$idf_t = \log_{10}(N/df_t)$$

- logarytm ma za zadanie “stłumić” efekt (podstawa logarytmu nie ma znaczenia)
- Jest tylko jedna liczba df_t i idf_t dla każdego słowa w zbiorze

idf weight - przykład

Założmy, że $N = 1$ milion dokumentów

term	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1,000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

idf weight

Wpływ wagi idf dla zapytań złożonych z wielu słów:

Zapytanie: **capricious person**

Słowo **capricious** będzie miało większe znaczenie w ustalaniu trafności, gdyż występuje rzadziej niż słowo **person**

tf.idf

tf-idf

tf x idf

Miara **tf.idf**

jest połączeniem **tf score** (term frequency) i **idf** (inverse document frequency)

$$w_{t,d} = (1 + \log tf_{t,d}) \times \log_{10}(N/df_t)$$

- Rośnie wraz ze wzrostem liczby wystąpień słowa w dokumencie
- Rośnie wraz z rzadkością wystąpień słowa w zbiorze

Score tf.idf

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

$$\text{Score}(q,d) = \sum_{t \in q \cap d} \text{tf.idf}_{t,d}$$

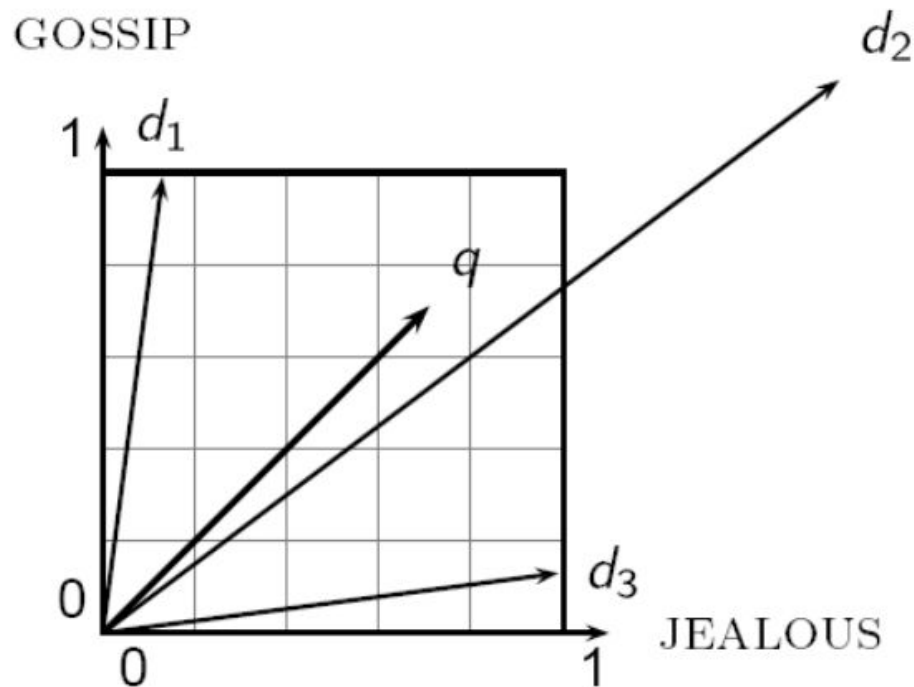
Dokumenty jako wektory

- Wyobraźmy sobie przestrzeń N -wymiarową
- Wymiary odpowiadają unikalnym słowom
- Dokumenty reprezentowane są jako punkty/wektory w tej przestrzeni
(są to rzadkie wektory, gdyż w większości wymiarów są to zera)

Zapytania jako wektory

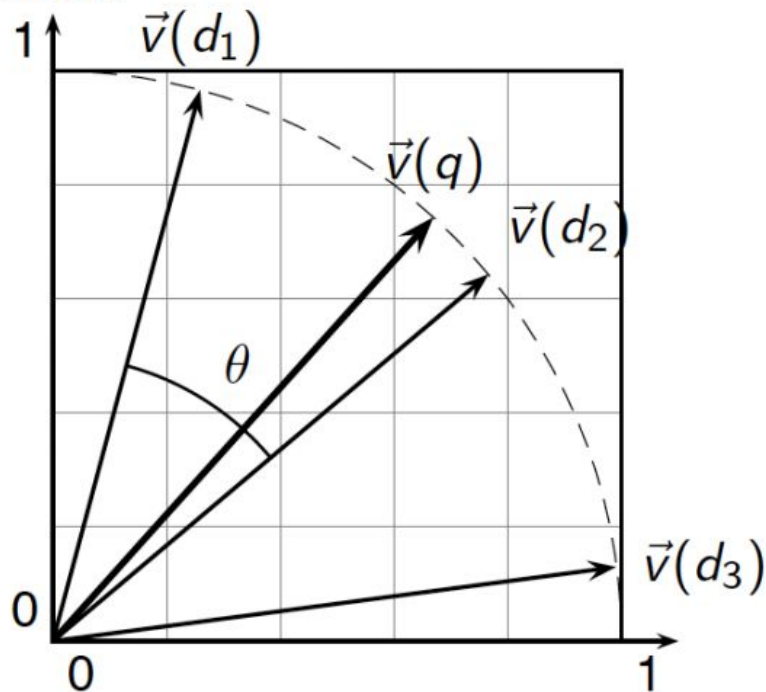
- Do ww. przestrzeni dołożmy zapytanie
- Najbardziej trafne będą dokumenty najbliższe naszemu zapytaniu

Odległość euklidesowa - kiepski wybór



Odległość kosinusowa

POOR



Wektor może być znormalizowany przez podzielenie każdego z wymiarów przez normę L_2 :

$$\|\vec{x}\|_2 = \sqrt{\sum_i x_i^2}$$

Podzielenie wektora przez jego normę L_2 zamienia go w wektor jednostkowy położony na hipersferze

RICH

Odległość kosinusowa

W

$$\cos(\vec{q}, \vec{d}) = \frac{\vec{q} \bullet \vec{d}}{|\vec{q}| |\vec{d}|} = \frac{\vec{q}}{|\vec{q}|} \bullet \frac{\vec{d}}{|\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2} \sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

q_i is the tf.idf weight of term i in the query

d_i is the tf.idf weight of term i in the document

Odległość kosinusowa

Dla znormalizowanych wektorów odległość kosinusowa to zwykły iloczyn skalarny

$$\cos(\vec{q}, \vec{d}) = \vec{q} \bullet \vec{d} = \sum_{i=1}^{|V|} q_i d_i$$

Odległość kosinusowa - przykład

Jak bardzo podobne do siebie są te powieści?

- SaS: *Sense and Sensibility*
- PaP: *Pride and Prejudice*
- WH: *Wuthering Heights?*

term	SaS	PaP	WH
affection	115	58	20
jealous	10	7	11
gossip	2	0	6
wuthering	0	0	38

Term frequencies (counts)

Odległość kosinusowa - przykład

Log frequency weighting

term	SaS	PaP	WH
affection	3.06	2.76	2.30
jealous	2.00	1.85	2.04
gossip	1.30	0	1.78
wuthering	0	0	2.58

After length normalization

term	SaS	PaP	WH
affection	0.789	0.832	0.524
jealous	0.515	0.555	0.465
gossip	0.335	0	0.405
wuthering	0	0	0.588

$$\cos(\text{SaS}, \text{PaP}) \approx 0.789 \times 0.832 + 0.515 \times 0.555 + 0.335 \times 0.0 + 0.0 \times 0.0 \approx 0.94$$

$$\cos(\text{SaS}, \text{WH}) \approx 0.79$$

$$\cos(\text{PaP}, \text{WH}) \approx 0.69$$

Ranked search vs wyszukiwanie binarne

- Wyszukiwanie binarne jest szybsze
 - łatwiej porównywać id dokumentów niż wykonywać mnożenia
 - jeśli potrzeba 10 wyników, to można przerwać wcześniej, ranked search wymaga sprawdzenia wszystkich dokumentów
- Wyszukiwanie binarne wymaga mniej pamięci
- Jednak użyteczność ranked search wygrywa!

Naive Bayes

Naive Bayes jest
dobry do zagadnień
klasyfikacji, np.
wykrywanie spamu

Example:
What is the probability to see a
dinosaur in the street?

Walking in the street you see this:

(that is our observation x)



Lets calculate what is the
probability that, conditioned on
our observation, it is truly a
dinosaur

$$P(y|x) = \frac{P(x|y) \cdot P(y)}{P(x)}$$

Likelihood – how
probable is it to see such
a thing, given that it is a
dinosaur. *Does it really
look like it?*

Prior probability to see
a dinosaur

Prior probability to see
such a scene


Inne zagadnienia text mining

- Oznaczanie części mowy
- Analiza sentymentu
 - naiwne podejście: “dobre” i “złe” słowa
 - negacje: “nie jest dobrze”
 - demo: <http://nlp.stanford.edu:8080/sentiment/rntnDemo.html>

Narzędzia

Polski wordnet: <http://plwordnet.pwr.wroc.pl/wordnet/>

Polski stemmer: <https://github.com/morfologik/polimorfologik>

RZECZOWNIK	samochód 1  napędzany silnikiem pojazd mechaniczny przeznaczony do przewożenia po drogach ludzi i różnego rodzaju ładunków
DOMENA	wytwory ludzkie (nazwy)
PRZYKŁADY	Najczęściej wykorzystywanym w samochodach typem silnika jest silnik spalinyowy tłokowy. Muszę pamiętać, że za tydzień kończy mi się ubezpieczenie samochodu.
SYNONIMY	auto 1 napędzany silnikiem pojazd mechaniczny przeznaczony do przewożenia po drogach ludzi i różnego rodzaju ładunków pojazd samochodowy 1 pojazd silnikowy, którego konstrukcja umożliwia jazdę z prędkością przekraczającą 25 km/h wóz 1 samochód, napędzany silnikiem pojazd mechaniczny przeznaczony do przewożenia po drogach ludzi i różnego rodzaju ładunków
ŹRÓDŁO	Słowność
HIPERONIMY	Pokaż ścieżkę do najwyższego hiperonimu`
GRAPH	Wyświetl wizualizację graficzną

Podsumowanie

- Dane typu unstructured stanowią zdecydowaną większość
- Wyzwania:
 - Skomplikowane i subtelne zależności między słowami
 - Dwuznaczność i zależność od kontekstu (apple vs Apple)
 - Normalizacja
 - Zaszumione dane (np. błędy w pisowni)
 - Wiele innych
- Wyszukiwanie binarne jest użyteczne tylko w ograniczonych przypadkach
- Tekst mining jest fajny :)