Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Propojení témat zpravodajských článků mezi jazyky

Plzeň 2019 Petr Kopal

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 25. června 2019

Petr Kopal

Abstract

The goal of this master thesis was to research cross-lingual document similarity methods, which were used then to design a system for linking of similar news topics across languages. Except of the common entities and Eurovoc descriptors, word embeddings models (CL-ESA, K-means, CL-LSA) were used as the main source of feature vectors. All these word embeddings models were trained on the Wikipedia comparable corpus. The clustering results were evaluated using various metrics (notably F-measure and purity) and documented in a separate chapter. The best results were achieved using the CL-LSA method in combination with common entities features.

Abstrakt

Cílem této diplomové práce bylo prozkoumat možnosti metod pro výpočet podobnosti textů napříč jazyky. Následně na základě těchto poznatků navrhnout systém, který bude schopen propojit tematicky podobné zpravodajské články v různých jazycích. Pro získání příznakových vektorů byly kromě společných entit a Eurovoc deskriptorů použity sémantické distribuční modely natrénované na srovnatelném korpusu Wikipedie. Konkrétně šlo o metody CL-ESA, K-means a CL-LSA. Výsledné shluky byly vyhodnoceny evaluačními metrikami (zejména pak F-mírou a purity) a zdokumentovány v samostatné kapitole. Nejlepších výsledků bylo dosaženo metodou CL-LSA v kombinaci se společnými entitami.

Obsah

1 Úvod

Informace vždy byly, jsou a s vysokou pravděpodobností i nadále budou důležitou a nedílnou součástí společnosti. V dnešní době, především díky internetu, jsou informace již o mnoho dostupnější než tomu bývalo v minulosti. Množství dat je tak již nyní nepředstavitelné a neustále roste. Proto je prakticky nemožné, i přes velký potenciál internetu, všechny články a novinky publikované za jeden den na internetu manuálně projít, přečíst či je tematicky roztřídit.

Řada velkých společností, organizací či politických stran si proto zřizuje svá vlastní oddělení, která jim pomáhají s analýzou zpravodajských článků a novinek. Jedním z jejich cílů může být například zjistit, jak se o jejich organizaci ve světě mluví, které téma je aktuálně světově nejvíce diskutované, či jak se na danou problematiku nebo událost nahlíží v jiných zemích.

To je také jeden z mnoha důvodů proč vznikly automatizované metody zkoumající podobnost textů napříč jazyky. Úkolem této práce je tyto metody prozkoumat a poté navrhnout a implementovat systém, který by umožňoval propojit témata zpravodajských článků mezi různými jazyky (konkrétně pak cz, en, de, it, fr).

Prvních několik kapitol se věnuje základním pojmům, které jsou nutné k pochopení problematiky, dále jsou vysvětleny některé jednojazyčné modely pro vnoření slov a samozřejmě jsou i popsány vhodné metody pro výpočet podobnosti dokumentů napříč různými jazyky.

V naší práci byly konkrétně použity tyto embeddingové modely: Cross-linguální explicitní sémantická analýza (CL-ESA), K-means a Cross-linguální sémantická analýza, které byly natrénovány na vícejazyčném korpusu Wikipedie a mohou být kombinovány se základními metodami pracující s Eurovoc tezaurem a společnými entitami.

Praktická část je pak zaměřena na popis architektury, předzpracování a získání trénovacích dat a samotnou implementaci systému. Na závěr byly provedeny testy na reálných datech a výsledky těchto testů byly následně zdokumentovány a prodiskutovány v kapitole 9.

2 Notace

Na samý úvod si nejprve uvedeme potřebné notace, které se v práci budou vyskytovat. Tato kapitola slouží pouze pro sjednocení a zpřehlednění značení v rámci práce a všechny pojmy budou podrobněji vysvětleny až v dalších kapitolách.

2.1 Souhrn základních notací

- d - označení dokumentů
- D - označení jazykového korpusu
- s - počet dokumentů v korpusu
- $|D|$ - celkový počet dokumentů
- N - celkový počet slov v korpusu
- n_i - velikost slovníku pro jazyk i
- u_i - jazyk i
- L_m - množina jazyků
- f_{ij} - počet výskytů termu t_i v dokumentu j
- $\phi(d)$ - TF-IDF reprezentace dokumentu
- $||.||$ - Eukleidovská norma
- $\langle \phi(d_1), \phi(d_2) \rangle$ - skalární součin
- X - VSM term-dokument matice (nxs)
- S_{ij} - transformační matice mezi jazyky i a j
- P_i - pseudoinverzní matice
- k - počet clusterů, resp. dimenzí
- E - množina konceptů
- e_i - koncept i

- V_{ij} - odhad kovarianční matice
- T - množina slov představující slovník
- w - slovo
- Z - počet kontextových slov
- z_i - $i-t$é kontextové slovo
- Oij - matice souvýskytů w_i s kontextovým slovem z_j
- $\tilde{x_i}$ - embeddings kontextového slova
- $sent_i^{l_1}$ - párově zarovnaná věta i ze zdrojového jazyka l_1
- $sent_i^{l_2}$ - párově zarovnaná věta i z cílového jazyka l_2
- b_i - bias korespondující s w_i
- $\tilde{b_j}$ - bias korespongující s kontextovým slovem z_j
- C - množina tříd
- c_i - třída i
- Ω - množina clusterů
- ω_i - cluster i
- a_ij - počet prvků z třídy c_i patřící do clusteru ω_j

3 Základní pojmy

3.1 Jazykový korpus

Jazykový korpus je rozsáhlá kolekce textů (dokumentů) určitého jazyka, kterou produkují skuteční uživatelé jazyka. Nachází své využití zejména v lingvistice, při tvorbě slovníků, korektorů či v pro nás nejpodstatnější oblasti zpracování přirozeného jazyka. V současnosti mají korpusy digitální podobu, což výrazně usnadňuje jejich sběr a zpracování. V softwarovém inženýrství se pak využívají zejména k vytvoření různých jazykových databází.

Jazykový korpus může být rozdělen do několika kategorií na základě obsahu, metadat, multimediálního obsahu či vztahu k jinému korpusu. Jeden korpus může být současně ve více kategoriích, pokud pro ně splňuje příslušná kritéria. Zde si popíšeme pouze ty typy, které jsou k tématu této práce relevantní a mohou se objevit v dalších kapitolách.

3.1.1 Jednojazyčný korpus

Jednojazyčný korpus je nejběžnějším typem korpusu. Jak již název napovídá, tento druh korpusu obsahuje texty pouze v jednom jazyce. Je používán širokou základnou uživatelů pro velké množství úkolů. Od ryze praktických (např. kontrola správného použití slova v textu, hledání nejpřirozenějšího vyjádření aj.) až k vědeckým účelům (např. identifikace nových vzorů, trendů v jazyce).

3.1.2 Vícejazyčný korpus

Vícejazyčný korpus obsahuje texty v několika různých jazycích, které jsou buď vzájemnými překlady, nebo alespoň pojednávají o stejném tématu. Proto dále vícejazyčný korpus rozdělujeme na paralelní a srovnatelný.

Paralelní

Paralelní korpus se skládá z množiny jednojazyčných korpusů a to tak, že tyto korpusy jsou vzájemnými jazykovými alternativami a musí být zarovnány, tzn., že si musí odpovídat příslušnými segmenty (obvykle věty nebo odstavce). Uživatel tak může sledovat, jak je hledané slovo nebo fráze ze zdrojového jazyka přeloženo v kontextu druhého jazyka.

Formálně tedy pro zdrojový dokument v jazyce L_1 máme k dispozici i jeho překlady v jazycích $L_2, ..., L_n$, přičemž ve většině případů se paralelní korpus skládá pouze ze dvou jazyků.

Srovnatelný

Srovnatelný korpus (v originále comparable) je množina dvou nebo více jednojazyčných korpusů, které pojednávají o stejném tématu, ale nejsou však vzájemnými překlady, a proto nejsou přímo zarovnány. Většinou mívají stejná metadata. Příkladem mohou být korpusy Wikipedie, kde jedno téma obecně bývá popsáno ve více jazycích zároveň.

3.2 Vektorová reprezentace textu

Text, na rozdíl od obrazu či zvuku, nemá vlastnost přirozené vektorové reprezentace. Ta je nezbytná např. pro porovnání dokumentů, vyhledávání a zpracování objemných dokumentů. Proto byly vymyšleny metody a způsoby jak text do vektorů převést. Souhrnně tyto techniky nazýváme distribuční sémantické modely (v originále word embeddings).

Protože složitější metody zabývající se word embeddings se téměř vždy odkazují na základní techniky reprezentace textu, vysvětlíme si nyní několik prvotních přístupů reprezentace textu a získané poznatky pak dále uplatníme v kapitolách 4 a 5.

3.2.1 One-hot

One-hot reprezentaci textu si můžeme představit jako vektor o délce rovné počtu slov v textu, kde nenulový prvek je vždy pouze jeden, a ten symbolizuje dané slovo. Nejprve tedy vytvoříme z termů slovník a každému termu přiřadíme unikátní identifikátor. Uvažujme např. následující jednoduchou množinu slov: `{cat, dog, mouse, window}`. Pak vektor reprezentující slovo `dog` bude vypadat následovně:

$$v_{dog} = (0, 1, 0, 0) \tag{3.1}$$

Tento model však neposkytuje kromě prezence slova ve větě (resp. v dokumentu) žádnou další dodatečnou informaci. Na této myšlence jsou však založeny další reprezentace popsané později.

3.2.2 Bag of words

Bag of words (BoW) reprezentace textu je způsob reprezentace, která neuchovává žádnou informaci o uspořádání slov (proto Bag of Words). Získáme ji takto: Nejprve ze všech dokumentů vytvoříme slovník (abecedně seřadíme termy) a konkrétní dokument pak reprezentujeme jako vektor, kde každá složka udává počet výskytů slova v dokumentu.

Formálně tedy můžeme dokument d vyjádřit vektorem $x \in R^n$, kde n představuje velikost slovníku a prvek x_k odpovídá počtu výskytů k-tého slova dokumentu d (viz term frequency). Mějme tyto 2 anglické věty představující naše dokumenty:

- `d1: The quick brown fox jumped over the lazy dog`
- `d2: The dog woke up and started chasing the fox`

výsledné vektory pak budou vypadat takto (termy abecedně seřazeny):

- `d1: (0,1,0,1,1,1,1,1,1,0,2,0,0)`
- `d2: (1,0,1,1,1,0,0,0,0,1,2,1,1)`

3.2.3 Vector Space model

Vector Space model (VSM) je technika založená na BoW. Funguje tak, že pro daný dokument vytváří jeden výsledný vektor, který ho reprezentuje. Každý prvek tohoto vektoru označuje skóre konkrétního termu a vzniká postupným vážením vektorů všech termů do jednoho (např. TF-IDF - vysvětleno dále). Skupina dokumentů je pak zobrazena do společného vektorového prostoru (Odtud Vector space model), v němž je možné dokumenty dále porovnávat [6].

Touto technikou stejně jako samotným BoW, ale často získáváme obrovské řídké matice, jejichž zpracování může být značně paměťově náročné, a proto vznikly další pokročilejší metody jako např. LSA (viz kapitola 4.1).

3.3 Term frequency

Schéma vážení pomocí term frequency pracuje s myšlenkou, že některé termy se v dokumentu vyskytují častěji než jiné a výsledná váha termu by tedy měla odpovídat jeho důležitosti pro daný korpus. Nejznámější schéma pro vážení termů se nazývá Term frequency Inverse frequncy (TF-IDF) vážení.

TF-IDF

TF-IDF je metodika používající se ke zjišťování relevance dokumentů. Název je spojením zkratek dvou termínů.

TF složka (term frequency) vyjadřuje, jak často se výraz vyskytuje v příslušném dokumentu. Většinou dochází také k normalizaci délkou dokumentu, aby nebyly znevýhodňovány krátké dokumenty před dlouhými, ve kterých by se mohl výraz vyskytovat častěji, aniž by byl dokument relevantnější. TF udává následující vzorec:

$$TF_{i,j} = \frac{f_{i,j}}{\sum_k f_{k,j}} \tag{3.2}$$

Kde $f_{i,j}$ je počet výskytů termu t_i v dokumentu d_j. Jmenovatel pak reprezentuje součet počtu výskytů všech slov v dokumentu d_j.

Složka IDF (Inverse dokument frequency) představuje důležitost termu v korpusu. Vychází z myšlenky, že čím častěji se term v dokumentech vyskytuje, tím méně je důležitý. IDF termu i vypočítáme tímto vzorcem:

$$IDF_i = log\frac{|D|}{DF_i} \tag{3.3}$$

|D| udává celkový počet dokumentů, ve kterých hledáme a DF_i je počet dokumentů obsahující term i.

Formálně pak TFIDF reprezentaci dokumentu d můžeme vyjádřit pomocí mapovací funkce $\phi : text \to R^n$, kde n představuje velikost slovníku

3.4 Podobnost dokumentů

Častým problémem v oblasti vyhledávání informací (information retrieval) je hledání relevantních dokumentů k dotazu či podobných dokumentů k vstupnímu dokumentu. Prozatím se omezíme pouze na jednojazyčný korpus.

Jelikož se nacházíme ve vektorovém prostoru, tak první myšlenkou, která by nás mohla napadnout, je počítat podobnost dvou dokumentů jako Eukleidovskou vzdálenost koncových bodů dvou vektorů. Výsledná podobnost nemusí být vůbec špatná, nicméně tato myšlenka nepočítá s tím, že Eukleidovská vzdálenost je velká pro vektory, které mají výrazně rozdílnou délku. Z toho vyplývá, že např. menší dokumenty, které by mohly být dostatečně podobné ke vstupnímu velkému dokumentu, nebudou správně vyhodnoceny a naopak.

Lepším způsobem jak získat podobnost dvou dokumentů/vektorů je pomocí úhlu, kterého mezi sebou svírají. Nejznámější metoda, která dokáže

vypočítat podobnost dvou dokumentů či dotazu a dokumentu, se nazývá kosinová podobnost.

3.4.1 Kosinová podobnost

Kosinová podobnost (v originále cosine similarity) pracuje s myšlenkou, že dva dokumenty jsou si nejvíce podobné, pokud jejich vektory mezi sebou svírají úhel 0°. Např. pokud máme dokument d_1 a dokument d_2, který je pouze dvojnásobnou kopií dokumentu d_1, získáme úhel 0° a považujeme pak dokumenty za maximálně podobné. Eukleidovská vzdálenost těchto dvou dokumentů by však zřejmě byla velká.

Kosinovou podobnost dvou různých dokumentů lze vypočítat následujícím vzorcem:

$$sim(d_1, d_2) = \frac{\langle \phi(d_1), \phi(d_2) \rangle}{\|\phi(d_1)\| \, \|\phi(d_2)\|} \tag{3.4}$$

kde $\langle ., . \rangle$ představuje skalární součin TF-IDF vektorů a $\|.\|$ představuje Eukleidovskou normu resp. velikost vektoru.

3.4.2 Kosinová podobnost napříč jazyky

Pokud bychom chtěli počítat kosinovou podobnost napříč různými jazyky, nemůžeme bohužel použít standardní vzorec pro jednojazyčnou podobnost, ale vektory reprezentující dokumenty musíme transformovat do společného prostoru. K tomuto účelu definujeme bilineární operátor (zmíněný v práci [15]) v podobě vyhodnocovací transformační matice $S_{i,j} \in R^{n_i \times n_j}$, jejímž úkolem je transformovat dokument z jazyka i do jazyka j a je definován takto:

$$sim_{i,j}(d_1, d_2) = \frac{\langle \phi_i(d_1), S_{i,j}\phi_j(d_2))) \rangle}{\|\phi_i(d_1)\phi_j(d_2)\|} \tag{3.5}$$

Jak získat transformační matici $S_{i,j}$ si ukážeme blíže až v kapitole 6.3.

4 Distribuční sémantické modely

Jak již bylo řečeno v úvodní kapitole, jedním z cílů této práce je nalézt vhodné metody výpočtu podobnosti textů napříč různými jazyky. Abychom však mohli tento úkol splnit, je nejprve zapotřebí získat základní znalosti o distribučních sémantických modelech, protože většina metod pracující s podobností textů napříč jazyky je na těchto modelech založena nebo se na ně alespoň nějakým způsobem odkazuje.

Distribuční sémantické modely (word embeddings) je souhrnný název pro metody jazykového modelování a získávání příznaků ve zpracování přirozeného jazyka, kde slova ze vstupních dokumentů jsou zobrazena do vektorů reálných čísel tak, aby se podobná slova ve výsledném prostoru nacházela blízko sebe. Word embeddings představují vektory, jejichž relativní podobnosti korelují se sémantickými podobnostmi. V současnosti se jedná o jeden z největších trendů v oblasti NLP. Koncepčně zahrnuje matematické vnoření z prostoru, kde každému slovu odpovídá jedna dimenze, do výsledného souvislého prostoru s mnohem menší dimenzí. Tento přístup lze zobecnit i pro celé věty či dokumenty.

Metody typově zahrnují např. dimenzionální redukci matic, neuronové sítě či pravděpodobnostní modely. Následující sekce pojednávají o nejpoužívanějších a nejznámějších jednojazyčných modelech vnoření slov.

4.1 Latentní Sémantická Analýza

Latentní sémantická analýza (LSA) [2] je technika, která zobrazuje dokumenty a dotazy do prostoru latentních sémantických dimenzí, přičemž slova, která jsou sémanticky podobná (měřeno mírou sou-výskytů v dokumentech), jsou zobrazována do stejných dimenzí a slova sémanticky odlišná do různých dimenzí. Dimenze zde spíše než slovům odpovídají tzv. konceptům - tematickým oblastem. Díky tomu mohou mít velkou sémantickou podobnost i dokumenty, které spolu ani nesdílejí žádná slova. Blíže vysvětleno v [7].

LSA, často bývá také označována jako Latentní sémantické indexování (LSI), navazuje na dříve zmiňovanou techniku VSM a zároveň ji i zdokonaluje. Jde vlastně o přesné aplikování analýzy hlavních komponent (v originále Principal Component Analysis - PCA) [4] na řídkou VSM term-dokument

matici X (n x s) za účelem redukce její dimenze, kde n udává počet unikátních termů - velikost slovníku a s je počet dokumentů. Konkrétně je použit singulární rozklad matice X (v originále Singular Value Decomposition - SVD) na součin tří dílčích aproximujících matic, který vypadá takto:

$$X = U\Psi V^T \tag{4.1}$$

$$X_k = U_k\Psi_k V_k^T \tag{4.2}$$

Kde U a V jsou matice ve sloupcově ortonormální formě s redukovanou dimenzí $k < s$ (za předpokladu, že $n > s$) a Ψ je diagonální matice obsahující prvních k singulárních hodnot.

Pokud vybereme k největších singulárních hodnot a příslušné singulární vektory z U a V, získáme aproximovanou matici X_k s nejmenší chybou měřenou Frobeniovou normou:

$$\|X - X_k\|_F = \sqrt{\sum_i^n \sum_j^s |x_{ij} - \dot{x}_{ij}|^2} \tag{4.3}$$

4.2 Explicitní sémantická analýza

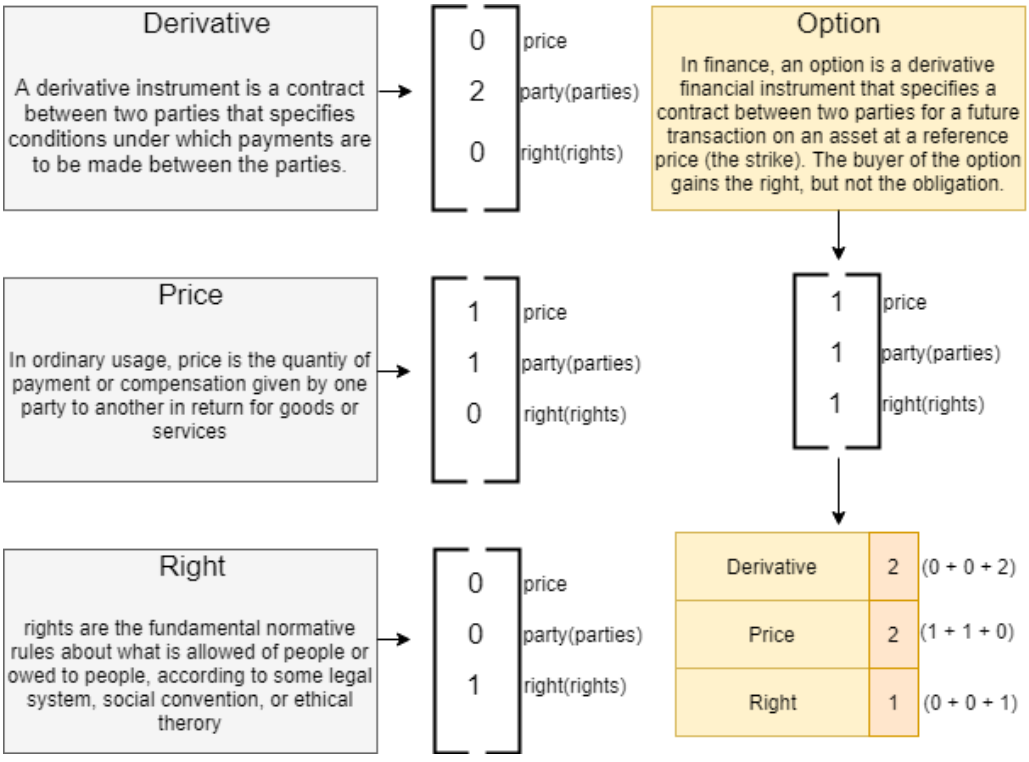
Explicitní sémantická analýza (v originále Explicit semantic analysis - ESA) je metoda, která pochází od autorů Gabrilovitche a Markovitche, a poprvé se objevila v roce 2007. Explicitní model je opět konceptuálně založený model vyhledávání, který nám umožňuje explicitně reprezentovat význam dokumentů založených na konceptech. Explicitní model pracuje s externě definovanými koncepty zakořeněnými v lidském poznání. ESA metoda vlastně reprezentuje sémantiku textů přirozeného jazyka pomocí přirozených konceptů, a proto je jednoduchá na pochopení [14].

Ve standardním VSM synonyma nijak nepřispívají k výsledné podobnosti dokumentů. Dokumenty, které jsou si sémanticky podobné (tj. pojednávají o stejném tématu), si pak nemusí být ve VSM podobné, pokud nepoužívají stejná slova. ESA, stejně jako LSA, ale problém synonym a sémantické podobnosti řeší.

Nyní si popíšeme ESA trochu formálněji. Mějme externě definovaný koncept $E = \{e_1, e_2, ..., e_r\}$ pro klasickou jednojazyčnou explicitní sémantickou analýzu o předem daném počtu dimenzí r, pak vstupní dokument d reprezentovaný VSM vektorem je převeden do tzv. koncepčního vektoru. Koncepční vektor je reprezentován kolekcí dokumentů D v jazyce i, kde každá dimenze

odpovídá konkrétně jednomu dokumentu. Vstupní matice tedy bude vypadat tak, že řádky budou odpovídat termům a sloupce konceptům. Ve finále tedy budeme porovnávat podobnost mezi získanými koncepčními vektory.

Abychom vše snáze pochopili použijeme názorný příklad. Mějme tři anglické dokumenty: `Derivative`, `Price`, `Right` popisující jednotlivá témata a představující naše koncepty. Náš konceptuální prostor bude mít tedy pouze tři dimenze. Dále mějme vstupní dokument s názvem `Option`, který budeme reprezentovat v našem trojdimenzionálním konceptuálním prostoru. V této ukázce se omezíme pouze na termy `price`, `party`, `right`, ze kterých sestavíme term-vektor, ale v reálném příkladu samozřejmě použijeme všechny dostupné termy. Následně je zapotřebí spočítat tzv. asociační sílu mezi vstupním dokumentem a koncepty, která udává jak silně mezi sebou souvisejí termy ze vstupního dokumentu s termy z příslušných konceptů. Vše znázorňuje následující obrázek:



Obrázek 4.1: ESA - mapování vstupního dokumentu do konceptuálního prostoru

Na tuto základní jednojazyčnou metodu dále navazuje metoda CL-ESA(6.3.4), která už se přímo hodí pro výpočet vícejazyčné podobnosti.

4.3 Max-margin hinge loss (MMHL)

Autoři Collobert a Weston [1] použili ve své práci jazykový model založený na textových oknech o dané velikosti, které slouží k natrénování neuronové sítě. Přičemž neuronová síť byla natrénována tak, aby dokázala rozeznat, zda slovo uprostřed okénka textu odpovídá jeho kontextu nebo ne.

Celý trénovací dataset vytvořili autoři pro různé velikosti textových okének získaných z korpusů anglické Wikipedie. Jako pozitivní příklady sloužila skutečná textová okénka z Wikipedie a jako negativní příklady pak ta samá okénka, jejichž prostřední slova byla nahrazena náhodným slovem.

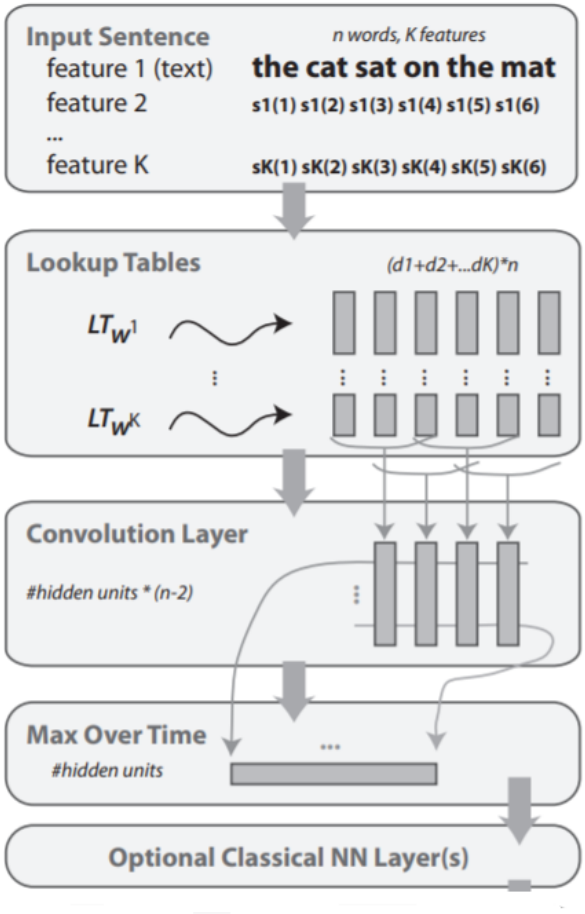
Formálně pak je metoda Max-Margin Hinge Loss (MMHL) definována takto:

$$MMHL = \sum_{sent \in S} \sum_{w \in T} max(0, 1 - fn(sent) + fn(sent^w)) \tag{4.4}$$

Kde S je množina pohyblivých okének o velikosti Z v rámci vět ze vstupního korpusu. $fn(.)$ je neuronová síť jejíž výstupní skóre je dáno vstupním pohyblivým okénkem $sent$, $sent^w$ je pohyblivé okénko, jehož prostřední slovo bylo nahrazeno slovem w a T označuje slovník.

Architekturu zmiňované neuronové sítě si můžete prohlednout na následujícím obrázku 4.2 pocházejícím z referenční práce [1],

Vytvořená architektura se ukázala jako velmi rychlá, což umožňuje pracovat s obrovskými databázemi v řádech stovek milionů slov a může být aplikována na řadů úkolů z NLP (konkrétně třeba popisování sémantických rolí, rozpoznávání jmenných entit, rozdělení textu na slovní druhy).



Obrázek 4.2: Architektura neuronové sítě pro metodu MMHL

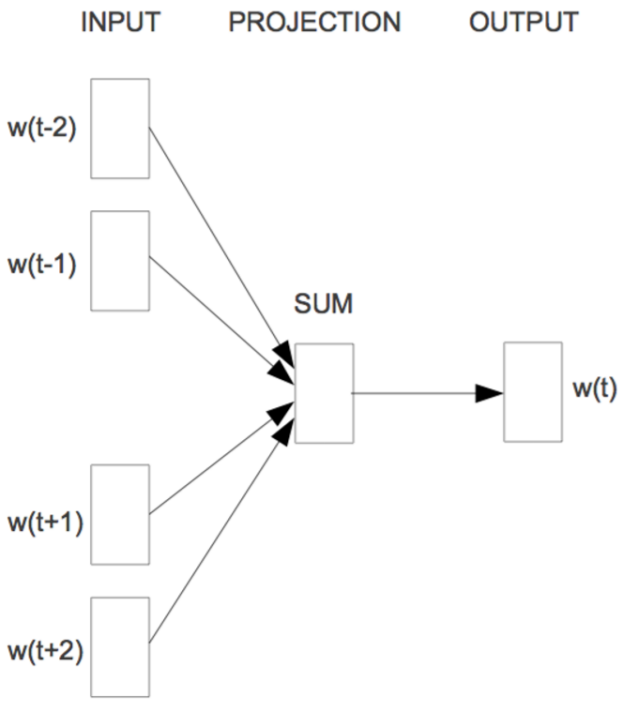
4.4 Word2vec

Word2vec je skupina příbuzných word embeddings modelů, která byla vytvořena týmem výzkumných pracovníků ze společnosti Google pod vedením T. Mikolova v roce 2013. Tyto modely jsou reprezentovány neuronovými sítěmi, které jsou natrénovány tak, aby dokázaly rekonstruovat kontextová slova. Hlavními dvěma modely, se kterými word2vec pracuje, jsou SGNS a CBOW.

SGNS

Kvůli vysoké efektivitě trénovacího procesu a své robustnosti je Skip-gram s negativním vzorkováním (SGNS) [8] pravděpodobně nejpoužívanější metodou učení word embeddings.

SGNS aproximuje jazykový model, ale spíše než na přesné modelování



Obrázek 4.3: Metoda SGNS

pravděpodobnosti slov se zaměřuje na učení efektivních slovních reprezentací. Vyvozuje tedy reprezentace, které jsou dobré v předpovědi okolních kontextových slov pro cílové slovo w_t. Blíže znázorněno na obrázku 4.3. Jednoduše řečeno, algoritmus se snaží předvídat kontext na základě aktuálního slova. Formálně pak SGNS udává vzorec 4.5.

$$SGNS = -\frac{1}{N}\sum_{t=1}^{N}\sum_{-Z\leq j\leq Z, j\neq 0} logP(w_{t+j}|w_t) \tag{4.5}$$

Kde N je počet slov z trénovacího korpusu, Z je velikost pohyblivého okénka a $P(w_{t+j}|w_t)$ vypočítáme pomocí tzv. softmax funkce:

$$P(w_{t+j}|w_t) = \frac{exp(\tilde{x}_{t+j}x_t)}{\sum_{i=1}^{|T|} exp(\tilde{x}_i x_t)} \tag{4.6}$$

Kde $|T|$ je velikost slovníku, x_i je slovo a $\tilde{x_i}$ jsou kontextové embeddings slova w_i. Vzhledem k tomu, že výpočet této funkce je poměrně časově náročný, SGNS využívá negativní vzorkování, které má za cíl zefektivnit výpočet aproximováním softmax funkce.

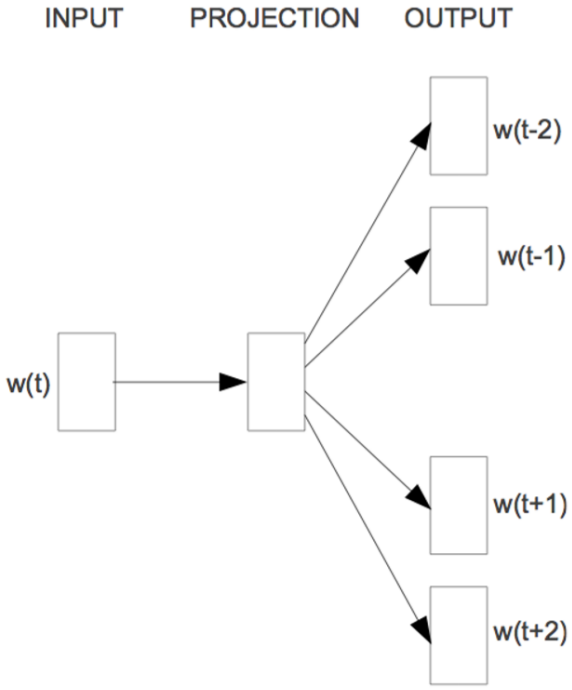
Negativní vzorkování je vlastně zjednodušení odhadu kontrastního šumu (Noise Contrastive Estimation [9]), které bylo aplikováno na jazykové modelování. Negativní vzorkování trénuje model tak, aby rozlišoval cílové slovo

w_t od jeho negativních vzorků získaných šumem. V tomto ohledu je tedy velmi blízký dříve zmiňované metodě MMHL.

CBOW

Continuous bag-of-words (CBOW) [8] je vlastně inverzí k předchozí metodě SGNS. Tedy v tomto případě je na vstupu okénko o velikosti Z obsahující kontextová slova a naším cílem je předpovědět cílové slovo w_t (viz obrázek 4.4). CBOW je definováno tímto vztahem:

$$CBOW = -\frac{1}{N}\sum_{t=1}^{N} logP(w_t|w_{t-Z}, ..., w_{t-1}, w_{t+1}, ..., w_{t+Z}) \qquad (4.7)$$



Obrázek 4.4: Metoda CBOW

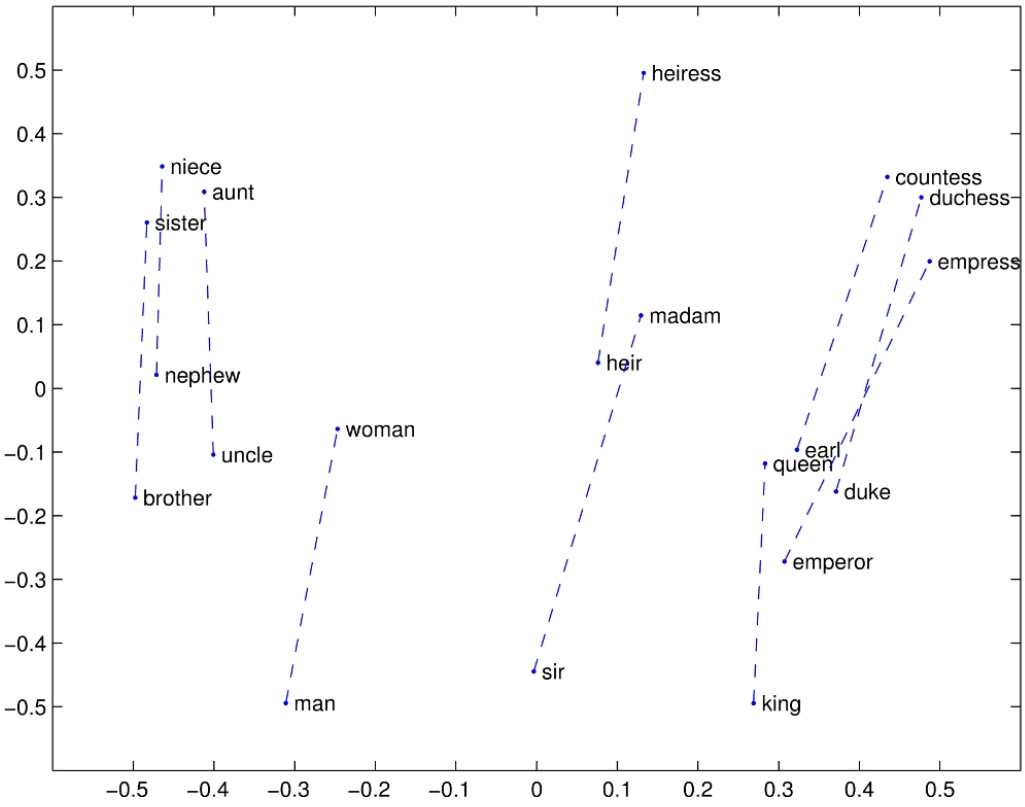
CBOW je podle autora obecně rychlejší než SGNS. SGNS bývá naopak rychlejší v případě méně frekventovaných slov.

4.5 GloVe

Pokud hledáme podobnost dvou slov, eukleidovská vzdálenost mezi nimi nemusí být dostačující, respektive téměř vždy slova vykazují složitější vztahy,

a nelze je proto zachytit pouze jedním číslem. Například slovo *muž* může být považováno za velmi podobné slovu *žena*, protože obě tato slova popisují lidské bytosti. Na druhé straně však tato slova mohou být často vnímána jako protiklady zdůrazňující, že jsou to lidé opačného pohlaví. Pokud bychom chtěli takové vztahy zachytit, je nutné, aby model spojoval více než jedno číslo s dvojicí slov, a proto vznikl GloVe, který se snaží rozdíly vektorů dané kontrastem slov co nejlépe zachytit.

Na obrázku 4.5 si můžeme všimnout, že pro matematicky vyjádřené dvojice slov: $man - woman$, $king - queen$, $brother - sister$ jsou jejich vektorové rozdíly přibližně stejné.



Obrázek 4.5: Vektorové vizualizace dvojic slov

GloVe (Globalní vektory) je v podstatě logaritmický bilineární model vážený metodou nejmenších čtverců. GloVe minimalizuje rozdíl mezi skalárním součinem embeddings slova w_i s jeho kontextovým slovem z_t a logaritmem počtu jejich sou-výskytů v rámci určité velikosti okénka:

$$GloVe = \sum_{i,j=1}^{|N|} fw(O_{ij})(x_i^T \tilde{x_j} + b_i + \tilde{b_j} - logO_{ij})^2 \tag{4.8}$$

Kde b_i a $\tilde{b_j}$ jsou biasy (kognitivní zkreslení) korespondující se slovem w_i a jeho kontextovým slovem z_j. $O_{i,j}$ zachycuje počet výskytu w_i s kontextovým slovem z_j a $fw(.)$ je váhová funkce, která přiřazuje relativně nižší váhu vzácným a velmi frekventovaným sou-výskytům.

Důležitou vlastností je pak to, že poměry pravděpodobností slovních souvýskytů mají velký potenciál pro zakódování nějaké formy významu. Tuto vlastnost si ukážeme na tabulce 3.1, kde jsou zachyceny pravděpodobnosti výskytu slov *led* a *pára* s některými kontextovými slovy v korpusu tvořeném cca 6 miliardami slov:

Ppst a poměr	**k = pevný**	**k = plyn**	**k = voda**	**k = móda**
P(1)=P(k|led)	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
P(2)=P(k|pára)	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
R=P(1)/P(2)	8.9	8.5×10^{-2}	1.36	0.96

Tabulka 4.1: Pravděpodobnostní rozložení klíčových slov v korpusu tvořeném cca 6 miliardami slov

Jak by se dalo očekávat slovo *led* se vyskytuje častěji ve spojení se slovem *pevný* než *plyn* a naopak *pára* se vyskytuje častěji se slovem *plyn* než *pevný*. Obě tato slova se vyskytují často se slovem *voda* a s nesouvisejícím slovem *móda* se vyskytují s mnohem menší frekvencí. Poměr pravděpodobností pak ruší šum pocházející od nediskriminačních slov jako *voda* či *móda*, takže vysoké hodnoty (mnohem vyšší než 1) dobře korelují s vlastnostmi typickými pro *led* a nízké hodnoty (mnohem menší než 1) pak dobře korelují s vlastnostmi typickými pro *páru*. Tímto způsobem poměr pravděpodobností kóduje nějakou informaci spojenou s abstraktní koncepcí termodynamické fáze.

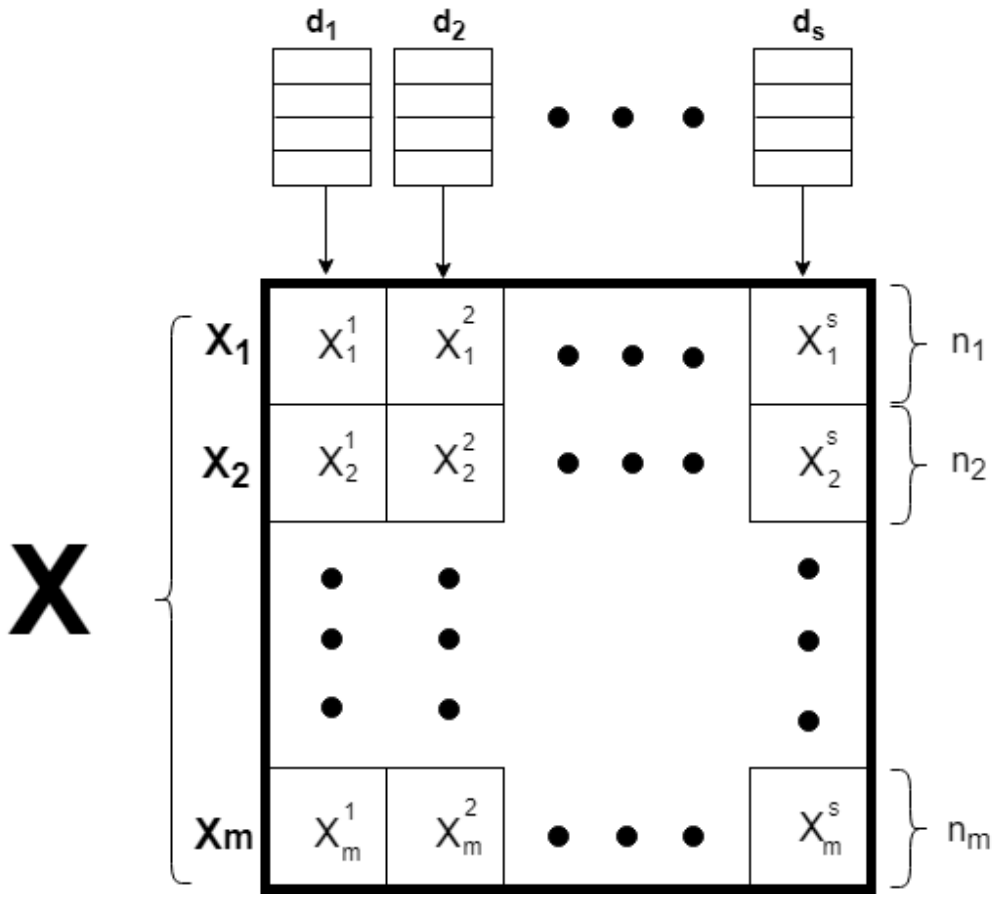
Jelikož poměry pravděpodobností mohou zakódovat nějaké formy významu a tyto informace vlastně mohou být zakódovány i jako vektorové rozdíly. Výsledné slovní vektory proto fungují velmi dobře na úlohách pracujících s analogickými slovy.

5 Vícejazyčný korpus pro CL embeddingové metody

Protože v následující kapitole bude velká většina metod pracovat na specifickém vícejazyčném srovnatelném korpusu, pokusíme se ho nejdříve formálně definovat.

Mějme vícejazyčný korpus D, který je tvořen množinou vícejazyčných dokumentů $d = \{d_1, d_2, ..., d_s\}$, přičemž každý vícejazyčný dokument d_i představuje nějaké téma (např. američtí prezidenti) a je reprezentován v m jazycích $u = \{u_1, u_2, ..., u_m\}$, u_i tedy značí konkrétní jazykovou podobu dokumentu. Může se ale stát, že k některému tématu nebudou k dispozici všechny jazykové reprezentace. Pak ale musí existovat alespoň pro 2 jazyky a ostatní jazykové reprezentace pak budou nahrazeny prázdnými dokumenty.

Vícejazyčný korpus D, získaný z Wikipedie, můžeme za použití VSM považovat za množinu m matic $X = X_1, X_2, ..., X_m$, kde $X_i \in R^{n_i \times s}$ je term-dokument matice pro jazyk i a n_i je velikost slovníku. Sloupec X_i^l pak představuje TFIDF vektor vícejazyčného dokumentu d_l pro jazyk i. N značí celkovou velikost všech slovníků a zároveň tedy udává počet řádků matice X ($X = \sum_{i=1}^{m} n_i$). Vše blíže ilustruje obrázek 5.1.



Obrázek 5.1: Spojení term-dokument matic dílčích jazyků do matice X

6 Cross-linguální podobnost dokumentů

V této kapitole si představíme několik základních typů přístupů jak lze počítat podobnost dokumentů napříč různými jazyky, které se vyskytují v souvisejících pracích.

6.1 Překladový a slovníkový přístup

Prvním a zřejmě nejvíce intuitivním přístupem, který se k tomuto účelu nabízí, je využití strojového překladu a následného výpočtu jednojazyčné podobnosti. V dnešní době je volně dostupná řada nástrojů pro překlad textů jako Google Translator, Moses [5] a další.

Tento přístup však s sebou nese spoustu úskalí. V první řadě vlastně řeší těžší problém než je potřeba vyřešit a kromě toho musíme mít i velké množství trénovacích dat (množin přeložených vět), aby systém mohl správně fungovat. Natrénování volně dostupného nástroje Moses pro jazyky s nedostatečnými jazykovými zdroji je proto velmi komplikované. U online služeb jako Google Translate pak zpravidla bývá jejich API limitováno pro volné použití (např. počtem požadavků za minutu) a při plnohodnotném využití již bývá placené.

Zajímavým článkem pro naši práci je například [13], ve kterém autoři porovnávají testované dokumenty za využití slovníků, konkrétně Eurovoc tezauru.

Eurovoc

Velmi oblíbenou technikou měření podobnosti dokumentů napříč jazyky jsou systémy založené na Eurovoc tezauru, ať už v podobě hlavního či částečného zdroje příznaků. Eurovoc je vícejazyčný polytematický tezaurus, který byl vytvořen Evropským parlamentem společně s Evropskou unií. Je zaměřen především na oblasti práva a legislativy Evropské unie. Obsahuje celkem 8 úrovní hierarchicky organizovaných deskriptorů (přibližně asi 6000), které jsou vzájemně přeloženy do 22 různých jazyků EU.

Práce [17] částečně rozšiřuje samotné deskriptory o tzv. asociovaná slova, tedy slova, která s daným deskriptor/tématem úzce souvisí, a pro každé takové slovo vypočítali i míru jeho důležitosti pro daný deskriptor. Jedno

slovo se může vyskytovat ve více různých deskriptorech, pro každý pak ale velmi pravděpodobně bude mít jinou míru důležitosti. Např. pro anglický deskriptor s názvem *Protection and Minorities* budou mezi asociovaná slova patřit: *racism*, *xenophobia*, *minority*. Na následující ukázce jsou zachyceny asociovaná slova i s reálnými koeficienty jak pro češtinu, tak pro angličtinu:

CS: Příprava při zaměstnání

```
školené 0.5708
vzdělávacího 0.5003
vzdělávání 0.3974
školených 0.3645
```

EN: In-service training

```
trainers 0.7299
underinvest 0.5005
training 0.4654
```

V článku [12] autoři získávají jako hlavní zdroj příznaků 100 nejvíce relevantních Eurovoc deskriptorů pro daný dokument (50% příznaků), dále zeměpisné reference a názvy (30%) a jako poslední zdroj jmenné entity - názvy osob, organizací, atd. (20%) a tyto příznaky pak vyhodnocují pomocí kosinové podobnosti.

6.2 Jednojazyčný přístup

Další kategorií je překvapivě jednojazyčný přístup. Tedy přístup, který zachází s dokumenty napsanými v různých jazycích, podobně jako by byly napsány pouze v jednom jazyce. Základní myšlenkou je fakt, že mnoho jmenných entit (např. Trump) či etymologicky příbuzných slov (např. tsunami) jsou napsány stejně nebo podobně i v mnoha jiných jazycích. V článku [11] autoři používají metodu Cross-Language Character n-Gram Model (CL-CNG), kde reprezentují dokumenty n-gramovými množinami. Další možností může být například použití seznamů jazykově specifických klíčových slov na základě etymologicky příbuzných slov (cognates), podobně jako tomu bylo v práci [13].

Společné entity

Poměrně jednoduchou a přesto efektivní metodou je hledání společných jmenných entit ve zpravodajských článcích. Jedna ze služeb umožňující detekovat a mapovat entity z článků v různých jazycích se nazývá Europe Media Monitor. Nalezené entity pak dále ukládá v rámci RSS souborů s počtem výskytů a pozicí v textu k dalšímu zpracování.

Z nalezených entit v článcích můžeme vytvořit vektory a dále je porovnávat např. kosinovou podobností či vybrat N nejčetnějších entit a následně porovnat jejich procentuální shodu v článcích.

Obecně tato metoda nefunguje vůbec špatně, a to především pokud máme výrazně odlišné články. Pokud však máme velkou část článků, které obsahují podobné entity, ale v zásadě se dost výrazně liší svým obsahem i tématem, úspěšnost metody výrazně klesá, protože tyto články často bývají zařazeny do stejného clusteru. Potřebovali bychom nějak docílit jemnějšího dělení. Další a největší problém nastává, pokud v článku nenalezneme vůbec žádné entity. Pak nemáme žádnou možnost, jak najít podobné články.

6.3 Faktorizace matic

Třetím způsobem počítání vícejazyčné podobnosti dokumentů je využití některé z metod založených na faktorizaci matic. Tyto modely zahrnují například: faktorizaci nezáporných matic, Cross-linguální latentní sémantickou analýzu (CL-LSA), Kanonickou korelační analýzu (CCA), Cross-linguální explicitní sémantickou analýzu (CL-ESA) nebo Orientovanou analýzu hlavních komponent (OPCA). Všechny tyto metody byly zmíněny v práci [15] a kromě OPCA budou dále popsány i v naší práci v následujících sekcích. Metoda OPCA se ukazuje jako ne příliš vhodná pro velká data vzhledem ke své kvadratické časové složitosti, prostorové závislosti a nutnosti zarovnání slovníků všech testovaných jazyků na stejnou velikost.

6.3.1 K-Means

Algoritmus K-means je zřejmě nejznámější a nejpoužívanější algoritmus pro shlukování. Tento pojem poprvé použil James MacQueen roku 1967 a vychází z myšlenky, že shlukované objekty lze chápat jako body v eukleidovském prostoru a že počet shluků k je předem dán. Každý shluk je definován svým centroidem, který také představuje bod v prostoru. Objekty jsou vždy přiřazeny k nejbližšímu centroidu. Ty jsou z počátku náhodně zvoleny a následně se iterativně přepočítávají tak, aby odpovídaly těžišti shluků bodů.

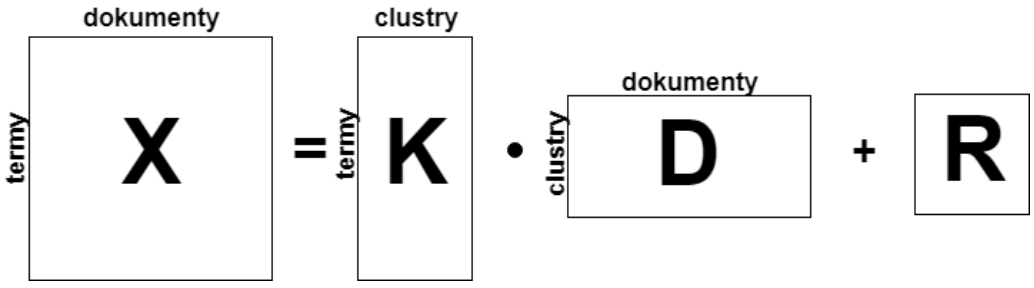
K-means nachází své využití také při počítání podobnosti dokumentů napříč různými jazyky. Jeho hlavním úkolem je nalézt tzv. množinu zarovnaných bází, které jsou pak dále použity k samotnému výpočtu podobnosti např. pomocí kosinové podobnosti.

Mějme dánu množinu dokumentů $D = \{d_1, d_2, ..., d_n\}$, kde každý pozorovaný objekt je d-dimenzionální reálný vektor, účelem k-means je rozdělit n pozorovaných dokumentů do $k(\leq n)$ shluků $S = \{s_1, s_2, ..., s_k\}$ tak, aby se minimalizoval součet čtverců uvnitř shluků. Tento vztah udává následující vzorec:

$$\arg\min_S = \sum_{i}^{k} \sum_{x \in S_i} \|x - \mu\|^2 = \arg\min_S \sum_{i=1}^{k} |S_i| Var S_i \tag{6.1}$$

Postup je takový, že nejprve spojíme všechny term-dokument matice získané pro jednotlivé jazyky (X_1 až X_m) do společné matice X (viz dřívější obrázek 5.1) tak, aby sloupce odpovídaly dokumentům a každý dokument pak bude reprezentován vektorem, tvořeným tf-idf hodnotami v rámci všech jazyků. Řádek X_1 tedy bude např. odpovídat term-dokument matici o velikosti slovníku n_1 získané ze všech anglických dokumentů a X_1^1 pak představuje TF-IDF vektor.

Následně spustíme k-means, abychom získali matici nových bází K, jejíž sloupce jsou tvořeny vektory k-means centroidů. Viz obrázek 6.1



Obrázek 6.1: Rozklad matice X po k-means algoritmu

Matici K můžeme zpětně vertikálně rozdělit na m dílčích matic podle velikostí slovníků konkrétních jazyků n_i:

$$K = [K_1^T ... K_m^T]^T \tag{6.2}$$

Každá K_i matice obsahuje vektorové báze a může být použita k přemapování bodů z prostoru R^{n_i} do k-dimenzionálního prostoru, kde k je počet clusterů. Nové souřadnice vektoru $x \in R^{n_i}$ pak můžeme vypočítat podle následujícího vztahu:

$$(K_i^T K_i)^{-1} K_i^T x_i \tag{6.3}$$

K výpočtu podobnosti dokumentů mezi jazyky i a j používáme jazykově nezávislý prostor vzniklý pseudoinverzí matic bází P_i:

$$P_i = (K_i^T K_i)^{-1} K_i \tag{6.4}$$

a následně vynásobíme matice takto a získáme dříve zmiňovanou transformační matici S_{ij}:

$$S_{ij} = P_i^T P_j \tag{6.5}$$

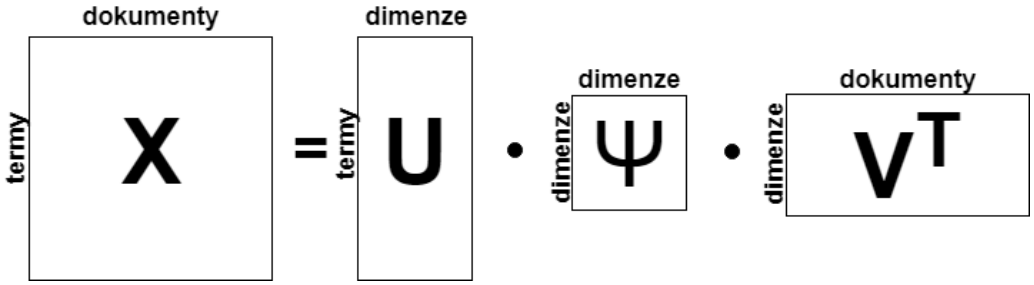
Přičemž předpokládáme, že vektory centroidů tvořící matici K jsou lineárně nezávislé.

6.3.2 CL-LSA

Další metodou, kterou lze pro naši práci použít, je metoda zvaná Cross-lingualní Latentní sémantická analýza [3]. Jak již z názvu vyplývá, jedná se o rozšíření LSA z kapitoly 4.1 pro více než jeden jazyk. Samotný algoritmus je velmi podobný k-means z předchozí sekce. Tedy stejně jako v předchozím případě jsou nejprve zřetězeny všechny term-dokument matice pro konkrétní jazyky do společné matice a v dalším kroku se opět snažíme získat redukovanou matici bází (v případě CL-LSA pomocí SVD). Konečná podobnost dokumentů může být opět vypočítána pomocí kosinové podobnosti.

Na původní obdélníkovou matici $X_{t,d}$ použijeme singulární rozklad (viz vzorec 4.2), který ji aproximuje na součin tří dílčích matic. Tento vztah znázorňuje obrázek 6.2.

Kde $U_{t,k}$ tvoří matici bází, v níž každý prvek určuje jak silně daný term souvisí s danou sémantickou dimenzí. $\Psi_{k,k}$ je diagonální matice obsahující prvních k singulárních hodnot ($k < s$), přičemž každá singulární hodnota odráží, jak je důležitá konkrétní sémantická dimenze k. $V_{k,s}$ je dokumentová matice v sloupcově ortonormální formě, kde její prvky symbolizují, jak je konkrétní dokument podstatný pro téma reprezentované sémantickou dimenzi.



Obrázek 6.2: Singulární rozklad matice X

Dále pokračujeme analogicky jako u předchozí metody a rozdělíme vzniklou matici U vertikálně do m bloků podle velikostí slovníků n_i konkrétních jazyků:

$$U = [U_1^T ... U_m^T]^T \tag{6.6}$$

,stejně jako v případě k-means podobnost dokumentů mezi jazyky i a j vypočítáme pseudoinverzní matice takto:

$$P_i = (U_i^T U_i)^{-1} U_i \tag{6.7}$$

a opět obě vzniklé matice vynásobíme a získáme tak matici S_{ij}:

$$S_{ij} = P_i^T P_j \tag{6.8}$$

6.3.3 CCA

Kanonická korelační analýza (v originále Canonical correlation analysis - CCA) [18] je technika dimenzionální redukce podobná již zmiňované analýze hlavních komponent (PCA). Oproti PCA tato metoda dále předpokládá, že data vzniklá z příznakových vektorů jsou ze dvou různých zdrojů (pohledů), které mezi sebou sdílejí některé informace. Příkladem mohou být například dvojjazyčné kolekce dokumentů či obrázky a jejich popisky.

Namísto hledání lineární kombinace příznaků, které maximalizují rozptyl (u PCA), hledáme v tomto případě lineární kombinaci příznakových vektorů z prvního zdroje (pohledu) a lineární kombinaci z druhého zdroje, které mezi sebou maximálně korelují.

Pokud se vrátíme zpět k naši term-dokument matici X z kapitoly 5, sloupce matice X_i představují pozorované vektory, přičemž $X_i \in R^{n_i}$. Naším cílem je nalézt takové dva váhové vektory $h_i \in R^{n_i}$ a $h_j \in R^{n_j}$, aby náhodné

proměnné $h_i^T.X_i$ a $h_j^T.X_j$ maximálně korelovaly (h_i a h_j se používají k mapování náhodných vektorů na náhodné proměnné výpočtem vážených součtů vektorových složek).

Nechť $\rho(x, y)$ představuje korelační koeficient mezi 2 pozorovanými vektory x a y. Pokud použijeme předchozí notaci matic X_i a X_j (pro zjednodušení předpokládáme, že nechybí žádná data), můžeme vztah formulovat následujícím optimalizačním problémem:

$$\underset{h_i \in R^{n_i}, h_j \in R^{n_j}}{maximize}\, \rho(h_i^T X_i, h_j^T X_j) = \frac{h_i^T V_{i,j} h_j}{\sqrt{h_i^T V_{i,i} h_i}\sqrt{h_j^T V_{j,j} h_j}} \tag{6.9}$$

Kde:

- $V_{i,i}$ a $V_{j,j}$ jsou empirické odhady pro rozptyly X_i a X_j
- $V_{i,j}$ odhad kovarianční matice

Za předpokladu, že pozorované vektory jsou vycentrovány, mohou být matice V vypočítány podle následujícího vzorce:

$$V_{i,j} = \frac{1}{n-1} X_i X_j^T \tag{6.10}$$

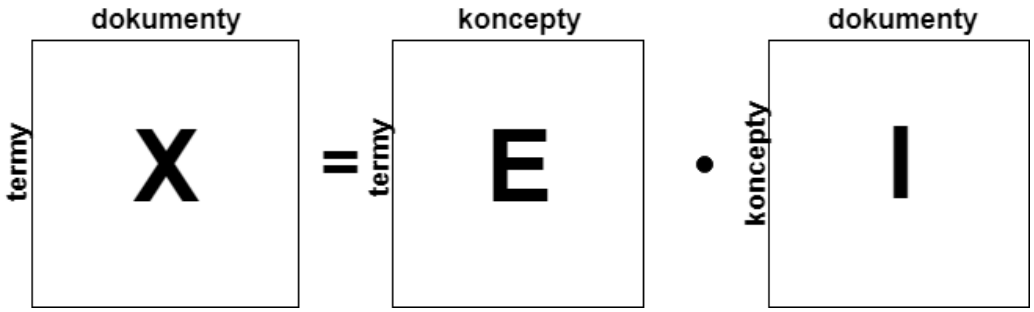
Optimalizační problém tedy může být redukován na problém vlastních čísel a zahrnuje převrácení (invertování) rozptylových matic $V_{i,i}$ a $V_{j,j}$. Pokud matice nejsou invertibilní, lze využít regularizační techniku nahrazením $V_{i,i}$ s $(1-\kappa)V_{i,i}+\kappa I$, kde $\kappa \in [0, 1]$ je tzv. regularizační koeficient a I je jednotková matice (analogicky pak $V_{j,j}$).

Tato metoda je v originální podobě použitelná pouze pro dva různé jazyky. Vznikly ale metody, které rozšiřují tuto metodu tak, aby ji šlo použít i na více různých jazyků zároveň (CL-CCA).

6.3.4 CL-ESA

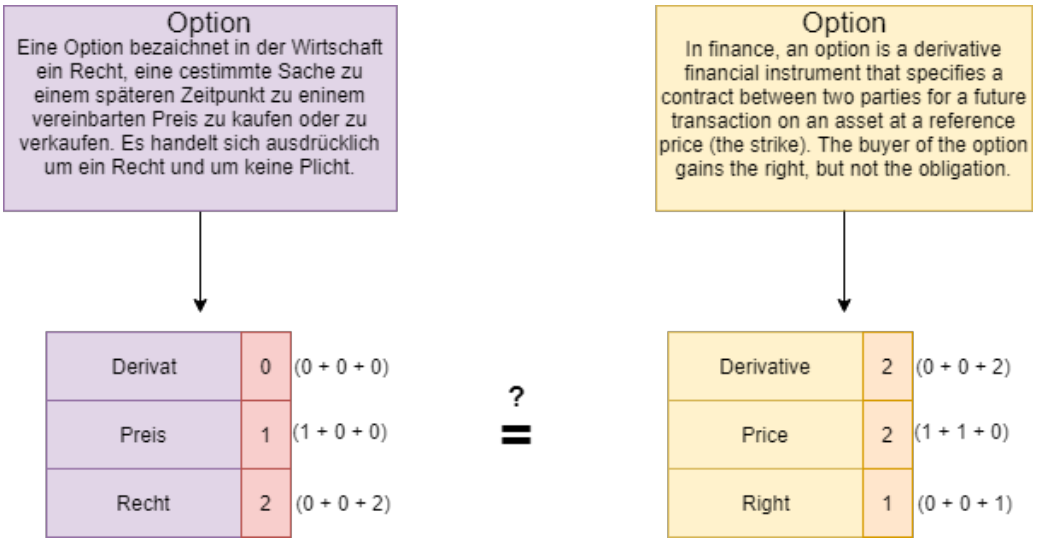
Cross-Linguální Explicitní sémantická analýza je pouze zobecněním ESA z kapitoly 4.2 pro více jazyků. Stejně jako u jednojazyčné ESA i zde je dána množina konceptů $E = \{e_1, e_2, ...e_r\}$ o předem definované velikosti r a kromě toho i množina jazyků $L = \{L_1, L_2, ..., L_m\}$ a vícejazyčná kolekce dokumentů $D = \{D_1, D_2, ..., D_r\}$, kde každá matice D_i obsahuje pouze dokumenty v jazyce L_i.

V Explicitní sémantické analýze jsou testované dokumenty reprezentovány vektory z koncepčního prostoru, kde každá dimenze představuje nějaký textový popis. Prostor tvořený koncepty u ESA pak přímo odpovídá prostoru tvořenému dokumenty. Tento vztah zachycuje následující obrázek.



Obrázek 6.3: Matice X jako Koncepční matice E

Pokud se vrátíme zpět k příkladu z kapitoly 4.2, kde jsme získali pro anglický dokument s názvem *Option* příznakový vektor reprezentující asociační sílu dokumentu k jednotlivým konceptům. Nyní tento příklad rozšíříme tak, že přidáme ještě jeden dokument, který ale tentokrát bude v němčině. Analogicky opět získáme asociační sílu pro dané koncepty a výsledné vektory pak můžeme porovnat. Vše znázorňuje následující obrázek:



Obrázek 6.4: Porovnání dokumentů v různých jazycích pomocí ESA

6.4 Pravděpodobnostní modely

Poslední možností, jak počítat podobnost dokumentů napříč různými jazyky, je pomocí pravděpodobnostních grafických modelů. Tyto modely zahrnují metody jako například: Joint Probabilistic Latent Semantic Analysis (JPLSA) [10] nebo Probabilistic Cross-Lingual LSA (PCLLSA) [19].

Tyto metody v zásadě popisují vícejazyčné kolekce dokumentů jako vzorky z generativních pravděpodobnostních modelů s variacemi předpokladů na strukturu modelu. Témata reprezentují latentní proměnné používající se k vytváření pozorovaných slov, které jsou specifické pro konkrétní jazyk. Odhad parametrů je inferenční problém, který je typicky obecně nezvladatelný a obvykle se řeší aproximacemi. Většina řešení je pak založena na Gibbsově vzorkování nebo Variantní Inferenci, jejichž implementace je složitá a ve většině případů vyžaduje i zkušeného odborníka. Z tohoto důvodu nebudou tyto metody v naší práci použity.

7 Způsoby vyhodnocení shlukování

Vyhodnocení shlukování se dá považovat za podobně těžké jako samotné shlukování. Velký podíl na tom má fakt, že shlukování patří mezi metody strojového učení bez učitele a zároveň téměř vždy může existovat více správných řešení. Proto je těžké nějakým způsobem zhodnotit kvalitu (správnost) výstupních clusterů testovaných metod.

Cíl je však jasný, snažíme se, aby stejné objekty (tzv. observation), byly přiřazeny do stejných clusterů a naopak aby rozdílné objekty byly přiřazeny do clusterů jiných.

Evaluační metriky lze rozdělit do několika skupin, přičemž základní dělení je na evaluaci pomocí interních a externích kritérií. Dalšími skupinami, které však déle nebudeme v této práci podrobněji rozebírat, jsou manuální kritéria (vyhodnocována lidským expertem) a evaluace za pomoci informačních kritérií (použití statistického modelu).

Žádný způsob vyhodnocování úspěšnosti však není dokonalý a dokonce i v případě manuální evaluace se musíme spoléhat na subjektivní názor lidského experta, který se může lišit s názorem jiného experta.

7.1 Evaluace pomocí interních kritérií

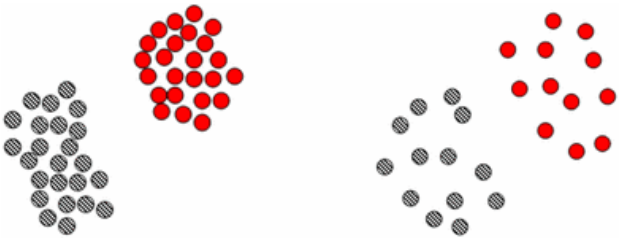
Interní evaluace znamená, že k určení metriky používáme pouze výsledné shluky, které mezi sebou porovnáváme (např. strukturou či vzájemnými vztahy) a nemáme k dispozici žádné další externí informace. Nejlepšího skóre při použití tohoto kritéria obvykle dosahují algoritmy, které produkují clustery s vysokou podobností uvnitř clusteru (kompaktností) a s velkými rozdíly oproti ostatním clusterům (separace).

Problémem však je, že vysoké skóre měřené pomocí interních kritérií nutně nemusí znamenat, že náš systém úspěšně shlukuje. Tento typ metod se nejlépe hodí pro situace, kdy chceme získat prvotní pohled na to, že jeden algoritmus funguje lépe než druhý, ale neznamená to však, že produkuje více správných výsledků.

7.1.1 Základní koncepty

Kompaktnost

Kompaktnost měří jak blízce jsou objekty v clusteru seskupeny. Standardně je založena na vzdálenostech mezi body v clusteru. Velmi oblíbeným způsobem výpočtu kompaktnosti je např. odchylka. Tedy vlastně počítáme průměrnou vzdálenost ke střední hodnotě. Malá odchylka značí velkou kompaktnost a naopak (viz obrázek 7.1).



Obrázek 7.1: Kompaktnost - vlevo znázorněna velká kompaktnost a vpravo nízká

Nejjednodušší metrika, která měří pouze kompaktnost se nazývá Root-mean-square standard deviation (RMSSTD) a je definována takto:

$$RMSSTD = \sqrt{\frac{\sum_i \sum_{x \in \Omega_i} \|x - \omega_{ci}\|^2}{DIM \sum_i (n_i - 1)}} \tag{7.1}$$

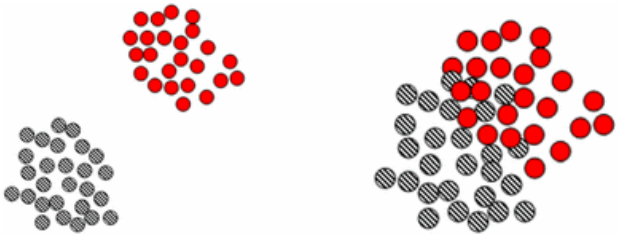
Kde:

- DIM - počet dimenzí vstupního datasetu
- Ω_i - i-tý cluster
- ω_{ci} - střed clusteru i
- n_i - počet bodů v Ω_i

Separace

Separace měří jak jsou nalezené clustery od sebe odlišné (obrázek 7.2). Čím jsou clustery odlišnější, tím je dané rozdělení na shluky pro uživatele srozumitelnější a clustery pak odpovídají jedinečnému vzoru.

Podobně jako u kompaktnosti se k výpočtu separace používají distanční metriky. Například vzdálenost mezi středy clusterů či 2 různými body v clusterech.



Obrázek 7.2: Separace - clustery vlevo mají vysokou míru separace zatímco clustery vpravo mají naopak nízkou

Jedna ze základních metrik, která měří pouze separaci se nazývá R-squared (RS) a vypočítává se podle následujícího vzorce:

$$RS = \frac{\sum_{x \in D} \|x - d_c\|^2 - \sum_i \sum_{x \in \Omega_i} \|x_i - \omega_{ci}\|^2}{\sum_{x \in D} \|x - d_c\|^2} \tag{7.2}$$

Kde:

- D - vstupní dataset
- Ω_i - i-tý cluster
- $\omega_c i$ - střed clusteru
- d_c - střed celého datasetu

Další metody

- Davies–Bouldin index
- Dunn index

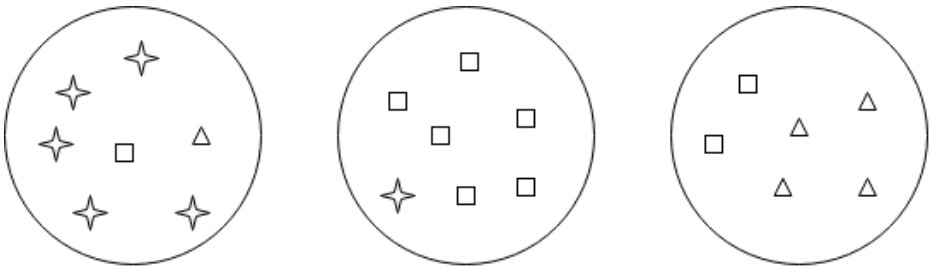
7.2 Evaluace pomocí externích kritérií

Při evaluaci pomocí externích kritérií jsou výsledky shlukování vyhodnoceny na základě dat, která nebyla použita pro shlukování samotné, ale tvoří je většinou množina předem klasifikovaných tříd, kterou často vytvářejí odborníci (tzv. ground truth data). Tento typ metod měří jak blízce jsou si podobné výsledné shluky k odborníky předem připraveným shlukům.

Problémem u externích kritérií je obvykle to, že nám často chybí zmiňovaná ground truth data a také nám může vadit, že popisují výsledné shlukování pouze z jednoho úhlu pohledu. V praxi však může nastat to, že naše

výsledné shlukování bude ve skutečnosti lepší než ground truth data, ale konkrétní evaluační metrika to správně nevyhodnotí.

Následující obrázek, na kterém je zachycen ilustrativní výsledek shlukování, nám poslouží k názorné ukázce počítání jednotlivých externích metrik.



Obrázek 7.3: Ilustrační příklad pro vyhodnocení výsledných shluků

7.2.1 Purity

Jedním z častých a oblíbených evaluačních metrik je tzv. *purity*. Ta udává jak čisté (pure) výsledné clustery jsou. Purity spočítáme tak, že pro každý cluster vypočítáme počet objektů (observation) z nejčetnější třídy a tento součet vydělíme celkovým počtem objektů.Formálně pak:

$$purity(\Omega, C) = \sum_{i=1}^{k} \frac{N_i}{N} max_j \frac{N_{ij}}{N} \tag{7.3}$$

a po úpravě pak získáme tento vztah:

$$purity(\Omega, C) = \frac{1}{N} \sum_{i=1}^{k} max_j N_{ij} \tag{7.4}$$

Kde:

- $\Omega = \{\omega_1, ..., \omega_k\}$ - množina clusterů
- $C = \{c_1, ..., c_j\}$ - množina tříd
- N - celkový počet objektů
- N_{ij} - počet objektů z clusteru i, které náleží třídě j
- $N_i = \sum_{j=1}^{c} N_{ij}$ - celkový počet objektů v clusteru i

Vysoké purity lze jednoduše dosáhnout tak, že pro každý objekt vytvoříme svou vlastní třídu. Získáme tak pro každý cluster i celkově purity = 1, z toho vyplývá, že tato metrika nijak nepenalizuje počet clusterů a nemůže

být proto použita na určení kvality shlukování v závislosti na počtu clusterů. Metrika, která nám pak tuto vlastnost umožní je např. normalized mutual information (NMI).

Názorný výpočet

Nyní provedeme výpočet purity pro náš ilustrační příklad :

$$purity(\Omega, C) = \frac{1}{20}(5 + 6 + 4) = \frac{15}{20} = 0.75 \tag{7.5}$$

7.2.2 Rand Index

Další oblíbenou metodou pro evaluaci shlukování je rand index. Rand index udává procentuální míru algoritmem správně zařazených objektů. Formálně ho můžeme vypočítat pomocí tohoto vzorce:

$$RI(\Omega, C) = \frac{TP + TN}{TP + FP + TN + FN} \tag{7.6}$$

Kde:

- TP (true positive) - počet správně přijatých objektů
- TN (true negative) - počet správně nepřijatých objektů
- FP (false positive) - počet objektů, které do clusteru nebyly přijaty, ale měly být (chyba 1. typu)
- FN (false negative) - počet objektů, které byly do clusteru přijaty, ale měly být odmítnuty (chyba 2. typu)

Algoritmus tedy penalizuje oba typy chyb. Jak falešně pozitivní tak falešně negativní výsledky.

Názorný výpočet

Pokud se opět vrátíme k ilustračnímu příkladu z úvodu kapitoly, rand index bychom vypočítali následovně. Nejprve si spočítáme celkový počet dvojic napříč všemi objekty.

$$TOTAL = TP + FP + TN + FN = \binom{N}{2} = \binom{20}{2} = 190 \tag{7.7}$$

Dalším krokem je výpočet součtu TP a FP. Tento součet reprezentuje počet dvojic objektů ve stejném clusteru bez ohledu na zařazení do třídy:

$$TP + FP = \binom{7}{2} + \binom{7}{2} + \binom{6}{2} = 21 + 21 + 15 = 57 \tag{7.8}$$

Součet TP a FN udává počet dvojic objektů na základě tříd, ale bez zařazení do clusterů. Pro náš příklad tedy 6 hvězd, 9 čtverců a 5 trojúhelníků.

$$TP + FN = \binom{6}{2} + \binom{9}{2} + \binom{5}{2} = 15 + 36 + 10 = 61 \tag{7.9}$$

TP je počet správně zařazených dvojic objektů o stejné třídě do stejných clusterů. Konkrétně pak 5 dvojic hvězd z 1. clusteru, dále 6 dvojic čtverců z 2. clusteru a ve 3. clusteru jsou správně 4 dvojice trojúhelníků a 1 dvojice čtverců.

$$TP = \binom{5}{2} + \binom{6}{2} + \binom{4}{2} + \binom{2}{2} = 32 \tag{7.10}$$

Z vypočtených hodnot můžeme vypočítat FN podle jednoduchého vztahu:

$$FN = (TP + FN) - TP = 61 - 32 = 29 \tag{7.11}$$

Stejně tak i FP:

$$FP = (TP + FP) - TP = 57 - 32 = 25 \tag{7.12}$$

Poslední neznámou vypočítáme rozdílem celkového počtu dvojic se součtem ostatních získaných hodnot:

$$TN = TOTAL - (TP + FP + FN) = 190 - (32 + 25 + 29) = 104 \tag{7.13}$$

Z vypočtených údajů můžeme vytvořit tzv. kontingenční tabulku a po dosazení do vzorce 7.6 obdržíme hledaný rand index.

Výsledek:

$$RI(\Omega, C) = \frac{32 + 104}{190} = \frac{136}{190} \approx 0.716 \tag{7.14}$$

7.2.3 F-míra

F-míra (též F-skóre) je míra, která pracuje s oběma typy chyb (FN i FP) a podporuje jejich rozdílné vážení. Samotný výpočet pak udává poměr mezi přesností a úplností. Přesnost (Precision - P) je dána tímto vztahem:

$$P(\Omega, C) = \frac{TP}{TP + FP} \tag{7.15}$$

Úplnost (Recall - R) je definována takto:

$$R(\Omega, C) = \frac{TP}{TP + FN} \tag{7.16}$$

Nyní již máme vše potřebné k výpočtu F_β-míry, kde β je parametr sloužící k vážení úplnosti:

$$F_\beta(\Omega, C) = \frac{(1 + \beta^2)P.R}{\beta^2 P + R} \tag{7.17}$$

Speciálním případem a zároveň velmi oblíbenou metrikou je tzv. F_1 míra, která udává harmonický průměr mezi přesností a úplností.

$$F_1(\Omega, C) = \frac{2PR}{P + R} = \frac{2TP}{2TP + FN + FP} \tag{7.18}$$

Názorný výpočet

Výpočet F_1 míry z našeho ilustračního příkladu bude vypadat takto:

$$F_1(\Omega, C) = \frac{2.32}{2.32 + 29 + 25} = \frac{64}{118} \approx 0.542 \tag{7.19}$$

7.2.4 V-míra

V-míra je míra založená na entropii a explicitně měří jak úspěšně jsou uspokojena kritéria pro homogenitu a celistvost (completeness). V-míra udává totiž jejich harmonický průměr, podobně jako F-míra, která je dána přesností a úplností, a stejně jako F-míra může být také vážena.

Homogenita:
Pokud chceme, aby výsledné shlukování bylo maximálně homogenní, tak každý výstupní cluster musí obsahovat vždy pouze prvky jedné třídy. Homogennost (h) (homogenita) je tedy definována takto:

$$h(\Omega, C) = \begin{cases} 1 & \text{pro } H(C) = 0 \\ 1 - \frac{H(C|\Omega)}{H(C)} & \text{jinak} \end{cases} \tag{7.20}$$

Přičemž $H(C)$ značí entropii a $H(C, \Omega)$ představuje podmíněnou entopii a vypočítají se následovně:

$$H(C, \Omega) = -\sum_{j=1}^{|\Omega|} \sum_{i=1}^{|C|} \frac{a_{ij}}{N} log \frac{a_{ij}}{\sum_{i=1}^{|C|} a_{ij}} \tag{7.21}$$

$$H(C) = -\sum_{i=1}^{|C|} \frac{\sum_{j=1}^{|\Omega|} a_{ij}}{N} log \frac{\sum_{j=1}^{|\Omega|} a_{ij}}{N} \tag{7.22}$$

Kde a_{ij} značí počet prvků z třídy c_i a patří do clusteru ω_j.

Celistvost:

Celistvost je symetrická vlastnost k homogenitě a její maximální hodnotu tedy získáme tak, že všichni příslušníci dané třídy náleží pouze jednomu clusteru a je definována tímto vztahem:

$$c(C,\Omega) = \begin{cases} 1 & \text{pro } H(\Omega) = 0 \\ 1 - \frac{H(\Omega|C)}{H(\Omega)} & \text{jinak} \end{cases} \tag{7.23}$$

Kde $H(\Omega, C)$ a $H(\Omega)$ vypočítáme takto:

$$H(\Omega,C) = -\sum_{i=1}^{|C|}\sum_{j=1}^{|\Omega|}\frac{a_{ij}}{N} log \frac{a_{ij}}{\sum_{j=1}^{|\Omega|} a_{ij}} \tag{7.24}$$

$$H(\Omega) = -\sum_{j=1}^{|\Omega|}\frac{\sum_{i=1}^{|C|} a_{ij}}{N} log \frac{\sum_{i=1}^{|C|} a_{ij}}{N} \tag{7.25}$$

Nyní již máme vše potřebné, abychom mohli sestavit vzorec pro výpočet V-míry, který vypadá takto:

$$V_\beta(\Omega,C) = \frac{(1+\beta)h.c}{\beta h + c} \tag{7.26}$$

Další metody

- Jacard Index
- Dice Index
- Mutual Information
- Normalized Mutual Information

8 Systém pro propojení témat zpravodajských článků

Hlavním cílem této práce je vytvořit systém, který bude schopen hledat podobná témata zpravodajských článků napříč různými jazyky a následně je pak zařadí do společných shluků. V této kapitole si tedy detailněji popíšeme jeho realizaci.

8.1 Návrh systému

Systém pro propojení témat zpravodajských článků napříč různými jazyky bude tvořen třemi druhy metod, které byly zmíněny v kapitole 6. Konkrétně se jedná o tyto druhy metod:

- **Společné entity** - viz kapitola 6.4
- **Eurovoc deskriptory** - viz kapitola 6.1
- **Embeddingové metody**

Všechny tyto metody představují typy příznaků, přičemž mezi vybrané embedingové metody natrénované na korpusu Wikipedie patří tyto metody:

- **K-means** - viz kapitola 6.3.1
- **CL-LSA** - viz kapitola 6.3.2
- **CL-ESA** - viz kapitola 6.3.4

Uživatel si bude moci v konfiguračním souboru vybrat, který typ metod bude chtít použít. Může si vybrat pouze jeden druh příznaků nebo je možné příznaky libovolně kombinovat s různými váhovými koeficienty, které jsou opět nastavitelné v konfiguračním souboru. Vždy však ale může být aktivní pouze jedna embeddingová metoda.

Na základě příznaků získaných ze zvolených metod jsou dokumenty rozděleny do společných clusterů a následně vyhodnoceny pomocí evaluačních metrik.

8.2 Datový model

Vstupní data (tedy zpravodajské články) pro náš systém pochází z aplikace MediaGist [16]. Mediagist je online systém pro cross-lingualní analýzu agregovaných zpráv a komentářů. Je navržen tak, aby novinářům umožnil detekovat a objevovat aktuální témata, která mohou být kontroverzně vnímána v různých zemích. Součástí Mediagist je i crawler, který ukládá data ze zpravodajských serverů do rss souborů. Pro natrénování embeddingových metod využíváme vícejazyčný Wikipedia korpus a pro samotnou evaluaci jsme použili předem připravená ground truth data.

8.2.1 RSS data

RSS (Rich Site Summary) je rodina XML formátů určených pro čtení novinek a poskytuje je k dalšímu zpracování. Cílem RSS formátu byla jednoduchost a snadná pochopitelnost. První verze tohoto formátu byly vyvinuty v roce 1999 společností Netscape.

RSS elementy důležité pro naši práci jsou tyto:

- **channel** - obsahuje základní informace o původu článku (zdroj, jazyk,...)
- **item** - komplexní element představující konkrétní článek
- **guid** - jednoznačný identifikátor článku
- **iso:language** - jazyk, ve kterém je článek napsán
- **emm:entity** - element popisující jmennou entitu, zároveň obsahuje i informaci o počtu výskytů entity v rámci textu
- **emm:text** - obsahuje část textu

Mediagist doplňuje RSS soubory i o elementy představující komentáře a zkoumá zda jsou entity pozitivně, neutrálně či záporně laděné. Pro naši práci však tyto informace používat nebudeme. Pro lepší představu o reálných datech si prohlédněte ukázku na obrázku 8.1.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0" xmlns:emm="http://emm.jrc.it" xmlns:iso="http://www.iso.org/3166">
 <channel>
  <title>Novinky</title>
  <language>cs</language>
  <guid>novinky.cz</guid>
  <pubDate>2017-03-11 20:01:04 CET</pubDate>
  <item emm:id="020c3b67249ab8a9cb7192ca203c8305">
    <title>
      WikiLeaks poskytne technologickým firmám klíč k opravě bezpečnostních děr
    </title>
    <link>
     https://www.novinky.cz/klic-k-oprave-bezpecnostnich-der.html
    </link>
    <description>
    Server WikiLeaks se rozhodl poskytnout technologickým společnostem některé...
    </description>
    <emm:contentType>text/html</emm:contentType>
    <pubDate>2017-03-09 17:50:00 CET</pubDate>
    <iso:language>cs</iso:language>
    <guid>020c3b67249ab8a9cb7192ca203c8305</guid>
    <emm:entity id="264532" type="o" count="1" pos="1532" name="IBM" sentiment="0">
        IBM
    </emm:entity>
    <emm:entity id="197075" type="o" count="1" pos="1615" name="Google" sentiment="0">
        Google
    </emm:entity>
    <emm:entity id="29114" type="o" count="8" pos="236,..." name="CIA" sentiment="0">
        CIA
    </emm:entity>
    <emm:tonality>-3</emm:tonality>
    <commentTonality pos="7" neut="9" neg="1">13</commentTonality>
    <commentSummary comments="17">Nesla by vytahnout slozka na Makrelu?...</commentSummary>
    <emm:text wordCount="547">
     Server WikiLeaks se rozhodl poskytnout technologickým společnostem některé podrobnosti
     o praktikách kybernetické špionáže americké Ústřední zpravodajské služby (CIA)...
    </emm:text>
   </item>
 </channel>
</rss>
```

Obrázek 8.1: Reálný zpravodajský článek v RSS formátu

Statistiky korpusu:

- **Počet článků:** - 1016
- **Celkem slov:** - 597 818
- **Unikátních slov:**

  EN: - 5843

  CZ: - 5243

  DE: - 4016

  FR: - 4816

  IT: - 4368

8.2.2 Wikipedia korpus

Jak jsme již naznačili v úvodu této kapitoly, jako zdrojová data k vytvoření vícejazyčného Wikipedia korpusu jsme využili dostupné dvojjazyčné korpusy dostupné v rámci linguatools[1]. Tyto dvojjazyčné korpusy jsou dokumentově zarovnané a dokumenty v nich obsažené představují reálné články z Wikipedie. Příklad takového korpusu ilustruje obrázek 8.2.

Struktura dvojjazyčných korpusů je poměrně intuitivní. Prvním důležitým elementem je *articlePair* reprezentující stejné téma přeložené do dvou jazyků. Jak si můžete všimnout, každý jazykový pár má své *id*. Bohužel toto *id* není stejné napříč všemi dvojjazyčnými jazykovými korpusy, a pokud chceme sestavit vícejazyčný korpus s více než dvou jazyků, musí to provést jiným způsobem.

Zároveň si můžeme všimnout, že v rámci elementu *content* nalezneme HTML obsah původního Wikipedia článku. To znamená, že je zde i spousta HTML tagů, které je potřeba odstranit před dalším zpracováním. Součástí předzpracování tedy bude jejich odstranění.

Místo *id* jazykového páru budeme využívat vždy název anglického článku (dostupný v rámci atributu *name* elementu *article*). Korpusy tedy mezi sebou budeme propojovat přes angličtinu tzn. pokud budeme chtít získat $cs - en - de$ vícejazyčný korpus, nejprve načteme $cs - en$ korpus a následně $de - en$ korpus. Pro náš výsledný korpus pak přemapujeme id článků tak, aby si odpovídaly všechny jazykové alternativy a zjednodušili jsme si jejich načítání.

[1] https://linguatools.org/tools/corpora/wikipedia-comparable-corpora/

Dalším zdrojem korpusů jsou např. OPUS[2] (Open Parallel korpus) korpusy. OPUS je rostoucí kolekce textů z webu, přičemž hlavním cílem tohoto projektu je sjednotit dostupná online data a doplnit je o další lingvistické anotace. Veškeré zpracování se provádí pouze automaticky a nebyly provedeny žádné manuální opravy. Najdeme zde i paralelní Wikipedia korpusy, které jsou zarovnané na úrovni vět.

Statistiky celého Wikipedia korpusu:

- **Počet vícejazyčných článků:** - 96000
- **Celkem slov:** - 328 524 335

- pozn. celý tento korpus, ale kvůli vysoké paměťové náročnosti nevyužíváme

Statistiky menšího referenčního Wikipedia korpusu:

- **Počet vícejazyčných článků:** - 13200
- **Celkem slov:** - 58 614 224
- **Velikost slovníků:**

  EN: - 15 350

  CZ: - 14 235

  DE: - 13 625

  FR: - 14 120

  IT: - 14 005

Statistiky většího referenčního Wikipedia korpusu:

- **Počet vícejazyčných článků:** - 16000
- **Celkem slov:** - 64 178 331
- **Velikost slovníků:**

  EN: - 23 850

  CZ: - 21 635

  DE: - 20 990

  FR: - 21 200

  IT: - 21 110

[2]http://opus.nlpl.eu/

8.2.3 Evaluační data

Abychom mohli vyhodnotit úspěšnost shlukování potřebujeme pro výpočet evaluačních metrik referenční ground truth data. Tedy data, která byla kolegy z katedry ručně rozdělena do tematicky podobných clusterů. Data jsou dostupná v csv a xlsx formátu a obsahují následující informace:

- **guid** - id zpravodajské článku
- **lang** - jazyk, ve kterém byl článek napsán
- **date** - datum publikace článku
- **title** - titulek
- **description** - krátký popis
- **link** - adresa odkud byl článek stažen
- **cluster** - přiřazené id clusteru

Všechny články pochází z časového horizontu jednoho týdne a byly publikovány v roce 2017 a obsahují celkem 5 různých jazyků.

Statistiky:

- **Počet článků:** - 1016
- **Počet jazyků:** - 5
- **Počet clusterů:** - 70
- **Nejvíce článků v clusteru:** - 97
- **Nejméně článků v clusteru:** - 4

8.2.4 Předzpracování

Před samotným hledáním nejrelevantnějších článků je nejprve nutné, kvůli zvýšení výkonosti metod, vstupní dokumenty i dokumenty z trénovacího korpusu nějakým způsobem předzpracovat. Hlavními důvody jsou úspora paměti a odstranění nežádoucích příznaků, které by vedly ke špatným výsledkům metod. Nyní si představíme hlavní techniky, které využíváme pro předzpracování dokumentů:

- **Převedení na malá písmena** - Převedeme všechna velká písmena na malá, aby nezáleželo, zda je slovo na začátku či uprostřed věty
- **Odstranění speciálních znaků** - Odstraníme speciální znaky jako interpunkční znaménka, speciální symboly atd.
- **Odstranění HTML tagů** - Odstraníme případné pozůstalé tagy jako <b>, <i>, ...
- **Odstranění vícenásobných mezer** - Pokud se v textu vyskytují několikanásobné mezery či bílé znaky (konec řádky, tabulátory) nahradíme je jednoduchou mezerou
- **Odstranění krátkých slov** - Odstraníme slova, která jsou kratší než 3 znaky
- **Odstranění stop slov** - Pro každý jazyk odstraníme nejběžnější množinu slov (stop slova)
- **Stemming** - Můžeme a nemusíme použít stemming - tj. odstranění koncovek slov. V našem programu je konkrétně použit Snowball stemmer, jehož zapnutí se nastavuje přes property.
- **Odstranění termů na základě frekvence výskytu** - Odstraňujeme slova, která se nevyskytují v téměř žádných dokumentech, či slova, která se vyskytují skoro ve všech dokumentech. Hranice pro odstraňování jsou nastavitelné přes property.

```
<?xml version="1.0" encoding="UTF-8"?>
<wikipediaComparable name="cs-en">
...
<articlePair id="15405">
 <article lang="cs" name="Vilniuská univerzita">
  <categories name="Vysoké školy v Litvě"/>
  <content>
   <p>Vilniuská univerzita je jedna z nejstarších univerzit ve
   východní Evropě... </p>
   <h>Dějiny</h>
   <link target="Tovaryšstvo Ježíšovo">Jezuité</link> v roce 1573 z
   <link target="Litva">Litvy</link>
   <h>Úspěšní absolventi univerzity</h>
   <p>Mezi úspěšnými absolventy Vilniuské univerzity byli mnozí literáti
   - polský básník ... </p>
   <h>Architektura stavby</h>
   <p>Za více než čtyři století se univerzita rozšířila.
   Dnes se jí budovy nacházejí nejen ve ...</p>
   <p>Knihovna univerzity byla založena roku <link target="1580">1580</link>.
   Skládá se z ...</p>
   <p><h>Současnost</h>
   <p>Dnes se v Sále Smugleviče nachází stálá expozice unikátních rukopisů.</p>
   <p><h>Reference</h>
  </content>
 </article>
 <article lang="en" name="Vilnius University">
  <categories name="Universities in Lithuania|..."/>
  <content>
  <p>Vilnius University was founded in 1579 as the
  <link target="Jesuit">Jesuit</link> Academy ...</p>
  <p>Following <link target="Soviet invasion of Poland">
  Soviet invasion of Poland</link> ... </p>
  <p><h>History</h></p>
  <p><h>Changes of the name</h></p>
  <p>The university has been known by many names during its history. ...</p>
  <p>1579–1782: Alma Academia et Universitas Vilnensis Societatis Iesu... </p>
  <p>1944–1955: Vilnius State University</p>
  <p>1955–1990: Vilnius State University of
  <link target="Vincas Kapsukas">Vincas Kapsukas</link></p>
  <p>Two of the faculties were turned into separate schools:
  the Medical and Surgical Academy ...</p>
  <p><h>1918-1939</h>
  <p>Lithuania declared its independence in February 1918.
  The university, with the rest of Vilnius ...</p>
  <p>The university quickly recovered and gained international prestige,
  largely because of the ...</p>
  </content>
 </article>
</articlePair>
...
</wikipediaComparable>
```

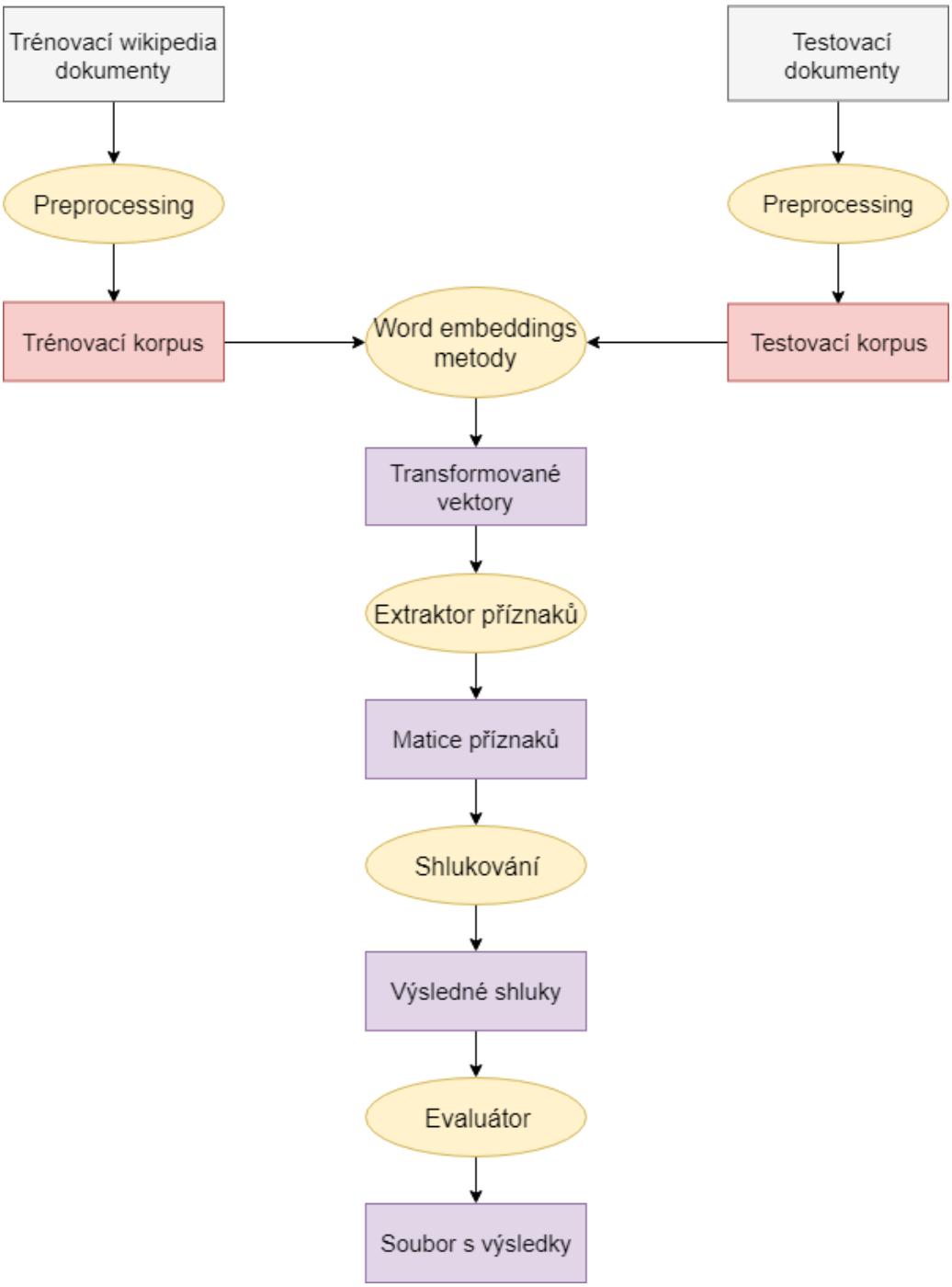
Obrázek 8.2: Příklad tématu v angličtině a češtině obsažený v originálním Wikipedia korpusu

8.3 Architektura

Vzhledem k faktu, že náš program především načítá a zpracovává data, nemusí být architektura systému nijak složitá. Náš systém se skládá z několika základních komponent, kde každá komponenta intuitivně obstarává svou vlastní funkcionalitu. Na tyto komponenty se nyní podrobněji podíváme a pro celkový přehled pak asi nejlépe poslouží data flow diagram z obrázku 8.3.

Komponenty

- **Načítání souborů** - Nejprve je nutné načíst všechny trénovací soubory a soubory se zpravodajskými články na výsledné testování
- **Preprocessing** - V další fázi musí články předzpracovat tak, aby jsme s nimi mohli lépe pracovat (tzn. odstraníme krátké termy, stop slova, převod na malá písmena,...)
- **Vytvoření korpusů** - Z článků vytvoříme korpus (pro vážení použijeme TF-IDF)
- **Embeddingové metody** - V této fázi natrénujeme embeddingové metody zvolené uživatelem a z testovaných zpravodajských článků následně získáme vektory transformované do společného prostoru.
- **Extraktor příznaků** - Transformované vektory nyní můžeme mezi sebou porovnávat a vypočítat podobnostní vektory, které spojíme do matice podobností
- **Clusterer** - Předposlední komponentou je clusterer, jehož úkolem je rozdělit testovací množinu do výsledných clusterů
- **Evaluátor výsledků** - Nakonec již nám zbývá pouze získané výsledky vyhodnotit pomocí evaluátoru, jehož výstupem je soubor porovnávající získané shluky oproti shlukům referenčním a který zároveň obsahuje i některé spočítané evaluační metriky.



Obrázek 8.3: Data flow diagram aplikace

8.4 Implementace

8.4.1 Volba programovacího jazyka

Před samotným vývojem aplikace, je vždy nutné zvolit vhodný programovací jazyk, který bude splňovat naše potřeby. Architektonický návrh aplikace primárně počítá pouze s rozhraním v podobě příkazového řádku (CLI), ale při dostatku času může být implementováno i grafické uživatelské rozhraní (GUI). Vývoj v jazyce měl být pokud možno rychlý a mělo by zde být dostatek knihoven pro NLP. Těmto požadavkům nejvíce vyhovují následující programovací jazyky.

Python

Python je vysokoúrovňový dynamicky interpretovaný programovací jazyk. Python umožňuje používat jak procedurální, tak objektově orientovaný přístup. K jeho hlavním přednostem patří jeho jednoduchá a čistá syntaxe, velmi rychlý vývoj programů a poměrně rychlý běh (zdaleka však nedosahuje rychlosti C/C++). Zároveň v současné době nabízí spousta knihoven pro vědecké výpočty.

Matlab

Matlab je komerční interaktivní programové prostředí a skriptovací jazyk. Obsahuje velké množství nástrojů pro vědecké výpočty, analýzu výsledků. Bohužel se jedná o placený software a licence nepatří mezi nejlevnější.

Java

Java je objektově orientovaný programovací jazyk. Jde o v současnosti nejpoužívanější programovací jazyk. K nejsilnějším stránkám tohoto jazyka patří jednoduchost, bezpečnost a v neposlední řadě platformová nezávislost. Zároveň poskytuje široké spektrum knihoven pro zpracování přirozeného jazyka a matematické výpočty.

Vzhledem k dosavadním zkušenostem a povaze této práce, byla po průzkumu dostupných technologií nakonec vybrána právě Java jako nejvhodnější programovací jazyk k realizaci diplomové práce.

8.4.2 Komponenta pro práci se soubory

Jelikož hlavní potřebou našeho systému je práce se soubory, bylo nutné v prvotní fázi tuto funkčnost řádně implementovat. Jak již bylo zmíněno v předchozích sekcích ,hlavními formáty, které bylo potřeba načítat, byly formáty **xml**, **rss**, **csv**, **txt**, konfigurační **properties** soubory a také soubory v serializované podobě sloužící zejména pro uložení Wikipedia korpusů či eurovoc deskriptorů.

Kromě již zmíněného načítání musí systém podporovat i ukládání a úpravu různých druhů souborů, operovat s konfiguračními soubory či vytvářet výstupní soubory.

Nyní se podíváme na nejdůležitější rozhraní spadající do vrstvy zajišťující práci se soubory a velmi stručně si je popíšeme.

Nejdůležitější rozhraní:

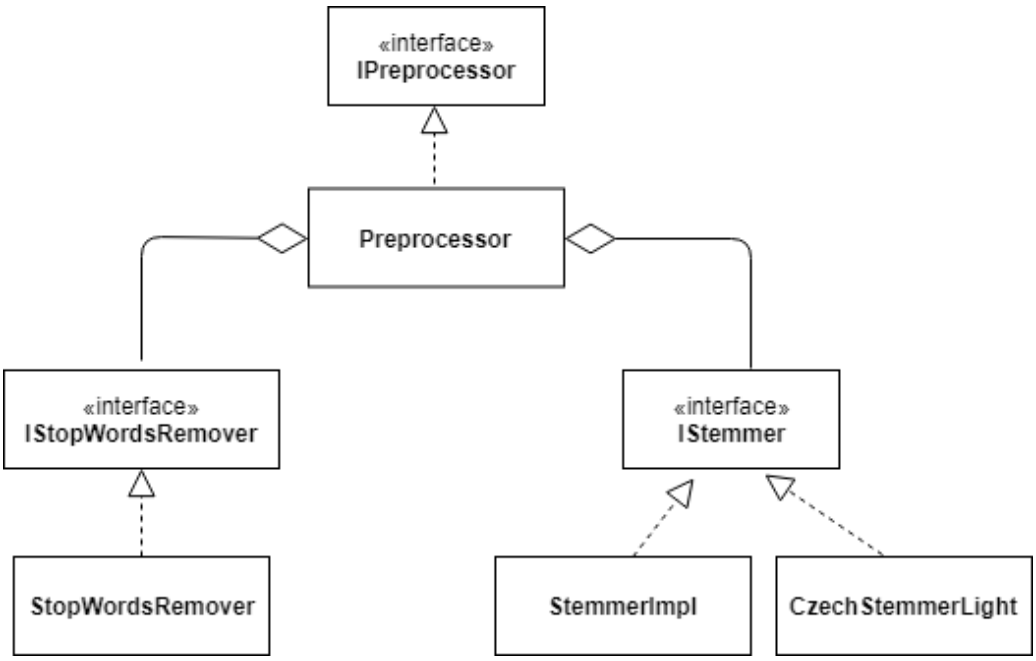
- **IFileLoader** - Rozhraní pro načítání textových a xml souborů, používá se zejména při načítání Eurovoc deskriptorů.
- **INewsLoader** - Rozhraní pro načítání testovacích zpravodajských článků v rss formátu
- **IWikiCorpusLoader** - Rozhraní pro načítání trénovacího Wikipedia korpusu.
- **IObjectFileHandler** - Rozhraní pro ukládání objektů do jejich serializované podoby a zároveň jejich zpětné načítání.
- **IPropertiesHandler** - Rozhraní pro vytváření konfiguračních souborů a jejich načítání.
- **IResultsWriter** - Rozhraní sloužící ke generování výstupních souborů s výsledky obsahující vypočítané metriky.

Použité externí knihovny:

- **jaxb** - Velmi známá knihovna pro převod xml do Javy a naopak - v aplikaci využívána zejména při načítání zpravodajských článků.
- **dom4j** - Opět knihovna pracující zejména s xml soubory, dokáže využívat i pokročilejší funkce jako XPath výrazy - v naší práci se používá pro načítání Eurovoc deskriptorů.

8.4.3 Komponenta pro předzpracování

Nyní navážeme na sekci 8.2.4, kde jsme hovořili o důležitých technikách sloužící k předzpracování trénovacích i testovacích dat. V této sekci si ukážeme, jak jsme vrstvu poskytující tuto funkcionalitu implementovali. K prvotnímu pohledu nám asi nejlépe poslouží UML diagram z obrázk 8.4.



Obrázek 8.4: UML diagram předzpracování

Stejně jako v předchozí sekci i zde si uvedeme nejpodstatnější rozhraní, jež výsledné třídy provádějící předzpracování musí implementovat.

Nejdůležitější rozhraní:

- **IStopWordsRemover** - rozhraní podporující odstranění stop slov ze vstupního textu - stop slova jsou vždy brána pro konkrétní jazyk a jsou načítána ze souborů
- **IStemmer** - rozhraní poskytující metodu pro provádění stemmování, tedy odstranění koncovek slova. Pro všechny jazyky kromě češtiny (tedy en, de, fr, it) používáme knihovnu Snowball, která obsahuje jazykově specifické Stemmery. Pro češtinu bohužel Snowball stemmer dostupný není, a proto jsme použili volně dostupnou implementaci ve třídě CzechStemmerLight

- **IPreprocessor** - rozhraní zaštiťující celkové předzpracování textu. Výsledný Stemmer pak používá obě předchozí rozhraní a poskytuje případy užití, které byly zmíněny v sekci 8.4.

8.4.4 Komponenta pro tvorbu korpusu

Po tom, co načteme zpravodajské články či Wikipedia témata a následně je předzpracujeme, je musíme převést do příslušné interní reprezentace (vybrali jsme BoW), a jelikož neoperujeme pouze s jedním jazykem, budeme vytvářet BoW reprezentaci termů pro každý jazyk zvlášť. Zpravodajský článek tedy bude reprezentován hashovací mapou, kde klíčem bude term a hodnota bude počet výskytů daného termu.

Tyto jazykově specifické slovníky dále použijeme k vytvoření TF-IDF matic, kde opět každý jazyk bude reprezentován svojí TF-IDF maticí. Vektory tvořící sloupce matic, ale nebudou klasické husté vektory ale pro ušetření paměti a urychlení operací při velkých rozměrech zvolíme vektory řídké tzv. SparseVectory, které obsahují pouze nenulové elementy a jejich indexy.

V následujícím výčtu si představíme třídy, které využíváme k tvorbě jazykových korpusů.

Nejdůležitější třídy a rozhraní:

- **IArticle** - Obecné rozhraní reprezentující článek, obsahuje metody, které usnadňují manipulaci s článkem
- **NewsContent** - Třída implementující rozhraní IArticle - představuje zpravodajský článek, oproti původnímu rozhraní obsahuje navíc ještě seznam všech nalezených entit
- **WikiArticle** - Třída, která reprezentuje Wikipedia článek, opět implementuje rozhraní IArticle
- **SparseVec** - Třída, která slouží k mapování standardního vektoru do vektoru hustého pro ušetření paměti
- **TFIDFMatrix** - TD-IDF matice tvořená množinou TF-IDF vektorů (implementovány pomocí SparseVec)
- **IWikiTrainingService** - Rozhraní služby, provádějící natrénování matic, které jsou využívány WikiFeaturesCreatorem.

8.4.5 Komponenta pro vytváření příznaků

Zřejmě nejpodstatnější součástí celého systému je vrstva zajišťující vytváření příznakových vektorů. K tomuto účelu používá vícejazyčné modely uvedené a popsané v kapitole 6.3.

Nyní si ukážeme nejdůležitější rozhraní této vrstvy.

Nejdůležitější rozhraní:

- **ISimFeaturesCreator** - Základní rozhraní pro práci s příznakovými vektory. Hlavním účelem tohoto rozhraní je možnost úpravy matice podobností zpravodajských článků. Toto rozhraní pak déle dědí další konkrétní Creatory zmíněné v dalších bodech.

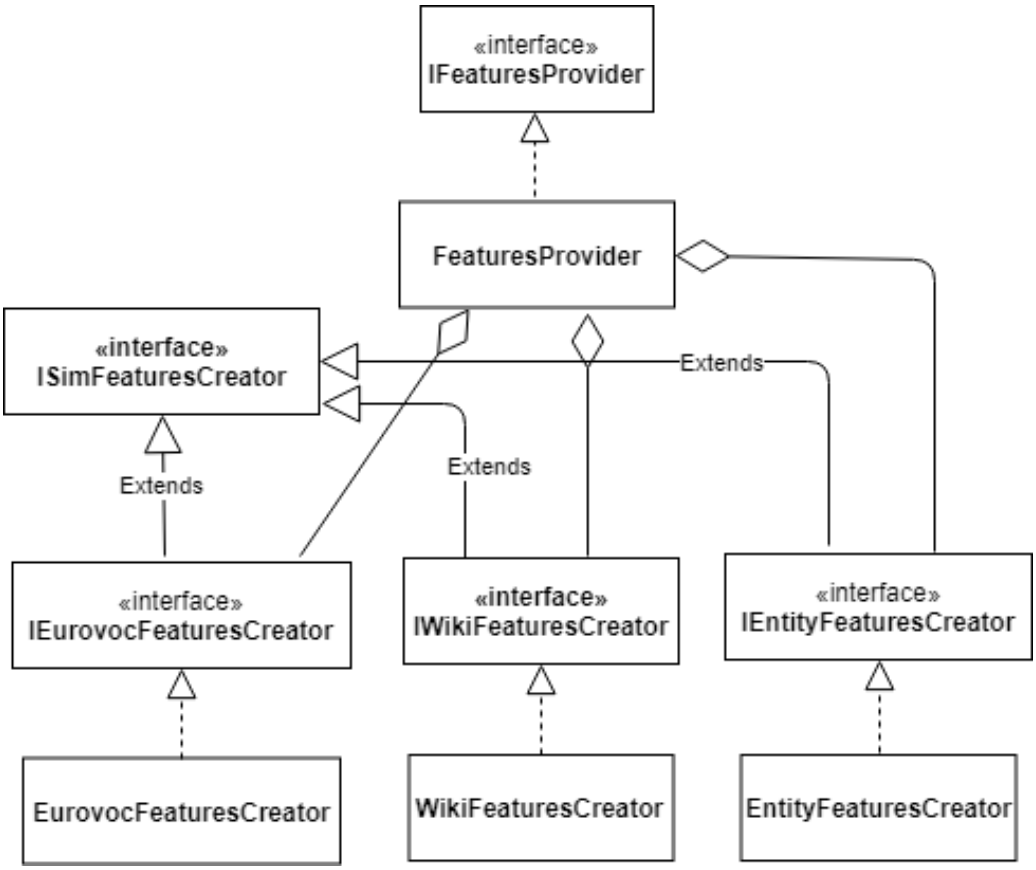
- **IEurovocFeaturesCreator** - Rozhraní, které značí, že příznakové vektory budou vytvořeny na základě Eurovoc deskriptorů. Konkrétní podobnost mezi dvěma zpravodajskými články pak počítá na základě poměru společných nejlépe ohodnocených deskriptorů. Počet dekriptorů, ze kterých vybíráme je nastavitelný v konfiguračním souboru.

- **IEntityFeaturesCreator** - Rozhraní velmi podobné předchozímu, s tím rozdílem, že nyní nepoužíváme k získání příznakových vektorů Eurovoc deskriptory, nýbrž poměr mezi nalezenými společnými entitami.

- **IWikiFeaturesCreator** - Rozhraní označující příznaky získané z embeddinových metod natrénovaných na Wikipedia korpusu (tedy K-means, CL-LSA a CL-ESA). K získání potřebných matic využívá třídu implementující rozhraní IWikiTraningService. Matice následně mohou být serializovány a uloženy do souborů pro rychlejší načítání. Volbu konkrétní metody a parametry metod nalezneme v konfiguračních souborech.

- **IFeaturesProvider** - Rozhraní, v rámci kterého se agregují všechny FeaturesCreatory, a jehož výstupem je matice podobností, která se dále využívá při shlukování.

Pro lepší představu o této komponentě si prohlédněte obrázek 8.5, na kterém je zachycen UML diagram použitých tříd a rozhraní.



Obrázek 8.5: UML diagram vytváření příznaků

Použité externí knihovny:

- **Colt**[3] - Colt je sada volně dostupných knihoven pro náročné vědecké a technické výpočty. V naší práci ji využíváme především pro operaci s maticemi a další operace spadající do oblasti Linerání algebry.
- **Smile**[4] - Smile nebo-li Statistical Machine Intelligence and Learning Engine je rychlá a komplexní knihovna podporující strojové učení, lineární algebru či interpolaci napsaná v Javě a Scale. Pro náš systém se tato knihovna hodí zejména pro výpočet SVD.
- **Weka**[5] - Weka (z angl. Waikato Environment for Knowledge Analysis) je knihovna obsahující celou řadu algoritmů strojového učení, napsaná v Javě. V našem systému ji využíváme na k-means algoritmus.

[3]https://dst.lbl.gov/ACSSoftware/colt/
[4]http://haifengl.github.io/smile/
[5]https://www.cs.waikato.ac.nz/ml/weka/

Kromě knihoven vyjmenovaných v předešlém výčtu jsme v rámci této práce vyzkoušeli i další knihovny pro operace z lineární algebry. Bohužel tyto knihovny nebyly dostatečně rychlé nebo komplexní, abychom je mohli použít.

Další vyzkoušené knihovny externí knihovny:

- **EJML**[6] - EJML nebo-li Efficient Java Matrix Library je knihovna pro Lineární algebru napsaná v Javě. V prvních pokusech s jednoduchým násobením matic dosahovala poměrně dobrých výsledků, nicméně zatím ještě neumožňuje výpočet SVD pro řídké matice a z tohoto důvodu v naší práci nebyla použita.

- **La4J**[7] - La4j je opět volně dostupná knihovna pro lineární algebru a je napsaná pouze v Javě. Tato knihovna umožňuje i výpočet SVD pro řídké matice. Nicméně její výkonost byla značně nedostačující a trénování matic trvalo příliš dlouho, a proto jsme ve finále zvolili knihovny jiné.

8.4.6 Komponenta na propojování článků

Nyní již máme připravenou matici podobností zpravodajských článků a můžeme přistoupit k samotnému shlukování článků s podobnými tématy, přičemž samotné shlukování je hierarchického typu. Zpočátku tedy vytvoříme spoustu malých clusterů a postupně se zbavujeme těch clusterů, které mají nejnižší vzájemnou podobnost, která se stanovuje podle aktuálních druhů příznaků. Vždy po několika iteracích clustery opět průběžně řadíme a algoritmus končíme až získáme požadovaný počet clusterů.

Stejně jako v předchozích sekcích i zde si uvedeme nejvýznamnější zástupce rozhraní a tříd sloužící ke shlukování.

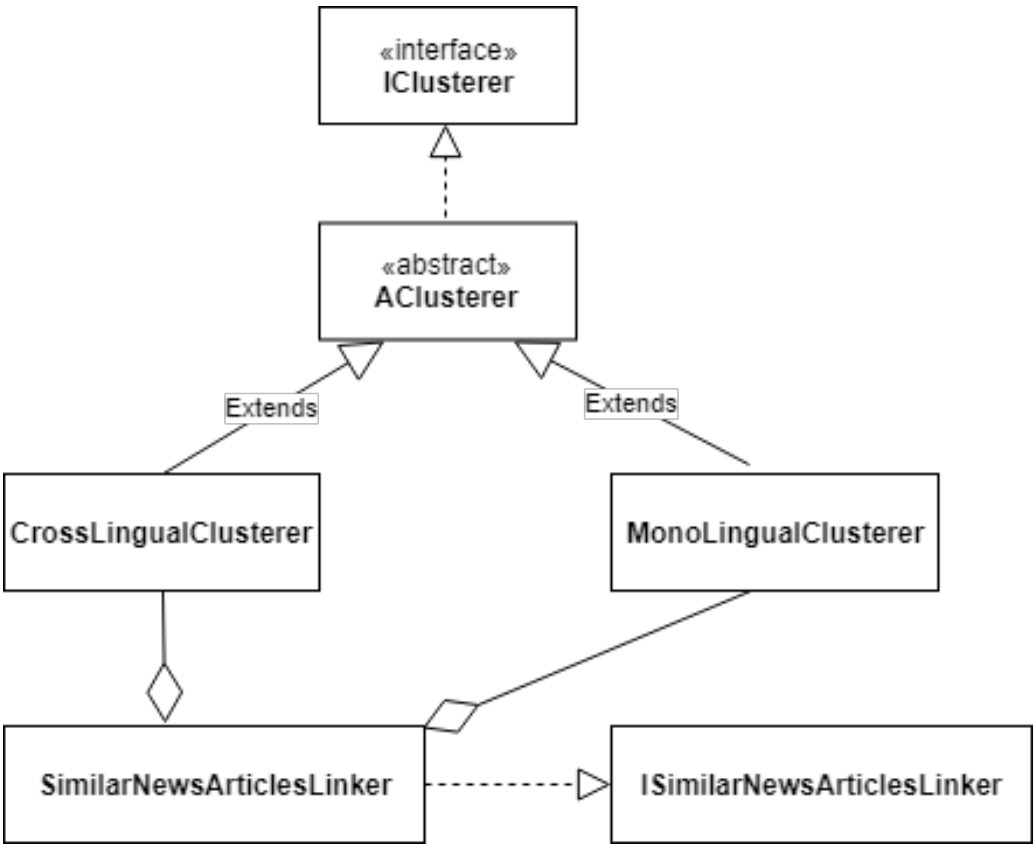
UML diagram těchto tříd pak nalezneme na obrázku 8.6.

Nejdůležitější třídy a rozhraní:

- **IClusterer** - Rozhraní popisující základní metody v rámci shlukování - tzn. inicializaci, provádění samotného shlukování a získání přiřazených clusterů

[6] http://ejml.org/
[7] http://la4j.org/

- **AClusterer** - Abstraktní třída implementující předchozí rozhraní, která obsahuje další užitečné protected metody, které jsou společné pro konkrétní clusterery
- **CrossLingualClusterer** - Vícejazyčný clusterer, který dědí od abstraktního clustereru. Shlukuje všechny zpravodajské články najednou na základě vypočítané podobnosti.
- **MonoLingualClusterer** - Clusterer, který shlukuje zpravodajské články vždy pouze v rámci jednoho jazyka. Takto získané shluky mohou být následně propojeny se shluky z jiných jazyků
- **SimilarNewsArticlesLinker** - Třída, která řídí samotné linkování, vytváří příslušné clusterery a operuje s výsledky.



Obrázek 8.6: UML diagram reprezentující shlukování

8.4.7 Komponenta pro vyhodnocení výsledků

V poslední fázi už nám nezbývá nic jiného, než vyhodnotit získané výsledky. Tedy vlastně použít některé z evaluačních metrik popsaných v kapitole 7 na naše clustery a výsledky případně zapsat do souboru.

Veškerá logika probíhá ve třídě LinksEvaluator, která nejprve načte ground truth data (budou blíže popsána v kapitole 9), následně začne rozřazovat získané clustery do tříd, přičemž výsledné clustery jsou reprezentovány objektem ResultCluster, a v posledním kroku už pouze spočítá příslušné evaluační metriky, přičemž momentálně podporované evaluační metriky jsou shrnuty v následujícím výčtu.

Použité evaluační metriky:

- **Purity**
- **F-Míra**
- **Přesnost**
- **Úplnost**
- **Rand-Index**

Kromě těchto metrik můžeme samozřejmě získat i počet správně přijatých objektů (TP), počet správně nepřijatých objektů (TN) a chybu prvního (FP) a druhého (FN) řádu. U F-míry lze samozřejmě libovolně nastavit parametr β (v našich testech, ale používáme pouze hodnotu 1).

Takto získané metriky pak mohou být v rámci třídy SimilarNewsArticlesLinker vypsány na standardní výstup a kromě toho mohou být pomocí třídy ResultsWriter zapsány do souboru, který kromě těchto metrik obsahuje i výpis získaných clusterů s dodatečnou informací o příslušnosti ke třídě v testovacích ground truth datech. Příklad takového souboru si můžete prohlédnout na obrázku 8.7.

```
News articles linking results
Settings: ENTITIES(1.0) EUROVOC(0.0) WIKI(1.0)
ESA:false, K-MEANS:false, SVD:true
SVD dimensions: 500, KMeans clusters: 500
Stemming: false, Min doc. freq:200
Clustering: Sort div interval: 4
Precisson: 0.5085458672719989, Recall: 0.5274200249272953
RandIndex: 0.9724874907877895
Purity: 0.6811023622047244, F(1)-measure: 0.5178110129163833
TP: 7617, TN: 493817, FP: 7361, FN: 6825

CL(125):
eeb5a23-Guatemala Brand in Kinderheim - Zahl der Opfer steigt auf mehr als 20(1-GUA)
15625ae-Brandkatastrophe in Kinderheim Zahl der Opfer in Guatemala steigt auf 34(1-GUA)
992836a-Guatemala Feuer in Kinderheim - mindestens 15 Tote(1-GUA)
847fc3b-Při požáru v guatemalském ústavu pro mládež zemřelo 30 dívek(1-GUA)
1d296a5-V dětském domově v Guatemale hořelo, 21 obětí(1-GUA)
666d106-V dětském domově v Guatemale hořelo, 19 obětí(1-GUA)
0b0998h-V Guatemale vzplál sirotčinec, uhořelo nejméně devatenáct dětí(1-GUA)
--------------------------------------------------------------------------------
CL(140):
1d980b1-Kindstötung in Herne Polizei prüft Hinweise auf weitere mögliche Opfer(2-HER)
df9a459-Mutmaßlicher Kindsmörder von Herne Zwei Tote und viele Fragen(2-HER)
d31d40e-Fahndung nach mutmaßlichem Kindsmörder 1400 Hinweise - aber keine heiße Spur(2-HER)
5edc3da-Mladík v Německu se chlubil na internetu, že zavraždil devítiletého chlapce(2-HER)
8bb7fds-Němci zadrželi mladíka podezřelého z vraždy malého chlapce(2-HER)
c7d69ec-V Německu zadrželi mladíka, který se chlubil vraždou chlapce. Sám se přihlásil(2-HER)
--------------------------------------------------------------------------------
CL(387):
779732c-Formula 1, Vettel giro record al Montmelo: "Bel passo avanti scorso anno"(42-FF1)
c2e984d-F1, Ferrari da record al Montmelo Ma Vettel frena: «Mercedes piu forte»(42-FF1)
6fb02bf-F1, Marchionne sprona la Ferrari: "Torniamo forti come tempi di Schumacher"(42-FF1)
6c51141-F1, Test a Barcellona: Massa leader, Vettel in palla, Hamilton punge(42-FF1)
d9da655-Addio a John Surtees, fu campione di F1 con la Ferrari nel 1964 e 7 volte(79-JSD)
35dfe33-Addio al mito John Surtees, fu l'unico campione del mondo in F1 e moto(79-JSD)
8924763-Nico Rosberg praises F1 cars as monstrous but looks can be deceiving(42-FF1)
67ccf77-Lewis Hamilton questions F1's new regulations before first test session(42-FF1)
7d132b6-Dark horses of Red Bull can turn F1 season into a much-needed thriller(42-FF1)
bd2fa4b-Lewis Hamilton fastest on first day of testing as Mercedes deliver on speed(42-FF1)
7c5863d-Lewis Hamilton warns new F1 rules will make overtaking more difficult(42-FF1)
efe09ab-John Surtees obituary(79-JSD)
d2289ef-John Surtees, former F1 and motorcycle world champion, dies aged 83(79-JSD)
c81c03a-Lewis Hamilton not missing Rosberg as he looks ahead to new F1 season(42-FF1)
--------------------------------------------------------------------------------
CL(445):
4ab925b-Ligue des champions : une « remontada » du FC Barcelone face au PSG ?(41-BPS)
ffb49fa-Arsène Wenger et Luis Enrique sur le banc de touche(43-AWE)
e07ca21-Ligue des champions : le Real Madrid et le Bayern Munich en quarts de finale(52-CHL)
d9c5c32-Tournoi des six nations : Novès rappelle Trinh-Duc avant l'Italie(44-RSN)
cedd2be-Dortmunds Sieg gegen Lissabon Fast so gut wie '63(52-CHL)
08f62ce-Bayerns Kantersieg in London 10:2(52-CHL)
168194d-Champions League Dortmund schlägt Benfica und steht im Viertelfinale(52-CHL)
548d1f2-Einzug ins CL-Viertelfinale 6:1 nach 0:4 - Barça schafft die Sensation(41-BPS)
b1ec25c-Champions-League-Achtelfinale Real schlägt Napoli(52-CHL)
741cccc-Premier League Chelsea wieder zehn Punkte vor Tottenham(47-PLM)
cc67ecf-Premier League Manchester City verpasst Sprung auf Platz zwei(47-PLM)
```

Obrázek 8.7: Příklad části výsledného souboru s výsledky

8.4.8 Komponenta pro správu konfigurace

Podstatnou součástí práce je také správa konfiguračních souborů, ze kterých se načítají jednotlivé properties. Každý properties soubor je spravován svým

manažerem a tito manažeři jsou dostupní v rámci statické metody třídy ConfigManager, jejíž instance představuje návrhový vzor jedináček.

Konfigurační strukturu si nyní popíšeme v následujícím přehledu nejdůležitějších tříd a rozhraní.

Nejdůležitější třídy a rozhraní:

- **IConfigManager** - Hlavní rozhraní agregující do sebe všechny ostatní properties manažery, instance třídy je dostupná jako singleton
- **IPropertiesManager** - Obecné rozhraní pro přístup k properties a jejich vytváření
- **APropertiesManager** - Abstraktní třída, implementující základní metody rozhraní
- **IPropertiesFactory** - Rozhraní obsahující definice klíčů a defaultních hodnot jednotlivých properties, dále obsahuje metodu na vytvoření properties s defaultními hodnotami
- **IAppMainPropertiesManager** - Rozhraní manažeru obstarávající hlavní aplikační properties soubor
- **INewsLinkingPropertiesManager** - Rozhraní manažeru pro správu properties na řízení propojování článků
- **IPreprocessPropertiesManager** - Rozhraní manažeru zajišťujícího předzpracování
- **ITrainingPropertiesManager** - Rozhraní manažeru pro správu trénovacích properties

9 Experimenty

9.1 Úvod

Cílem experimentů popsaných v rámci této kapitoly je ověřit kvalitu dosažených výsledků všech implementovaných metod. Pro vyhodnocení získaných clusterů jsme využili ručně vyhodnocený soubor s testovacími daty, který se skládá celkem z 1016 zpravodajských článků v pěti různých jazycích (konkrétně čeština, angličtina, němčina, italština, francouzština). Tyto články byly kolegy z katedry ručně rozřazeny do 70 clusterů na základě společných témat.

Zpravodajské články pocházejí z několika světových zpravodajských serverů a byly publikovány v rámci období jednoho týdne roku 2017. Tato testovací data jsou tvořena celkem 597 818 slovy a nejvíce článků je zastoupeno v angličtině.

Složitější metody využívající Wikipedia korpus byly trénovány na korpusu o velikosti 13200 článků, kde by každý článek měl mít svou jazykovou alternativu, přičemž tato podmnožina článků byla získána z přibližně 96000 různých Wikipedia článků tak, aby články byly přeloženy pokud možno do co nejvíce jazyků. Celkový počet slov v rámci našeho trénovacího korpusu pak byl 58 914 654. Nutno dodat, že tato data musíme z úsporných důvodů nejdříve filtrovat na základě četnosti, která se dá nastavit pomocí parametrů v konfiguračních souborech (jak minimální a maximální celkový počet výskytů, tak minimální a maximální počet dokumentů, ve kterých se slova vyskytují).

Jako referenční (baseline) metody nám poslouží implementačně jednoduché metody jež reprezentují: metoda hledání společných entit a metoda zkoumající podobnost společných Eurovoc deskriptorů. Pouze pro srovnání a představu pak máme k dispozici ještě naivní náhodné propojování článků.

Všechny metody byly testovány s různými parametry a výsledky byly sepsány do tabulek s evaluačními metrikami. V rámci práce byly použity dva základní přístupy ke shlukování. Prvním algoritmem jsme se pokoušeli shlukovat zpravodajské články rovnou vícejazyčně. Hledali jsme tedy nejlepší kandidáty napříč všemi jazyky. V druhém algoritmu jsme nejprve články shlukovali v rámci jednotlivých jazyků a následně jsme se pokoušeli hledat k těmto článkům jejich jazykové alternativy. Pro jednojazyčné shlukování jsme využili word2vec příznaky v kombinaci s entitami.

9.2 Propojování článků bez předchozího jednojazyčného shlukování

V rámci prvního algoritmu jsme na základě vzájemné podobnosti postupně shlukovali zpravodajské články přímo napříč různými jazyky.

U náhodného shlukování (Rand) předpokládáme náhodné zařazení článku do 1 ze 70 clusterů a výsledné hodnoty evaluačních metrik jsou pro něj zprůměrovány z 5 měření.

Pro metodu pracující s Eurovoc deskriptory (Eurovoc) počítáme vždy podobnost 30 nejlépe ohodnocených deskriptorů mezi dvojicí zpravodajských článků. U metod zkoumající entity (Entity) pak počítáme jejich procentuální shodu.

Emeddingové metody (K-means, CL-ESA, CL-LSA) ke svému chodu vyžadují natrénované TF-IDF matice X_i, které vypočítáme z trénovacího korpusu. Výsledné TF-IDF matice jsou o velikosti cca 15000 x 13200 pro každý jazyk a celkově jsou tyto metody velmi paměťově i výpočetně náročné. Počet sémantických dimenzí k byl zvolen na 500.

Získané výsledky jsou shrnuty do následující tabulky:

	Rand	**Eurovoc**	**Entity**	**K-means**	**CL-ESA**	**CL-LSA**
$Pur(\Omega, C)$	0.161	0,507	0,569	0,594	0,548	0,63
$RI(\Omega, C)$	0.958	0.965	0.961	0,963	0,968	0,971
$F_1(\Omega, C)$	0.02	0.2257	0.319	0,336	0,314	0,364
$P(\Omega, C)$	0.029	0.296	0.349	0,314	0,371	0,479
$R(\Omega, C)$	0.015	0.183	0.293	0,345	0,272	0,294
TP	220	2637	4250	4145	3682	4256
TN	493861	494888	493231	495145	495869	496554
FP	7317	6290	7908	6206	6235	4624
FN	14222	11185	10231	10124	9834	10186

Tabulka 9.1: Evaluační metriky testovaných metod pro algoritmus bez prvotního jednojazyčného shlukováním

9.3 Propojování článků s předchozím jednojazyčným shlukováním

Jak již bylo naznačeno v úvodu této kapitoly, v rámci toho algoritmu jsme nejprve hledali nejpodobnější články vždy v rámci jednoho jazyka, kde jsme jako hlavní zdroj příznaků používali word2vec příznakové vektory.

Konkrétně šlo o již natrénované příznakové vektory pomocí algoritmu word2vec continuous skipgram o dimenzi 100 bez použití stematizace a po následném předzpracování jsme vybrali cca 10 000 slov z každého jazyka. Tyto příznaky jsme částečně doladili společnými entitami.

Následně jsme opět použili již známé metody pro vícejazyčné propojování témat článků a hledali jsme nejlepší kandidáty k propojení nalezených shluků napříč různými jazyky. Pokud podobnost shluků byla dostatečně velká, provedli jsme spojení clusterů, v opačném případě jsme po nedostatečném ohodnocení všech potencionálních kandidátů nechali cluster samostatně.

Pro metody založené na Wikipedia korpusu jsme nejprve použili stejné velikosti TF-IDF matic jako v předchozím algoritmu tedy 15 000 x 13200 a opět $k = 500$.

Získané výsledky si můžete prohlédnout v následující tabulce:

	Rand	**Eurovoc**	**Entity**	**K-means**	**CL-ESA**	**CL-LSA**
$Pur(\Omega, C)$	0,161	0,606	0,632	0,638	0,619	0,642
$RI(\Omega, C)$	0,958	0,972	0,972	0,973	0,974	0,976
$F_1(\Omega, C)$	0,02	0,388	0,414	0,429	0,422	0,47
$P(\Omega, C)$	0,029	0,52	0,525	0,539	0,52	0,575
$R(\Omega, C)$	0,015	0,31	0,342	0,357	0,35	0,397
TP	220	4561	5006	5104	4998	5485
TN	493861	496683	496465	496974	496697	497768
FP	7317	4195	4523	4350	4610	4042
FN	14222	10181	9626	9192	9045	8325

Tabulka 9.2: Evaluační metriky testovaných metod pro algoritmus s prvotním jednojazyčným shlukováním

Kombinace embeddingových metod a společných entitami

Protože při použití druhého algoritmu, který nejprve jednojazyčně shlukoval témata, jsme dosáhli výrazného zlepšení oproti prvnímu algoritmu, rozhodli jsme se dále vyzkoušet, zda se výsledky ještě zlepší při použití kombinace příznaků z embeddingových metod s dalším druhem příznaků (konkrétně společnými entitami).

Získané výsledky jsme shrnuli v tabulce 9.3. Pro všechny metody byly opět použity stejné velikosti matic a sémantických dimenzí jako v předešlých případech.

	K-means	**CL-ESA**	**CL-LSA**
$Pur(\Omega, C)$	0,652	0,644	0,684
$RI(\Omega, C)$	0,97	0,969	0,972
$F_1(\Omega, C)$	0,487	0,472	0,524
$P(\Omega, C)$	0,477	0,468	0,513
$R(\Omega, C)$	0,497	0,476	0,535
TP	7162	6988	7867
TN	493390	493005	493467
FP	7834	7938	7461
FN	7234	7689	6825

Tabulka 9.3: Evaluační metriky testovaných metod pro algoritmus s prvotním jednojazyčným shlukováním a následným použitím embeddinových metod s doprovodnými příznaky ze společných entit

Kombinace metod - větší TF-IDF matice

Aby naše výsledky byly kompletní, rozhodli jsme se udělat ještě jeden experiment, kde jsme zkoumali, zda bude mít vliv na úspěšnost i výrazně větší velikost trénovacích TF-IDF matic.

Velikosti TF-IDF matic pro jednotlivé jazyky se pohybují okolo 22 000 x 16 000 - tedy cca 22 000 unikátních slov a 16 000 různých Wikipedia článků pro každý jazyk. Nutno dodat, že získané transformační vyhodnocující matice jsou velmi objemné (nejsou řídké) a celkový výpočet lze provést pouze na silných strojích s dostatečnou RAM a vysokým výpočetním výkonem.

Získané výsledky z tohoto experimentu jsou předmětem následující tabulky:

	K-means	**CL-ESA**	**CL-LSA**
$Pur(\Omega, C)$	0,692	0,689	0,714
$RI(\Omega, C)$	0,973	0,972	0,977
$F_1(\Omega, C)$	0,549	0,519	0,602
$P(\Omega, C)$	0,515	0,487	0,558
$R(\Omega, C)$	0,588	0,555	0,652
TP	8256	7643	8697
TN	493832	493851	495432
FP	7754	8022	6869
FN	5778	6104	4622

Tabulka 9.4: Evaluační metriky testovaných metod pro algoritmus s prvotním jednojazyčným shlukováním a následným použitím embeddingových metod s doprovodnými příznaky ze společných entit. Tentokrát s větším trénovacím korpusem

9.4 Diskuze

V rámci diplomové práce jsme vyzkoušeli 2 různé algoritmy na propojování článků a celkem 5 různých metod na výpočet podobnosti dokumentů napříč různými jazyky. Při provádění experimentů jsme přišli na to, že je nejprve vhodné články jednojazyčně propojit a až poté propojovat shluky s jinými jazyky, protože přímé propojování článků obecně dosahuje výrazně nižší úspěšnosti pro všechny použité metody.

Hlavní podíl na tom má poměrně zdařilé jednojazyčné shlukování, kterého se nám podařilo dosáhnout pomocí již natrénovaných word2vec příznakových vektorů. Zajímavé by jistě bylo tyto příznakové vektory nějakým způsobem i adaptovat na zkoumané embeddingové metody a vytvořit z nich taktéž transformační vyhodnocovací matici, která by sloužila k výpočtu podobnosti napříč různými jazyky. Nicméně tato adaptace by si zřejmě vyžádala poměrně velké množství času a dalšího experimentování, a proto v naší práci již nebyla vyzkoušena.

Implementačně nenáročné metody jako hledání společných entit či porovnávání Eurovoc deskriptorů fungují lépe než použité embeddingové metody natrénované na malých trénovacích datech, ale s rostoucí velikostí trénovacího korpusu postupně roste i úspěšnost embeddingových metod a implementačně jednoduché metody následně překonávají. Velké rozměry transformačních matic (u CL-LSA a K-means) znamenají vysokou paměťovou náročnost a výrazně pomalejší běh než jednoduché metody. Pokud bychom teoreticky uvažovali o nasazení v praxi, museli bychom metodu ještě mno-

hem více zefektivnit (např. paralelizací či použitím rychlejší knihovny pro násobení matic).

Vzhledem k tomu že Wikipedia obsahuje miliony článků z desítek různých jazyků, je náš systém snadno rozšiřitelný o další nové jazyky. O trénovací data nebudeme mít nouzi a bude nám stačit pouze vytvořit jen další transformační matice.

Vůbec nejlepších výsledků bylo dosaženo při prvotním jednojazyčném propojení témat a následném použití metody CL-LSA s doprovodnými příznaky získanými ze společných entit. Konkrétně f-míry 0.602 a purity 0.714. Lze předpokládat, že s ještě vyšším počtem témat z trénovacího Wikipedia korpusu bychom dosáhli ještě lepších výsledků, nicméně již tuto úspěšnost můžeme považovat za poměrně solidní.

Z pozorování výsledných clusterů lze konstatovat, že oproti referenčním ground truth datům často dochází k tomu, že se několik specifických témat spojí nakonec do jednoho velkého clusteru (např. fotbalový zápas FC Barcelona v lize mistrů a fotbalový zápas Arsenalu v rámci Premier League se spojí do jednoho clusteru pro fotbal a nerozdělí se správně na ligu mistrů a Premier League, apod.). Jemnějšího dělení lze často dosáhnout právě pomocí dodatečného hledání společných entit. Nutno však dodat, že velká část článků společné entity vůbec neobsahuje, a proto by bylo vhodné použít ještě další množinu dodatečných příznaků (nabízí se např. zeměpisné pojmy / města), která už se nám ale do této práce nevešla.

10 Závěr

V rámci této diplomové práce jsme se v prvních kapitolách nejprve seznámili se základními pojmy z oblasti vyhledávání informací a zpracování přirozeného jazyka. Dále jsme prozkoumali jak některé jednojazyčné metody pro výpočet podobnosti dokumentů, tak i vícejazyčné metody zkoumající podobnost textů napříč jazyky.

Jako vhodné metody na výpočet podobnosti jsme pro naši práci vybrali metodu porovnávající Eurovoc deskriptory, metodu hledající společné entity a 3 embeddingové metody natrénované na Wikipedia korpusu. Konkrétně pak CL-ESA, K-Means a CL-LSA.

Na základě získaných znalostí jsme navrhli metody pro propojení tématicky podobných shluků, přičemž jsme použili dva hlavní přístupy. V prvním přístupu jsme se pokoušeli články rovnou propojovat napříč různými jazyky a v druhém přístupu jsme nejprve články jednojazyčně shlukovali (pomocí word2vec příznaků) a až poté jsme hledali nejpodobnější clustery z jiných jazyků.

Metodu jsme následně implementovali v programovacím jazyce Java a v rámci jednoduchého experimentátoru jsme prováděli různé experimenty s navrženými metodami. Výslednou kvalitu propojování jsme otestovali na ground truth datech o 1016 článcích v 5 různých jazycích (dosažené výsledky včetně diskuze naleznete v kapitole 9), přičemž potřebné evaluační metriky jsme si předem blíže vysvětlili v rámci kapitoly 7.

Nejlepších výsledků jsme dosáhli při jednojazyčném předshlukování a následném použití metody CL-LSA s dodatečnými příznaky ze společných entit.

Navržené řešení má určitě ještě v mnoha oblastech své rezervy (ať už je to úspěšnost, rychlost metod či paměťová náročnost), avšak pozitivně hodnotím jeho snadnou rozšiřitelnost o další jazyky a možnost kombinovat s dalšími příznaky.

Realizace výsledného systému nebyla nikterak jednoduchá. Nejtěžší částí dle mého názoru bylo počáteční proniknutí do problematiky, obzvlášť bez předchozích znalostí např. z předmětu ANLP, který by se při realizaci této práce jistě hodil. Zpětně jsem velmi litoval, že jsem si ho nezapsal.

11 Seznam zkratek

BoW - Bag of Words
LSA - Latent Semantic Analysis
LSI - Latent Semantic Indexing
CL-LSA - Cross-lingual Latent Semantic Analysis
NLP - Natural Language Processing
VSM - Vector Space Model
PCA - Principal Component Analysis
SVD - Singular Value Decomposition
ESA - Explicit Semantic Analysis
CL-ESA - Cross-lingual Explicit Semantic Analysis
MMHL - Max-Margin Hinge Loss
SGNS - Skip Gram Negative Sampling
CBOW - Continuous bag-of-words
GloVe - Global Vectors
API - Application Programming Interface
CL-CNG - Cross-LanguageCharactern-GramModel
CCA - Canonical Correlation Analysis
CL-CCA - Cross-lingual Canonical Correlation Analysis
JPLSA - Joint Probabilistic Latent Semantic Analysis
PCLLSA - Probabilistic Cross-Lingual LSA
RMSSTD - Rootmean-Square Standard Deviation
NMI - Normalized Mutual Information
TP - True Positives
TN - True Negatives
FP - False Positives
FN - False negatives
CLI - Command line interface
GUI - Graphical user interface
UML - Unified Modeling Language
HTML - Hypertext Markup Language

Literatura

[1] Collobert, R. – Weston, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, s. 160–167. ACM, 2008.

[2] Deerwester, S. et al. Indexing by latent semantic analysis. *Journal of the American society for information science*. 1990, 41, 6, s. 391–407.

[3] Dumais, S. T. et al. Automatic cross-language retrieval using latent semantic indexing. In *AAAI spring symposium on cross-language text and speech retrieval*, 15, s. 21, 1997.

[4] Jolliffe, I. *Principal component analysis*. Springer, 2011.

[5] Koehn, P. et al. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, s. 177–180. Association for Computational Linguistics, 2007.

[6] Manning, C. – Raghavan, P. – Schütze, H. Introduction to information retrieval. *Natural Language Engineering*. 2010, 16, 1, s. 100–103.

[7] MATERNA, J. Sémantická analýza textů (5). *Blog fulltextového týmu*. 2019.

[8] Mikolov, T. et al. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, s. 3111–3119, 2013.

[9] Mnih, A. – Teh, Y. W. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*. 2012.

[10] Platt, J. C. – Toutanova, K. – Yih, W.-t. Translingual document representations from discriminative projections. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, s. 251–261. Association for Computational Linguistics, 2010.

[11] Potthast, M. et al. Cross-language plagiarism detection. *Language Resources and Evaluation*. 2011, 45, 1, s. 45–62.

[12] Pouliquen, B. et al. Multilingual and cross-lingual news topic tracking. In *Proceedings of the 20th international conference on Computational Linguistics*, s. 959. Association for Computational Linguistics, 2004.

[13] Pouliquen, B. – Steinberger, R. – Deguernel, O. Story tracking: linking similar news over time and across languages. In *Proceedings of the workshop on Multi-source Multilingual Information Extraction and Summarization*, s. 49–56. Association for Computational Linguistics, 2008.

[14] Rettinger, A. et al. Cross-lingual document linking prototype. *Xlike Deliverable D*. 2012, 4.

[15] Rupnik, J. et al. News across languages-cross-lingual document similarity and event tracking. *Journal of Artificial Intelligence Research*. 2016, 55, s. 283–316.

[16] Steinberger, J. MediaGist: A cross-lingual analyser of aggregated news and commentaries. *Proceedings of ACL-2016 System Demonstrations*. 2016, s. 145–150.

[17] Steinberger, R. – Pouliquen, B. – Hagman, J. Cross-lingual document similarity calculation using the multilingual thesaurus eurovoc. In *International Conference on Intelligent Text Processing and Computational Linguistics*, s. 415–424. Springer, 2002.

[18] Wolfgang, H. – Leopold, S. Canonical correlation analysis. *Applied Multivariate Statistical Analysis*. 2007, 3.

[19] Zhang, D. – Mei, Q. – Zhai, C. Cross-lingual latent topic extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, s. 1128–1137. Association for Computational Linguistics, 2010.

A Uživatelská dokumentace

A.1 SW požadavky

Abychom mohli aplikaci přeložit a spustit je zapotřebí mít na PC následující SW:

- **Java** - ve verzi 1.8 a vyšší
- **Maven** - ve verzi 3.3.9 a vyšší

A.2 Přeložení

Prvním krokem, který musíme provést je přeložit aplikaci. To provedeme tak, že se přepneme do složky **build** a spustíme následující dva příkazy:

- **mvn clean install**
- **mvn clean compile assembly:single**

Pokud vše proběhlo správně v **target** složce se nám vygeneroval náš jar archiv s názvem **thesis-1.0.0-SNAPSHOT.jar**. V dalším kroku musíme přemístit náš jar archiv do složky **environment**, pokud jsme tak učinili, můžeme překročit k samotnému spuštění aplikace.

A.3 Spuštění

Aplikaci spustíme jednoduše tímto příkazem:

- **java -jar thesis-1.0.0-SNAPSHOT.jar <Parametry>**

Parametry

- **-link** - parametr pro spuštění propojování článků

  <output filename> - explicitní jméno výstupního souboru
- **-wiki_corp** - parametr pro trénování nového wikipedia korpusu
- **-help** - parametr pro zobrazení nápovědy

Většina parametrů je pak brána přímo z konfiguračních souborů, které jsou dostupné ve složce **src/main/resources**.

A.4 Možné problémy

- **Chybný build kvůli snowball stemmeru** - pokud se nepodaří build z této příčiny, je nutné nejprve provést **mvn clean install** ve složce **tmp** - tato složka obsahuje zdrojové soubory snowball stemmeru, které se po přeložení dostanou do lokálního maven repositáře a chyba by měla zmizet

- **Aplikace nelze správně spustit** - ujistěte se, že jste přemístil jar archiv do složky **environment**

- **Nelze přeložit ani spustit** - ujistěte se, zda máte nastavený **maven** a **Javu** v systémových proměnných.

- **Nedostatek heap space** - Pokud není nastavená dostatečná paměť pro heap je nutné ji zvýšit argumentem: **-Xmx** a velikost v MB (pozn. výše paměti je ovlivněna především velikostí trénovacího wikipedia korpusu - **doporučená velikost přidělené paměti je minimálně 5-8 GB**).

B Konfigurace

Seznam konfiguračních souborů včetně významu jednotlivých klíčů

app.properties

- **default.output.dir** - defaultní výstupní adresář, kam se budou ukládat generované soubory s výsledky
- **news.files.format** - formát vstupních zdravodajských článků
- **default.input.directory** - defaultní vstupní adresář, který obsahuje zpravodajské články
- **app.corpus.from.file.enabled** - flag, který symbolizuje, zda mohou být načítány matice ze souborů
- **app.sim.matrix.ser.file.load.enabled** - flag povolující načítání již natrénované matice podobností - pro urychlení testování (na velkých korpusech trvá výpočet dlouho)

preprocess.properties

- **preprocess.doc.filter.esa.min** - minimální počet dokumentů, ve kterých se musí term vyskytovat - pro ESA
- **preprocess.doc.filter.esa.max** - maximální počet dokumentů, ve kterých se musí term vyskytovat - pro ESA
- **preprocess.doc.filter.main.min** - minimální počet dokumentů, ve kterých se musí term vyskytovat - pro SVD a KMeans
- **preprocess.doc.filter.main.max** - maximální počet dokumentů, ve kterých se musí term vyskytovat - pro SVD a Kmeans
- **preprocess.stemmer.enable** - flag pro aktivaci stematizace
- **preprocess.word.filter.enable** - flag pro aktivaci slovního filtru
- **preprocess.doc.filter.main.max** - maximální počet výskytů slova v korpusu

- **preprocess.doc.filter.main.min** - minimální počet výskytů slova v korpusu
- **preprocess.stopwords.dir** - adresář, který obsahuje stopslova pro konkrétní jazyky
- **preprocess.root.dir** - základní adresář pro preprocessing
- **preprocess.tmp.dir** - pomocný adresář pro preprocessing

training.properties

- **training.kmeans.ser.file** - jméno pod, kterým bude uložena serializovaná k-means transformační matice
- **training.wiki.aligned.dir** - adresář obsahující tematicky zarovnané wikipedia články - srovnatelný korpus
- **training.wiki.ser.root.dir** - jméno, pod kterým budou uloženy serializované TF-IDF matice
- **training.esa.ser.root.dir** - sllžka, do které budou uloženy serializované objekty potřebné pro ESA
- **training.svd.ser.file** - jméno, pod kterým bude uložena serializovaná svd transformační matice
- **training.esa.root.dir** - adresář obsahující zmenší wikipedia korpus pro ESA
- **training.kmeans.clusters** - počet clusterů pro k-means
- **training.svd.size** - počet dimenzí SVD
- **training.eurovoc.desc.count** - počet nejlepších Eurovoc deskriptorů
- **training.reduced.vocab.enabled** - flag, který zapíná či vypíná komprimovaný slovník

news_linking.properties

- **linking.entities.weight** - určuje váhu jakou budou mít příznaky získané ze společných entit
- **linking.wiki.weight** - určuje váhu jakou budou mít příznaky embeddingových metod natrénovaných na Wikipedia korpusu
- **linking.eurovoc.weight** - určuje váhu jakou budou mít příznak získané z eurovoc deskriptorů
- **linking.kmeans.enabled** - flag označující zda je aktivní k-means
- **linking.esa.enabled** - flag označující zda je aktivní ESA
- **linking.svd.enabled** - flag označující zda je aktivní SVD
- **linking.sorting.div.int** - poměr, kterým budeme dělit počet clustrů - pro průběžné řazení
- **linking.mono.start.enabled** - flag označující zda nejprve budeme články jednojazyčně shlukovat
- **linking.big.res.matr.enabled** - flag označující, zda bude použita při výpočtu velká transformační matice pro každou dvojici jazyků (má vysokou úspěšnost, ale trénování je extrémně pomalé)

C Obsah DVD

- **doc** - složka obsahující java doc dokumentaci a text diplomové práce v pdf
- **build** - složka obsahující vše potřebné k přeložení aplikace
- **environment** - složka obsahující pomocné soubory pro běh aplikace (konfigurační soubory, korpusy, serializované soubory, atd.)
- **results** - složka s výběrem několika výsledných souborů
- **tmp** - složka s pomocnými soubory - např. zdrojové soubory snowball stemmeru
- **Poster** - složka obsahující poster
- **readme.txt** - nápověda