SPECIFIKACE ROČNÍKOVÉHO PROJEKTU

KAREL BÍLEK

Obsah

1. Úvod

Ročníkový projekt „Analýza zpravodajských témat“ (*také „Rozpoznávání okurkové sezóny“*) si za úkol dává automatické rozpoznávání zpravodajských témat v českých internetových denících, včetně rozpoznávání diskutovaných jmen. Projekt bude používat již hotových modulů projektu TectoMT na lemmatizaci slov nebo rozpoznávání pojmenovaných entit.

2. Stávající systémy

2.1. **Google News.** Jeden z mála veřejných systémů, zabývajících se podobným tématem, je projekt Google News společnosti Google na adrese `http://news.google.cz`. Google News seskupuje dohromady články na stejná témata a články, zabývající se nejvíce diskutovanými tématy, zobrazí na hlavní straně. články též třídí do „skupin“, jako *Domov*, *Svět* nebo *Sport*. V anglické verzi také umožňuje tzv. News Alert - pravidelné upozorňování na nové články o uživatelem zadaném tématu. Taktéž umí ve článcích vyhledávat a výsledky vyhledávání seřadit podle času.

V minulosti fungoval nějakou dobu český klon služby Google News, nazvaný Nový den, odkoupený portálem Atlas. Po odkoupení se vývoj služby zastavil a po několika letech bez vývoje byl projekt ukončen. Služba Nový den fungovala nejdříve na adrese `http://www.novyden.cz`, následně `http://novyden.atlas.cz`. Dnes je první doména mrtvá, druhá vede na server `http://aktualne.cz`.

V něčem bude můj projekt podobný Google News - bude schopný rozpoznávat základní témata článku, nevím, jestli lépe než Google News - narozdíl od nich ale témata jasně pojmenuje a vypíše. Nebude dávat dohromady články podle podobnosti témat, ale bude fungovat „naopak“ - zobrazí konkrétní témata, a teprve k nim články *(viz 7)*. Také nebude seskupovat články do skupin jako *Domácí* nebo *Svět*. Naopak bude lépe zobrazovat historii konkrétního tématu. *(definice tématu viz 5.1.)*

Na druhou stranu ale vyhledávač nebude nic fulltextově vyhledávat - každé téma bude mít na disku „natvrdo“ napsán seznam článků, které dané téma zmiňují. *(viz 11.9)*

3. Základní systémové požadavky

Program, provádějící analýzu, by možná mohl běžet na všech systémech založených na UNIXu, prakticky bude ale přizpůsoben na systémy Linux, hlavně z důvodu navázanosti na TectoMT, který musí být na počítači nainstalován. Dále je potřeba Perl ve verzi alespoň 5.8 a některé Perl moduly ze CPANu. V téhle fázi je opravdu nedokážu vyjmenovat všechny.

Program ale nebude, alespoň ne ze začátku, koncipován jako stand-alone aplikace, ale přizpůsoben bude serverům na Malé Straně, opět hlavně z důvodů bezproblémového běhu TectoMT.

Na systému už musí běžet webserver.

4. Velmi hrubý popis

4.1. **Hlavní součásti programu.** Program bude rozdělen na dvě části - jedna část bude robot, sbírající čistá data a analyzující je, který bude pravidelně spouštěn např. přes `cron`; druhá část bude skript na webserveru, který bude generovat samotné výstupy.

4.2. **Interakce s uživatelem.** Uživatel bude používat program jako webovou službu. Uživatel navštíví hlavní stránku a uvidí nějak zpracované výsledky analýzy ve formátu HTML. Odkazy v tomto dokumentu povedou na detailnější výsledky.

5. Definice

5.1. **Téma článku.** Téma článku je skupina jednoho nebo více slov, která se často v daném článku opakuje a článek nějak charakterizuje.

Naopak, v tomto projektu to není nějaké abstraktní spojení, popisující daný článek. Pokud se například v článku často mluví o zatmění slunce, tématem článku je slovní spojení „*zatmění slunce*“. Není to ale spojení „*astronomické jevy*“, pokud se toto spojení nějak výrazně neopakuje.

Je pravda, že tyhle dvě definice se trochu prolínají - pokud článek mluví například o nové politické straně, často se ve článku například objeví slovo *volby* nebo *politika* - ale již se příliš neobjeví spojení *„česká politická scéna“*.

Mimo této „definice“ jsou do témat počítány **všechny** názvy osob, míst a institucí (a možná některé další tzv. pojmenované entity), protože mají ve článku mnohem větší význam, než ostatní slova. Výjimku tvoří některé příliš často se opakující názvy, jako je například *„česko“*, viz také 5.2.

Napůl formálně a pro potřeby dalších definic je možné vnímat množinu všech témat, to znamená vlastně množinu všech slovních spojení, následovně:

5.1.1. *Množina slov.* Množina všech existujících slov S. Ve skutečnosti nejde přesně o slova, ale o rozložení na třídy ekvivalence podle shodného normalizovaného tvaru - slova „chřipka“ a „chřipky“ se fakticky nerovnají, v množině S jakoby ano, jsou tedy pouze ve stejné třídě ekvivalence.

Tyto třídy ekvivalence budu ale většinou ignorovat, tj. pokud budu dále ve specifikaci mluvit o rovnosti slov, je většinou myšlena stejná třída ekvivalence.

5.1.2. *Množina všech témat.* Množina všech témat, tj. množina všech slovních spojení, je množina $\{t : n \to S | \forall n \in N\}$, tj. množina všech uspořádaných n-tic, tj. množina všech konečných posloupností n.

5.1.3. *Nadtéma.* Rekneme, že spojení $a : m \to S$ je nadtéma $b : n \to S$, pokud $m < n, \exists k < n, k + m \leq n, \forall i \in N : i < m \Rightarrow a(i) = b(i)$ - řečeno jasněji, pokud ve spojení b je spojení a obsaženo. Např. spojení *„prasečí chřipka“* je nadtématem *„výskyt prasečí chřipky“*.

5.1.4. *Svaz témat.* Pokud přidáme k množině S relaci „je nadtéma“ jako relaci menší než a do nosné množiny témat přidáme prázdné slovo, vznikne svaz témat.

5.1.5. *Maximální témata v množině.* V každé konečné množině témat lze vybrat všechna maximální témata (podle relace na svazu).

Ve skutečnosti se u článků nebudu pokoušet najít všechna jejich témata, ale jenom maximální témata jejich množiny témat. Pokud tedy někde uvedu „všechna témata článku“, je tím často myšleno pouze všechna maximální témata na svazu s trochu složitě definovanou, ale poměrně jasnou relací.

Ještě jednou - znamená to, např., že pokud bude množina témat článku {„prasečí chřipka“, „výskyt prasečí chřipky“, „povinné očkování“}, téma „prasečí chřipka“ mě nezajímá - není maximální, je nadtématem „výskyt prasečí chřipky“.

5.2. **Stopslovo.** Klasická definice stopslova je slovo, co nenese žádnou významovou informaci, pouze syntaktickou. V tomto projektu je to rozšířeno ještě o další slova, co sice můžou nést významovou informaci, ale v novinových článcích se opakují tak často, že daný článek nijak necharakterizují. Jsou to například tvary slova *člověk* - množné číslo *lidé* se objevuje v článcích příliš často na to, aby bylo pro článek jakkoliv charakteristické. Dalšími stopslovy jsou také například jména samotných zpravodajských serverů.

Při výpisu témat se chci stopslovům úplně vyhnout - na druhou stranu, pokud nějaké *víceslovné* téma obsahuje stopslovo, stále je to téma (například *desítky truchlících lidí*).

Pomocí už definovaného se dá stopslovo také definovat jako minimální prvek v množině témat článku, který se vyskytuje příliš často v jiných článcích.

6. Hrubý popis

6.1. **Použitá technologie.** Robot, který bude mimo jiné analyzovat výsledky a ukládat témata, bude napsán kompletně v Perlu, jednak kvůli integraci se systémem TectoMT, jednak díky jeho schopnosti rychle pracovat s velkým množstvím textu.

Skripty, které budou výsledky zobrazovat uživateli, budou nejspíš také v Perlu jako CGI skripty, ale tady si zvolenou technologií nejsem tak jist. Možná, že nakonec použiji místo perlu PHP skripty.

Program využívá systému TectoMT a hlavně dvou modulů - morfologického taggeru MorČe a automatického rozpoznávače pojmenovaných entit. Oba TectoMT moduly už musí na počítači být nainstalovány.

6.2. **Části programu.** Samotný analyzér bude rozdělen do několika modulů.

- Jeden modul bude jednoduchá „čtečka“ RSS, která projde přes veřejné RSS feedy, a najde články, co ještě nepřečetll
- Další modul vezme HTML kód a z něj vypíše pouze čistý text článku
- Další modul vezme tento text článku a najde v něm všechna témata
- Další modul, spouštící všechny předchozí, projde několikrát za den zadané RSS feedy a výsledky uloží do *dočasných* výsledků.
- Další modul z dočasných výsledků spočítá tzv. žebříček témat
- Další modul jednou za den vezme tyto dočasné výsledky a uloží je do *trvalých* výsledků pro tento den.
- Další modul jednou za delší periodu (týden, měsíc) prohlédne několik posledních výsledků a vytvoří statistiku.
- Skript na webserveru, zobrazující výsledky, bude poté přistupovat jak k dočasným, tak k trvalým výsledkům za minulé dny.

7. Co přesně chceme?

Po programu vlastně chceme několik úkolů, které nejsou úplně totéž.

- Pro jeden článek v databázi vypsat jeho témata.
- Pro pevně zadaný den vypsat nejčastější / nejdůležitější témata.
- Pro pevně zadané téma vypsat jeho historii v počtech zpráv, případně i podílech jednotlivých serverů, na časové ose.
- Pro pevně zadané téma a den vypsat odpovídající články.
- Pro pevně zadané téma a periodu vypsat odpovídající články (tento bod možná bude vynechán)
- Pro některé periody (30 dní, 7 dní) vypsat nejčastější / nejdůležitější témata.

8. RSS čtečka

RSS čtečka využívá už hotového modulu XML::RAI.

Už *sama RSS čtečka* bude kontrolovat, které články už byly nebo nebyly přečteny (tj. ostatní moduly už to nekontrolují). Za tím účelem má RSS čtečka uloženy již

přečtené webové adresy v textovém souboru. Pro lepší kontrolu jsou ještě adresy rozděleny do více souborů podle domény (např. *idnes*).

Adresy jsou uloženy v prostém textovém souboru. Protože tento soubor RSS čtečka načítá celý, v souboru se zachovává pouze nějaké množství (řekněme 100) nejnovějších článků - více je zbytečné, protože články starší už se v RSS znovu stejně neobjeví.

Konkrétní RSS feedy jsou čtečce zadávány v textovém souboru, jehož relativní adresa je „napevno“ nastavená v modulu.

Hlavním výstupem modulu je perlovské pole s odkazy.

9. Extraktor textu

Podoba tohoto modulu ještě není jistá; v současné pre-alfa verzi funguje systémem, založeným na ručně vedeném seznamu „dobrých“ a „špatných“ HTML elementů; pro dlouhodobější fungování pokud možno bez zásahu člověka by asi bylo dobré systém změnit.

9.1. **Stávající řešení.** HTML::DOM modul chápe HTML jako strom, přičemž tagy jsou uzly a synové můžou být buď další tagy, nebo čistý text.

Tento DOM strom je extraktorem procházen do hloubky následujícím způsobem, přičemž existuje funkce, co ohodnotí jakýkoliv tag buď jako *chtěný*, *nechtěný* nebo *neutrální*, kde kořen - *<HTML>* tag - je neutrální.

Dále mám dvě procedury, *zpracuj_neutrální* a *zpracuj_chtěný*.

Procedura *zpracuj_neutrální* udělá pro každého syna toto:

- je-li to text, ignoruje ho
- je-li to chtěný tag, pustí na něj *zpracuj_chtěný* a výsledek připojí ke svému výsledku
- je-li to nechtěný tag, ignoruje ho
- je-li to neutrální tag, pustí na něj *zpracuj_neutrální* a výsledek připojí ke svému výsledku

Procedura *zpracuj_chtěný* udělá pro každého syna toto:

- je-li to text, připojí ho ke svému výsledku
- je-li to chtěný tag nebo neutrální tag, pustí na něj *zpracuj_chtěný* a výsledek připojí ke svému výsledku
- je-li to nechtěný tag, ignoruje ho

Hodnotící funkce vlastně ukazuje na části, které *opravdu jsou* text článku; a zase naopak na ty, které *určitě nejsou* text článku - například reklamní bloky.

Právě teď je hodnotící funkce nastavená velmi primitivně - ve dvou externích textových souborech je jednoduše uložen seznam chtěných a nechtěných tagů. Seznamem tagů mám na mysli seznam přímo názvů tagů (zřídka), ale hlavně atributů ID a CLASS. Pokud tedy má nějaký tag atribut ID, který je např. ze seznamu chtěných, je chtěný.

Tato funkce je oním slabým místem. Seznamy se musí zatím ručně, tj. ne automaticky, updatovat, podle toho, jak se zrovna webdesigner toho kterého serveru rozhodl nazvat třídu DIVů, kde je skutečný článek. Zatím ale musím říci, že velké zpravodajské servery svůj design příliš často nemění, takže tento systém, alespoň dočasně, funguje.

9.2. **Jiné řešení.** Pokud bych zvolil jiné řešení, asi bych nechal DOM průchod tak, jak je, jen bych výrazně změnil hodnotící funkci.

Jedno z možných řešení by mohlo být hodnotit, kolik vět v DIVu je - pokud se odstavec skládá pouze z věty nebo dvou, jde často o reklamu nebo upoutávku na jiný článek. Naopak, články jsou vždy dlouhé alespoň tři věty.

Tento přístup je ale pouze nápadem a zatím nevím, jestli by skutečně fungoval.

Dalším nápadem je použití už vytvořeného systému Viktor - `http://ufal.mff.cuni.cz/victor/`.

10. Analyzér témat

Tato část je první a jediná, kde se vůbec používá systém TectoMT a obecně morfologická analýza. Dříve, než se pustím do popisu algoritmu, uvedu několik pozorování.

10.1. **Pozorování.**

- množina stopslov v lemmatizované formě je poměrně malá, i s mojí mírně rozšířenou definicí
- víceslovná spojení mnohem lépe charakterizují článek než jednoslovné výrazy
- výrazy se mnohem lépe počítají v lemmatizovaném tvaru, ale...
- samotná lemmatizace „slovo po slově“ někdy dělá slovní spojení prazvláštním a někdy i otáčí význam (například ze slovního spojení *neskutečně velké zatáčky* se stane *skutečně velký zatáčka*)
- pokud se často opakuje nějaké téma, opakují se, přirozeně, i jeho nadtémata
- z článku mě ale zajímají jenom maximální prvky v množině témat

Na základě těchto pozorování byl vymyšlen algoritmus, který využívá schopnosti Perlu rychle ukládat a číst informace z asociativních polí, i poměrně velkých.

10.2. **Algoritmus.** Analyzér nejdřív spustí na text morfologický tagger, čímž je každému slovu přiřazena jeho lemmatizovaná verze.

Následně vytvoří asociativní pole `%formy`, které **každému** „po slovech“ lemmatizovanému sousloví, kratšímu než nějaké pevně dané k (např. 10), přiřadí zpátky některou jeho formu, tj. nelemmatizovanou verzi. Například
`$formy{"skutečně velký zatáčka"}="neskutečně velké zatáčky"`.

Poté, v jednom průchodu textem, do dalšího asociativního pole `%skóre`, pro každé takové lemmatizované sousloví spočítá jeho *skóre*. Skóre je číslo, závislé jak na *počtu výskytů*, tak na *počtu slov v sousloví* - delší spojení jsou zvýhodňovány.

Vzorec na skóre musí být zvolen tak, aby byly delší skupiny slov zvýhodňovány dostatečně, ale zároveň tak, aby krátká slova vůbec měla šanci je „přebít“. Také musí být skóre lineárně závislé na počtu výskytů, aby šlo při průchodu textu do asociativního pole `%skóre` jednoduše přičítat. V současnosti funguje dobře vzorec $s = n \times (2 - \frac{1}{m})$, kde n je počet výskytů a m je počet slov - tedy pro jednoslovné výrazy je skóre rovno počtu výskytů, naopak pro libovolně zvýhodňovaná sousloví bude skóre menší, než dvakrát počet výskytů.

Tedy, například, pokud se v textu bude vyskytovat *neskutečně velké zatáčky* a *neskutečně velkých zatáček*, ve `$výskyty{"skutečně velký zatáčka"}` bude $3\frac{1}{3}$.

V počítání skóre se úplně ignorují stopslova - pokud je slovo stopslovo, nepočítá se mu skóre. Jelikož je množina stopslov v lemmatizovaném tvaru poměrně malá, je vytvořená ručně a uložená v pomocném textovém souboru.

Následně se vybere nějaké pevně dané množství, např. 20, lemmatizovaných tvarů s největším skóre. Z nich se ještě odstraní všechny podskupiny slov, aby dvě témata nebyly porovnatelná podle 5.1.4. Z těch témat, která zbydou, se pomocí asociativního pole `%formy` stvoří původní řetězce.

K takto vzniklým tématům se ještě přidají některé nalezené pojmenované entity, konkrétně například všechna jména osob nebo míst. Tyto témata ale nejsou od „počítaných“ témat odděleny, zvenku je tedy nerozlišitelné, jestli jde o pojmenovanou entitu, nebo o „počítané“ téma.

Přidání entit se může zdát umělé, ale názvy osob nebo míst, ač se objevují třeba jen jednou, mají vyšší důležitost než ostatní slova a doplňují charakteristiku článku.

10.3. **Nápady na zlepšení.** Je pravda, že tady se analyzér naprosto spoléhá na správnost morfologického taggeru, což se ne vždy vyplácí - např. některá nově převzatá slova nebo novotvary někdy tagger zlemmatizuje špatně a pokaždé trochu jinak.

Místo na základě vztahu mezi formou a lemmatem by se mohly tedy skóre počítat i např na základě jakési „podobnosti“ - pro každé sousloví by se přičetlo nejvyšší skóre nejvíce podobným souslovím, a menší skóre méně podobným souslovím. Celé to lemmatizování by potom šlo pryč a jiné tvary stejného sousloví by si byly nejvíce podobné.

Nejsem si ale jist, jakým způsobem by tenhle nápad šel provést efektivně, aby při každém sousloví nemusel algoritmus procházet všechny ostatní sousloví a počítat podobnosti.

Dále by se mohl „evolučně“ / adaptivě zlepšit vzorec na výpočet skóre - ještě před ostrým spuštěním by na omezeném počtu článků systém automaticky našel více témat, uživatel by témata potvrdil nebo zamítnul a systém by nějak upravil samotný vzorec.

11. Ukládání výsledků

Opět uvedu krátkou diskuzi o možnostech ukládání.

11.1. **Co ukládat.** Naskýtá se otázka - které informace vůbec ukládat a které naopak nechat být.

- Pokud se budou ukládat pouze zdrojové texty článků, to znamená bez denních výsledků nebo nalezených témat, musely by se tyto při každém požadavku znovu generovat a vyhledávat, což se mi nezdá rozumné - tagger a lemmatizér sám o sobě má nemalou spotřebu zdrojů, navíc by se např. při každém požadavku na denní žebříček témat musel pouštět znovu, na všechny články. To znamená, že ukládat si ke článku i jeho témata je určitě rozumné.
- Další otázka je, má cenu původní články *vůbec* ukládat? Pokud bychom články neukládali, asi bychom získali dost místa na disku. Na druhou stranu, je možné, že časem se analyzér témat ještě více vylepší, nebo nějak radikálně změní vůbec svůj výstup, a nasbírané články se v tu chvíli budou hodit - znovu je po serverech *sbírat* bude skoro nemožné.
- Další otázka je, jestli je nutné ukládat žebříčky za konkrétní periody v samostatných souborech, nebo tyto také generovat automaticky. Tady si chci ponechat trochu volnou ruku, ale zatím jsem toho názoru, že je lepší ukládat „extra“ jak výsledky za dny, tak za týdny a měsíce. Znamenalo by to tedy například, že výsledky za týden by existovaly pouze za každý kalendářní týden a za měsíc za kalendářní měsíc (to znamená, že nebude možno uživatelsky zvolit vlastní počáteční a koncové datum, pro které by se vytvořil žebříček - naopak, všechny výsledky za kalendářní týdne a měsíce budou uloženy na disku).
- Dokonce bych s ukládáním co nejvíce informací na disk, místo počítání na místě, šel ještě o krok dál, ale tady jsem si *ještě nejistější*. Jedna z informací, kterou budu po systému chtít, je historie konkrétního tématu. Protože nechci fulltextový vyhledávač, *možná* by stačilo uložit historii **každého** tématu do zvláštního souboru, který by se po vyhledávání načetl - vyhledání historie tématu by tedy byla otázka jednoho načtení. Na druhou stranu, nedokážu až tolik odhadnout, kolik témat za delší časový úsek vlastně bude, a nakolik bude tedy možné ukládání každého tématu do zvláštního souboru provést.
- Tedy - nakonec jsem se rozhodl uládat téměř **vše** - od jednotlivých článků a jeho témat až po historii konkrétních témat. Víc v následujících odstavcích.

11.2. **Krátkodobé / dlouhodobé výsledky.** Další otázkou je, jestli má smysl dělit výsledky na *krátkodobé* a *dlouhodobé*, nebo jde o něco, co pouze zdržuje. Podle mě má rozdělení smysl.

- Jelikož v systému je dnem myšlen čas od půlnoci do půlnoci, články za „dnešní“ den je lepší dávat do dočasného místa, kde budou soubory a případné výsledky přibývat. Jakmile den uplyne a nastane půlnoc, počty článků se přidají do trvalé historie témat - a cokoliv ze složky pro tento den už nebude měněno, bude z ní pouze čteno, a všechny nové články budou ukládány do další dočasné složky jinam.
- Protože do dočasných výsledků se spíše píše a z trvalých se spíše čte, mohou se zvolit jiné formáty ukládání, více uzpůsobené na čtení nebo zápis - ačkoliv program předpokládá, že číst se bude i z dočasných výsledků, ale ne tak často (dočasné výsledky jsou dočasnými pouze maximálně 24 hodin). Zatím to ale není pravda - jak dočasné, tak trvalé výsledky mají stejný formát uložení, jen jsou uloženy *jinde*.
- Složkám a všem podsouborům se můžou nastavit read-only atributy, což zabrání buď chybě v programu, nebo i případně útočníkovi zničit příliš mnoho dat
- Na druhou stranu, samozřejmě, se správou výsledků je spojena nějaká administrativa, což se může zdát zbytečné, navíc když se formát dočasných a trvalých výsledků vůbec neliší. Proto je také možné, že nakonec bude myšlenka dvou typů výsledků úplně opuštěna, ale zatím s ní spíš počítám.

11.3. **Možnosti ukládání.**

- Můžeme zvolit ukládání v některé z relačních databází. Má to tu výhodu, že množství problémů s vyhledáváním odpadne „samo“. Na druhou stranu, projekt se chce opírat hlavně o open source databáze, tedy MySQL nebo SQLite, které by mohly mít problémy už jen s velkým množstvím dat.
- Nebo můžeme zvolit kompromis - krátkodobé výsledky (např. za den) ukládat do relační databáze, dlouhodobé výsledky jako plaintext na disk. To také nedává příliš dobrý smysl, protože v krátkodobých výsledcích se tolik nehledá.
- Nakonec je tedy asi nejlepší ukládat vše na disk jako plaintext, resp. jako čistá data, která opět načítám. V následujících podsekcích se chci věnovat tomu, jak budou výsledky uloženy a jak opět načítány.

11.4. **Co chci po systému?** Co od systému pro ukládání dočasných výsledků chci?

- chci rychle ukládat „na konec“
- chci rychle uložit někam k sobě text článku a jeho témata
- chci rychle načíst všechna témata všech článků, které ten den vyšly, pro vytváření dočasných žebříčků (ty budou vytvářeny několikrát za den)
- chci rychle načíst (kvůli zobrazování) aktuální žebříček

Co požaduji od trvalých výsledků?

- chci, aby bylo možné ihned přečíst jména článků

- a aby pro určitý článek bylo možno nalézt témata
- chci, abych rychle našel žebříčky pro den, týden a měsíc
- chci, aby byla jednoduše dohledatelná historie tématu.

K tomu mi slouží poměrně jednoduchá adresářová struktura - mám adresáře `temporary` a `permanent`.

11.5. **Ukládání jednotlivých článků.** V adresáři `temporary` jsou adresáře, pojmenované čísly od 1 do počtu článků, přičemž v textovém souboru `count` je uložen počet článků. V každém adresáři jsou soubory `article`, `themes`, `name`, `link` a `source`. V souboru `article` je extrahovaný článek, v souboru `source` celý HTML zdrojový kód, v souboru `themes` všechna témata, oddělená řádkami. V souboru `link` zptáky link na článek, v souboru `name` jméno článku z tagu TITLE.

Ve skutečnosti je v souboru `themes` každé téma dvakrát - jednou v lemmatizované verzi, jednou v nelemmatizované. To proto, že počítání denního žebříčku je opět přes lemmatizované verze, a opět je lepší dát výsledek nelemmatizovaný (tj. čitelnější).

11.6. **Pomocné soubory.** Dále je v celém dočasném adresáři soubor `article_themes`, kde na každém řádku jsou znovu zopakována témata tolikátého článku, kolikátý je to řádek. Tj. pokud máme články 1-20, je v souboru `all_themes` 20 řádků. Tady nejsou samozřejmě oddělená témata řádkami, ale znakem `|`.

K čemu to je? Abychom pro počítání denního žebříčku nemuseli znovu načítat všechny soubory, stačí nám načíst tento.

Při načtení nového souboru se tedy číslo v `count` zvýší o 1, do nového adresáře se nahrají soubory a na konec souboru `article_themes` se připíší témata.

Dalším souborem je `names`, kde jsou znovu napsané názvy článků, oddělené novou řádkou, pro jejich rychlejší načítání.

11.7. **Dočasný žebříček.** Kromě toho je ještě v adresáři soubor `all_themes`, kde je na každém lichém řádku téma, na sudých rádcích čísla odpovídajících článků. Témata jsou seřazena podle důležitosti. Takto jsou vypsána všechna témata.

Rozdíl souborů `all_themes` a `article_themes` je jednoduchý - zatímco v obou jsou vlastně tytéž informace, v `all_themes` jsou vypsána témata a k nim články, tak v `article_themes` jsou naopak články, a k nim témata. Soubor `all_themes` získáme zpracováním souboru `article_themes`, který už po zaarchivování nebude k ničemu.

11.8. **Trvalé výsledky.** V adresáři `trvalý` jsou adresáře `days`, `weeks`, `months` a `themes`.

Jednou za den se celý adresář `temporary`, s nejnovější verzí žebříčku, přejmenuje na konkrétní datum ve formátu DD-MM-YYYY, přesune do adresáře `days`, případně se jemu a všem podsouborům nastaví read-only atributy.

11.9. **Historie témat.** Do adresáře `themes` se potom uloží *každé* téma, zmíněné v kterémkoliv článku, a to tak, že se uloží datum ve formátu DD-MM-YYYY a k němu počet článků, který tento den na dané téma vznikly. Tímto způsobem se velmi rychle vyrobí historie tématu, která se také rychle zobrazí, aniž by se načítaly všechny články.

Je důležité poznamenat, že do historie témat se ukládají témata *lemmatizovaná*. Pokud poté uživatel hledá některou formu tématu, je nějdřív dotaz lemmatizován a až poté se zjistí, jestli soubor se zadaným jménem existuje, nebo ne.

11.10. **Žebříčky pro periody.** Do adresářů `weeks` a `months` se ukládají pouze žebříčky pro měsíce a týdny, tedy v každém budou sice adresáře pro každý měsíc, ale v každém bude pouze jeden soubor se žebříčkem.

11.11. **Úvaha nad celým ukládacím systémem.** Je pravda, že jsem si vytvořil poměrně složitý systém, co v podstatě spočítá a uloží *předem* veškeré výsledky na disk, a z těch je potom čteno. Je pravda, že by možná bylo lepší, aby se některé z mých výsledků, například ona historie témat, počítaly až po requestu uživatele v CGI skriptu, ale nejsem si jist, jak rychle jsem schopen v textových souborech vyhledat.

Můj systém mi přijde lepší, ačkoliv má jistou redundanci informací. Na druhou stranu, snažil jsem se, aby jak redundance informací, tak časy pro vyhledávání byly co nejmenší i bez použití databází.

12. Vytváření žebříčků

Zatím jsem psal o žebříčcích, bez přílišného vysvětlení, o co jde. žebříček je stupnice témat, seřazených podle toho, jak moc charakterizují daný časový úsek - přičemž při jeho vytváření už mám k dispozici témata, vypočítaná ze všech článků.

Upozorňuji - u vypočítávání témat článku se sice počítá důležitost, ta ale při počítání žebříčků nehraje roli a není vidět. To jest, článek je *o daném tématu*, už není počítáno, jak moc.

U počítání žebříčků beru opět ohled na počet slov. Tentokrát ale o něco víc - ve skutečnosti se víceslovná témata *příliš často neopakují*, kromě jmen. Také chci dát každému opakujícímu se tématu mnohem větší význam.

Opět tedy pro každé téma počítám skóre, tentokrát ale jiným vzorcem. Skóre je počítáno jako $s = n^m$, kde n je počet výskytů a m je počet slov v tématu. Ve skutečnosti je místo m použito $f(m)$, kde f je lineární funkce m. S funkcí f ještě pořád experimentuji, každopádně, vždy platí, že skóre se zvyšujícím se opakováním narůstá rychle, víceslovná témata mají mnohem větší růst, ale při jediném výskytu je skóre stále jedna.

13. Výstup

Výstupem bude HTML stránka, možná s prvky JavaScriptu. Na hlavní stránce bude dočasný žebříček témat za dnešek, už spočítané žebříčky za minulý den, týden a měsíc a možnost vyhledávat historii tématu.

13.1. **Stránka o článku.** Skriptu, který bude vypisovat stránku o konkrétním článku, se dá datum a ID článku. Sám článek se, z copyright důvodů, nikdy nevypíše, co se vypíše je název článku, jeho URL a jeho témata. Samotná témata budou odkazy na stránky o tématech pro konkrétní den, viz další odstavec.

13.2. **Stránka o tématu pro konkrétní den.** Skriptu, který bude vypisovat informace o tématu pro konkrétní den se zadá téma a datum. Skript pak vypíše (přečtením souboru `all_themes`) všechny články, kde se téma vyskytuje.

13.3. **Stránka o tématu pro jiný úsek?** Tak, jak mám navržený ukládací systém, je stránka o tématu na týden/měsíc problematická - musel by se v každém adresáři pro každý den načíst soubor `all_themes`, odkud by se načetly seznam článků pro dané téma. Proto možná tuto možnost vyloučím, na druhou stranu, pokud povolím maximálně měsíční úsek, není to nakonec až tolik přečtení.

13.4. **Stránka o celé historii tématu.** Historie tématu bude 2D graf, nejvíc podobný grafům na `http://trends.google.com`. Pokud se rozhodnu do historie témat (viz 11.9) ukládat i jednotlivé servery, bude složení jednotlivých serverů barevně zvýrazněno.

Důležité je, že v tuhle chvíli se konkrétní články nenačítají, pouze ona historie z adresáře *themes*.

Z této stránky se bude možno dostat na stránky pro téma a konkrétní den, případně na stránku o tématu pro jiný časový úsek.

13.5. **Stránka s žebříčky.** Skript pro žebříčky dostane periodu, pro kterou má zjišťovat (tj. den, měsíc nebo týden), a vypíše několik nejdůležitějších témat. Ke každému z témat bude několik příkladů článků.

Pro každé z témat bude odkaz vést na stránku o historii tématu.

14. Závěr

Nejsem si úplně jist, jestli tento projekt naplňuje původní zadání, tj. rozpoznávání okurkové sezony. Možná ne, a spíš jde jen o systém pro dlouhodobý sběr zpravodajských dat, jelikož samotná analýza zatím neprobíhá.

Je to pravda, ale i samotný sběr většího množství textu a *prosté* počítání opakujících se slov a témat mi přijde jako zajímavé téma, přičemž věci, jako rozpoznávání okurkové sezony nebo analýza korelace jednotlivých témat může přijít ihned poté - naopak, bez většího množství nasbíraných dat nemůžu analyzovat v podstatě nic.