



Roman Isaev  
Daniel Janota

## Popis projektu

- Cílem projektu je implementace rozšířeného booleovského modelu, který řeší problematiku normálního booleovského modelu a jeho výsledcích bez hodnocení jejich relevantnosti.
- 

## Způsob řešení

### Inverted index list

- Klíčovým pro řešení této úlohy byl datový typ Pythonu dictionary (nebo-li slovník, mapa).
- Naší sadou klíčů je množina všech termů, které se v sadě daných dokumentů vyskytují.
- Hodnotou je seznam, jehož prvky mají 2 elementy - jednak ID dokumentu, ve kterém se term vyskytuje, a jednak váhu, kterou tento term v daném dokumentu má.
- Každý seznam je seřazen vzestupně dle čísla ID, ve kterém se term vyskytuje.

### Term by document matrix

- Narozdíl od inverted listu, který má pro každý term IDs výhradně těch dokumentů, ve kterých se vyskytuje, matice si ukládá ID i těch dokumentů, ve kterých příslušné termy nejsou. Váha je v takovém případě nulová.

### Dotazování

- Dotazy tvaru “blue AND pink”, “NOT (strong AND NOT(weak))”, “green or (yellow and ( not blue ))” aj. jsou považovány za korektní.
  - Dotazy tvaru “blue AND pink ”, “NOT strong AND NOT(weak )”, “green or yellow and ( not blue )” “\$ #”, “@ or .”, aj. jsou považovány za nekorektní.
  - Pokud se booleovský dotaz odpovídá nějakým dokumentům, dotaz vrátí seznam dokumentů seřazený dle relevance.
-

# Implementace

## Python

- Tento jazyk byl zvolen jednak pro jednoduchost jeho syntaxe, a jednak pro jeho velmi pestrou nabídku všelijakých knihoven (Pyeda, NLTK, ...).

## Stop words

- Stop words jsou běžná slova s velmi vysokou četností, avšak téměř nulovou informační hodnotou.
- Pro angličtinu se jedná o slova jako: "a, the, for, then, have, be"

## Lemmatizace

- Použili jsme nástroj WordNetLemmatizer z knihovny NLTK.
- Lemmatizace je pokročilejší způsob předzpracování textu, kdy dojde ke sjednocení významově identických slov.
- Například slova "go, goes, went, gone" se všechny namapují na "go"
- Nástroj pro lemmatizování je dostatečně chytrý na to, aby věděl, kdy lemmatizuje podstatné jméno, kdy přídavné jméno, a kdy sloveso. Toho jsme ostatně využili.

## Tvorba inverted listů

- Nejdříve jsme si prošli jednotlivé dokumenty a pro každý z nich vytvořili slovník, který měl jako klíč term, jenž se v daném dokumentu vyskytuje, a jako hodnotu číslo, které ukazuje, kolikrát se tento term vyskytuje v dokumentu. Tuto sadu slovníků jsme využívali při tvorbě inverted index listů.
- Také jsme vytvořili globální slovník, který celkovou četnost termů napříč celým datasetem.

## Parsování dotazů

- Pro zjednodušení parsování dotazů byla využita knihovna Pyeda. Tato knihovna umí převést například dotaz "(red OR white) AND NOT blue" do tvaru "And(Or(red, white), ~blue)", což výrazně ulehčuje parsování, neboť pořadí operací je zleva doprava, nikoliv na náhodném místě v rámci celého dotazu. Nově vytvořený dotaz se dal díky Pyeda jednak zjednodušit, a jednak převést to Disjunktivní Normální Formy. Pro parsování přetvarovaného dotazu byly využity vlastní funkce, napsané v Pythonu.
-

## Příklad výstupu

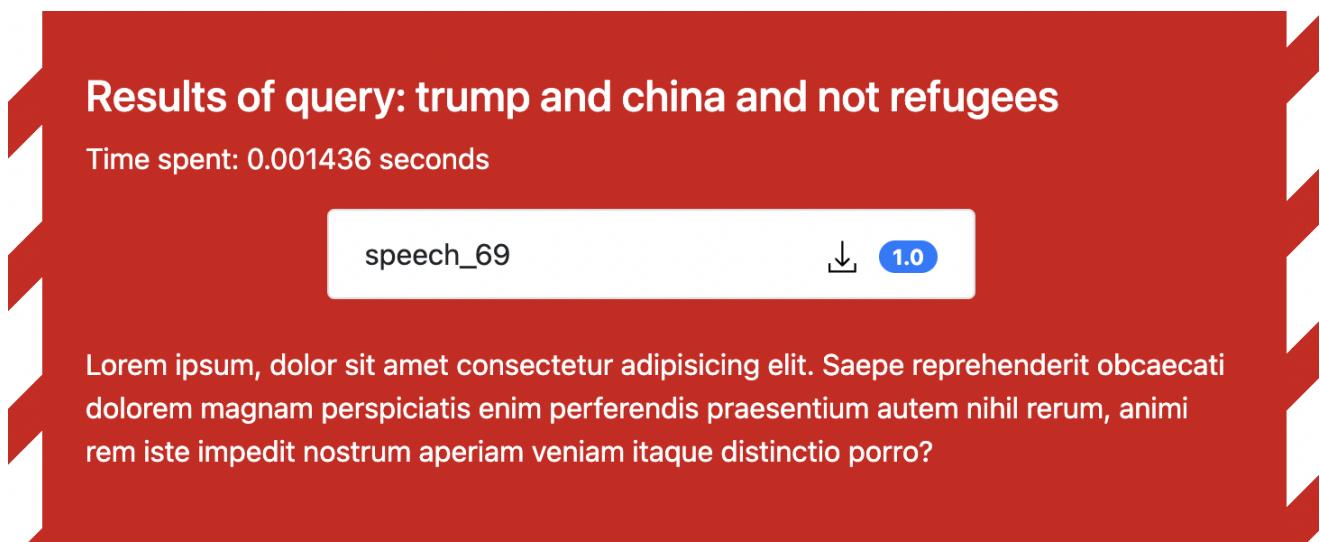
- Vstupem programu je dotaz složený z jednotlivých termů a základních logických operací



Enter your desired query

Submit

- Výstupem programu je seznam relevantních dokumentů seřazených sestupně včetně odkazu ke stáhnutí/zobrazení obsahu daných souborů.
- Relevanci jsme normalizovali (hodnoty se tedy pohybují mezi 0.0 až 1.0).



### Results of query: trump and china and not refugees

Time spent: 0.001436 seconds

speech_69	 1.0
-----------	---

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Saepe reprehenderit obcaecati dolorem magnam perspiciatis enim preferendis praesentium autem nihil rerum, animi rem iste impedit nostrum aperiam veniam itaque distinctio porro?

## Experimentální sekce

- V tomto zadání se dá překvapivě experimentovat s lecčim, žel na to nebylo dostatečně času.
  - Případnými optimalizacemi by mohly být věci, jako jsou multihtreading, průběžné ukládání výsledků poddotazů do paměti.
  - Jednou z důvodů optimalizací bylo průběžné doplňování doplňku inverted listů, které v paměti již existují. Např.: je-li v paměti inverted index list pro term Blue, a proběhl-li již dotaz "Blue", pak se v paměti vytvoří inverted index list (popř. matice) pro ~Blue, která má v sobě ID dokumentů, ve kterých Blue není. V budoucnu tedy dotaz typu "Yellow OR Green AND NOT BLUE" proběhne rychleji, neboť IDs dokumentů pro "NOT BLUE" již v paměti jsou a nemusí se to znova počítat.
- 

## Diskuse

- Jedním z pozorovaných nedostatků algoritmu je poměrně strojený způsob zadávání dotazů, jehož parsování na námi požadovaný formát vstupu je samo o sobě velmi obsáhlou problematikou. Byť se tato část úlohy výrazně zjednodušila díky již zmíněné knihovně Pyeda. Napsat neprůsřelný parser bez využití této knihovny je velmi složitá úloha.
- 

## Závěr

- Rozšířený booleovský model je poměrně efektivní a na svou jednoduchost přesný algoritmus pro vyhledávání relevantních výsledků.
  - Například oproti jednoduchému pattern matchingu má jednoznačně velké výhody právě v obecnosti použití a relevantnosti vrácených výsledků.
  - Term by document matici by se NIKDY neměla používat. Při 1.000.000 dokumentech, v rámci kterých se slovo Pong vyskytne například jen 1 krát, je zcela neefektivní mít 1.000.000 sloupců pro Pong, ze kterých jen 1 bude mít nenulovou váhu. Kdyžto inverted index list má jednoduše jednoprvový seznam pro term Pong.
-