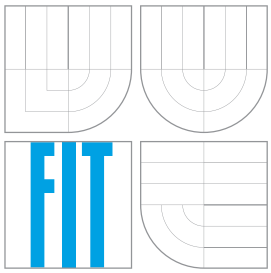
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

# AUTOMATICKÉ HLEDÁNÍ VAZEB MEZI ČÁSTMI AUDIOVIZUÁLNÍCH DOKUMENTŮ

AUTOMATIC LINK DETECTION IN PARTS OF AUDIOVISUAL DOCUMENTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MAREK SYCHRA
AUTHOR

VEDOUCÍ PRÁCE — Ing. IGOR SZŐKE, Ph.D.
SUPERVISOR

BRNO 2015

# Abstrakt

Tato práce se zabývá tématem hledání tématu v textu. Konkrétně hledání spojitostí mezi krátkými texty a hledání hranic jednotlivých částí stejného tématu v jednom hlavním textu. Hlavní motivací výzkumu bylo zavedení do praxe a to v rámci aplikace na přednáškové materiály na FIT (provázání jednotlivých částí různých přednášek). Přístup k porovnávání textů spočívá v analýze textu a slov, která obsahuje a zjišťování významu a důležitosti jednotlivých slov. Segmentace textu toto využívá, když hledá předěly mezi tématy v textu. Obě části problému (*link detection*, *story segmentation*) měly velmi vysokou úspěšnost na testovacích datech (zprávy ze světových novin). Při subjektivním vyhodnocování u částí přednášek byla úspěšnost nižší, ale stále dobrá.



# Abstract

This paper deals with topic detection. Specifically link detection - finding similarities amongst a group of short documents according to their topic and story segmentation - finding borders between two topically different parts in a large document. The main motivation for research was practical application with the use of presentation materials from lectures at FIT (linking parts of different lectures and courses). The solution of link detection is achieved by text and word analysis, which includes learning the meaning and importance of each word. Story segmentation uses this while searching for the boundaries. Both parts of the problem (link detection, story segmentation) gave great results while testing with a standard dataset (world news reports). During evaluation of lecture processing the success rate was lower, but still good.



# Klíčová slova

detekce tématu, hledání vazeb mezi texty, segmentace textu, zpracování přirozeného jazyka, zpracování přednášek

# Keywords

topic detection, link detection, story segmentation, natural language processing, lecture processing



# Citace

Marek Sychra: Automatické hledání vazeb mezi částmi
audiovizuálních dokumentů, bakalářská práce, Brno, FIT VUT v Brně, 2015

# Automatické hledání vazeb mezi částmi audiovizuálních dokumentů

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Szőkeho, PhD.

.....
Marek Sychra
20. května 2015

## Poděkování

Děkuji svému vedoucímu, Ing. Szőkemu, PhD. za čas strávený při konzultacích a nasměrování při nesnázích. Dále děkuji Ing. Josefu Žižkovi za pomoc při praktickém testování.

# Obsah

# Kapitola 1

# Úvod

Tato práce má za cíl nalézt podobnosti mezi částmi přednášek zaznamenaných zde na FIT. Hlavní motivací je vytvořit studijní pomůcku, která studentům dopomůže při učení (hledání podobných částí v rámci jednoho kurzu, např. počítání příkladu - teorie k němu) nebo při pochopení kurzu v širším rozsahu (návaznosti na jiné kurzy).

V prvním kroku je potřeba identifikovat jednotlivá témata v rámci přednášek a v druhém kroku tyto části mezi sebou porovnávat a hledat podobné dvojice. Vstupem do systému je už vytvořený přepis řeči (transkript) k přednáškám. Jedná se tedy o problematiku *topic detection and tracking*, hledání tématu v textu podle slov (různé metody jsou popsány v kapitole 3).

Zvolené metody (postup je detailně popsán v kapitole 4) byly testovány na krátkých anglických zprávách ze světových novin. Při testování porovnávání dvou článků z hlediska jejich tématu jsem dosáhl úspěšnosti kolem 90 %. Co se týče segmentace, je relativní, jakou toleranci by člověk určil při hledání přesného předělu mezi dvěma tématy. Nicméně úspěšnost byla také dobrá (přesné výsledky viz kapitola 5).

Systém byl také nasazen na přednášky ze dvou kurzů na FIT, a poté otestován uživateli (převážně studenty a pracovníky na FIT). V případě kladného ohlasu u výzkumné skupiny Speech@FIT by tento systém mohl být nasazen jako součást webu SuperLectures.com.

# Kapitola 2

# Bližší specifikace problému

Práce jako taková byla zadána s cílem zlepšit studijní stránky s přednáškami SuperLectures[1], které obsahují záznamy z hlavního záznamového serveru FIT[2] a v současné době i z mnoha dalších externích akcí. Stránky samotné obsahují:

- možnost pustit nebo stáhnout si přednášku,
- paralelně sledovat slajdy, které se zrovna promítají v přednášce,
- zapnout si textový přepis obsahu (titulky),
- vyhledávat v přepisu.

Mezi několika vymoženostmi, které chybí, je i nabízení přednášek s podobným obsahem. Jak už v rámci celé přednášky, tak i v jejích jednotlivých částech. Chybí tedy vyhledávání podobných částí v širším kontextu v závislosti na tématu části. Například když se v půlce semestru bude na přednášce počítat několik různých příkladů k půlsemestrální zkoušce, mohly by být studentovi nabídnuty části jiných přednášek, kde se teorie k příkladům probírá. Cílem je tedy provázat jednotlivé přednášky v rámci kurzů (pro lepší pochopení celé látky) a případně napříč rozdílnými kurzy (pro pochopení návazností mezi nimi). Vzhledem k tomu, že stránky zatím obsahují pouze záznamy ze dvou kurzů, orientoval jsem se spíše na první cíl. Podklad, který je vstupem do systému (tzn. prostředek k posuzování podobností) bude přepsaný text přednášky.

## 2.1 Hledání tématu a podobné problémy

Výzkum hledání tématu (*topic detection and tracking* - dále jen TDT) vznikl kvůli narůstajícímu množství informací, které jsou všude okolo nás - především kvůli zprávám (získaným z přepisu řeči). Aby se ulehčilo vyhledávání podstatných a požadovaných informací, je potřeba informace filtrovat a shlukovat dle podobnosti obsahu. Výzkum hledání tématu a problémů podobného zaměření začal velmi expandovat kolem roku 1997. V tomto roce za finanční pomoci společnosti DARPA vznikl program *Translingual Information Detection, Extraction, and Summarization* (TIDES), který byl pro toto odvětví velkým přínosem.

Celá iniciativa rozdělila TDT problém na pět dílčích podproblémů [1, 5]:

[1] http://www.superlectures.com
[2] https://video1.fit.vutbr.cz/

**Story segmentation** Chronologicky první úkol ze všech. Po předložení delšího textu je nutné jej segmentovat na drobnější části (články), které samy o sobě jsou tematicky homogenní (tedy obsahují právě jedno hlavní téma). Výstup tohoto úkolu někdy nutně předchází dalším TDT úkolům.

**First story detection** Po předložení několika článků (získaných segmentací) je při postupném procházení potřeba zjišťovat, které články referují o zcela novém tématu, o kterém jsme z předchozích článků ještě neslyšeli. Každý článek podle TDT obsahuje právě jedno téma.

**Topic tracking** Podle naučeného tématu (naučeným se rozumí s dostatkem předložených článků patřící k danému tématu) se vyhledávají další články tohoto tématu. Vyhledávání by mělo probíhat tematicky nezávisle, tedy nevyužívat vědomostí o přičlenění článků k jiným tématům (oficiální metrika).

**Topic detection** Tento úkol představuje shlukování jednotlivých článků dle jejich domnělého tématu. Jednotlivé nahromaděné skupiny budou představovat různá témata.

**Link detection** Tento úkol by se dal označit jako základ (alespoň částečný) všech ostatních. Jeho cílem je po předložení dvou článků určit, jestli mají stejné téma. Třetí i čtvrtý úkol lze tedy vyřešit pomocí tohoto, není třeba se učit charakteristiku témat, pouze se využitím porovnání jednotlivých dvojic dopracovat k výsledku.

Texty, na kterých byly tyto úkoly v rámci výzkumu TDT testovány byly výhradně seskupeny ze světových televizních zpráv. Díky rostoucím možnostem v oblasti přepisu řeči byly všechny texty strojově přepsány z mluvené řeči. Některé úkoly by také měly být jazykově nezávislé, a proto dalším kritériem pro soubor dat bylo, aby obsahoval texty různých jazyků (strojově přeloženy).

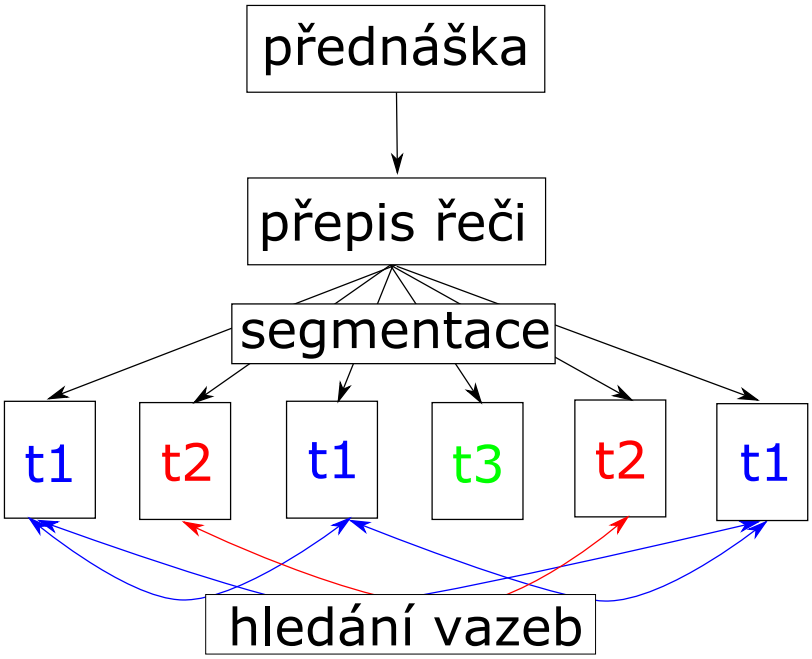
Výzkum TDT u všech úkolů definuje dvě možné chyby: planý poplach (*false alarm - fa*) a zmeškaná detekce (*missed detection - miss*). Ať už je to *link detection* (označení dvou článků stejného tématu za tematicky neshodné - *miss*; označení dvou naprosto různých článků jako tematicky shodné - *fa*), *topic detection* (odmítnutí správného tématu pro článek - *miss*; zařazení článku do špatného tématu - *fa*) nebo *story segmentation* (nenalezení správné hranice - *miss*; určení hranice tam, kde být nemá - *fa*).

Ke komplexnímu problému TDT se dá stavět z několika stran a je podstatné, jaké jsou vstupní informace a co má být výstupem ze systému. Při hledání tématu článku přichází v úvahu vhodný klasifikátor, který je potřeba naučit znakům a rysům jednotlivých témat, aby po vstupu článku do něj mohl rozpoznat příslušné téma. To se například může hodit při potřebě roztřídit zprávy, novinky, apod. Další možnou potřebou je filtrování hlavních klíčových slov, která co nejlépe popisují daný článek; vhodné pro abstrakty a rychlé shrnutí. V neposlední řadě je to i hledání podobností mezi dvěma texty.

## 2.2 Shrnutí problému

Z výše uvedeného výčtu definovaných úkolů pro TDT a zadání plyne, na které části je potřeba se zaměřit. Následující popis je doprovázen obrázkem 2.1.

Převod zvuku přednášky na text není zahrnut v zadání, tím pádem prvním úkolem je z přepisu řeči získat jednotlivé „články“ - části, které obsahují právě jedno téma (jde o úkol *story segmentation*). V přepisu řeči je těžké přesně zjistit, kde se hranice mezi tématy




Obrázek 2.1: Schéma popisující celý proces problému. Na začátku se zaznamená přednáška. Poté je vytvořen přepis řeči (skupinou Speech@FIT[3]), který už slouží jako vstup do systému. Text je dále rozdělen na jednotlivé větší či menší části mající stejné téma. Mezi jednotlivými získanými částmi se poté hledají podobnostní vazby. Zde na obrázku jsou symbolicky vyobrazena tři různá hlavní témata (t1, t2, t3), která přednáška obsahuje.

nacházejí, vzhledem ke spoustě doplňujících slov, nejasnostem v přepisu a především faktu, že téma je relativní pojem. Dalším problémem je také daleko menší množství podstatných slov. Celkově je lidský projev daleko více objemný (co do počtu slov) než odborný psaný článek (je daleko více strukturovaný a stručný). Je potřeba vhodně určit toleranci hranic, kde se dělí dvě různá témata.

Po vygenerování jednotlivých částí přichází problém vyhledávání podobností mezi částmi navzájem - základní TDT úkol *link detection*. Pro účely této práce tedy není potřeba nikterak zjišťovat přesné téma částí, ale spíše pro každou část nalézt několik dalších částí, které jsou dané části co nejpodobnější.

[3] http://speech.fit.vutbr.cz/

# Kapitola 3

# Metody pro hledání tématu

V rámci první studie na téma TDT byly před každý z pěti úkolů postaveny týmy z několika vědeckých skupin (DARPA, Carnegie Mellon University, Dragon Systems, University of Massachusetts) s tím, aby přišly se svým řešením daných problémů. Vzhledem k roku vydání studie (1997) se jedná o jeden z prvních článků na toto téma. Jejich hlavním zaměřením byly vysílané zvukové zprávy mezi lety 1994 až 1995 [1].

V následujících podkapitolách budou představeny některé vybrané prvotní přístupy k jednotlivým úkolům, které prokazovaly dobrou úspěšnost. Budou popsány ale také nové, vylepšené metody, se kterými přišli lidé až po vydání studie.

Nejdříve je potřeba zavést jednoduché názvosloví [1, 5]:

**Událost** představuje v rovině zpráv jednu specifickou událost ve specifickém čase.

**Téma** mělo v první studii stejný význam jako **událost** výše. Později se to však změnilo na seskupení všech událostí, které jsou navzájem nějak spjaty. Tím ovšem ale vzniká problém určení hranice, kam až se určuje podobnost.

**Článek** je jedna zpráva týkající se právě jedné události.

## 3.1 Sestavování vektoru pro články

V každé z metod hledání tématu se nějakým způsobem musí porovnávat články navzájem. Je tedy potřeba je umět nějak reprezentovat a pokud možno rozpoznat, která slova mají větší váhu než ostatní. Přístupů jak tohoto dosáhnout je několik.

### 3.1.1 Reprezentace slov pomocí ID témat

K této metodě je zapotřebí rozdělit si data na trénovací a testovací sety (ideálně aby oba sety obsahovaly stejná témata) [10]. V prvním kroku se pomocí trénovacího setu sestrojí LDA[1] modely pro témata. V nich se zkoumá, ve kterém tématu se která slova objevují nejčastěji. Poté se před modely předloží testovací data a v několika iteracích se ke každému slovu podle kontextu, distribucí témat a pravděpodobností pro slova přiřadí ID nějakého tématu (pro příklad viz obrázek 3.1). Velmi záleží právě na kontextu, slova v různých částech článku mohou mít ID jiného tématu.

[1] LDA - Latent Dirichlet allocation

Fourierova:1 tranformace:1 je:3 integrální:1 transformace:1 převádějící:2 signál:1 mezi:3 časově:2 a:4 frekvenčně:2 závislým:3 vyjádřením pomocí harmonických:1 signálů:1, tj.:5 funkcí:1 sin:4 a:4 cos:5, obecně:3 tedy:4 funkce:1 komplexní:1 exponenciály:1.

Obrázek 3.1: Příklad ohodnoceného textu pomocí ID jednotlivých témat. Modře jsou vyznačena slova úzce spjatá s tématem úryvku (text převzat z Wikipedie - Fourierova transformace).

| **ID témat** | t1 | t2 | t3 | t4 | t5 |
|---|---|---|---|---|---|
| **četnosti** | 11 | 3 | 4 | 4 | 5 |

Tabulka 3.1: Výsledný vektor pro předchozí úryvek.

Pro každý článek je výsledkem vektor, kde indexy jsou ID témat a hodnotami jsou četnosti jednotlivých témat (četnosti slov spjatých s tímto tématem) ve článku.

### 3.1.2 TF-IDF

Metoda *term frequency - inverse document frequency* vzniká spojením dvou metod - *term frequency* (TF) a *inverse document frequency* (IDF) a snoubí potřebné vlastnosti z obou metod. Nejprve se spočítá (relativní) četnost pomocí TF, dále se určí váha slov pomocí IDF a obě hodnoty se vynásobí.

Metoda TF je nejzákladnější metodou pro reprezentování textu vektorem. Mnoho složitějších metod vychází právě z ní. Při zpracování právě dvou textů metoda TF pro každý z nich vytvoří $n$-rozměrný vektor, přičemž $n$ se rovná velikosti sjednocení slov z obou textů. Pro každé slovo z hromadného seznamu slov se na příslušnou pozici vektoru vloží jeho četnost ve článku. Tímto způsobem by metoda vracela uspokojivé výsledky u článků podobné délky. Problém by nastal u článků nesrovnatelné délky, je tedy potřeba výsledná čísla normalizovat. Nabízí se několik možností (výklad znaků rovnic je za rovnicí 3.5):

- výslednou četnost vydělit celkovým počtem slov v dokumentu - relativní četnost,

$$tf(t,d) = \frac{f(t,d)}{|d|} \tag{3.1}$$

- výslednou četnost vydělit nejvyšší četností v dokumentu - relativní četnost,

$$tf(t,d) = \frac{f(t,d)}{max\{f(w,d) : w \in d\}} \tag{3.2}$$

- použít logaritmickou normalizaci.

$$tf(t,d) = \begin{cases} 0, & f(t,d) = 0 \\ 1 + \log_{10} f(t,d), & f(t,d) \in \mathbb{N} - \{0\} \end{cases} \tag{3.3}$$

Nevýhodou metody TF je nepřidělování různých vah jednotlivým slovům. Kupříkladu kdybychom vzali jednotlivé přednášky z matematiky, všude se určitě budou objevovat slova jako „rovnice“, „proměnná“, apod. Kdyby tato slova měla stejnou váhu jako nějaká slova, která se objevují pouze v některé části přednášek, mátlo by to výsledné porovnávání. Nutné řešení tohoto problému proto vylučuje možnost použití této metody samotné.

Metoda IDF bere v potaz globální četnosti slov, co se týče všech zpracovávaných článků. Zohledňuje významovou hodnotu slova, tedy kolik informace v sobě nese, zdali jde jen o běžné slovo, které právě v tomto kontextu porovnávání nikam neposune, nebo jestli jde o slovo, které specificky popisuje charakter článku (např. „derivace“). Je tedy potřeba zpracovávat články hromadně a u jednotlivých slov ukládat, ve kterých článcích se nalézalo. Podle četnosti slova napříč články se určí jeho významová hodnota (časté mají menší) podle následujícího vzorce:

$$idf(t, D) = \log_{10} \frac{|D|}{|\{d \in D : t \in d\}|} \tag{3.4}$$

Po vypočtení obou hodnot se výsledná hodnota TF-IDF získá vynásobením obou hodnot.

$$tf\text{-}idf(t, d, D) = tf(t, d) \times idf(t, D) \tag{3.5}$$

Kde:

| | |
|---|---|
| $t$ | slovo |
| $d$ | článek |
| $D$ | sada článků |
| $f(t, d)$ | četnost slova t v článku d |
| $tf(t, d)$ | hodnota TF slova t v článku d |
| $idf(t, D)$ | hodnota IDF pro slovo t v rámci sady článků D |

### 3.1.3 TF-LLR

Metoda *term frequency - log likelihood ratio* je dalším ze způsobů reprezentace textu s vlastností zjišťování důležitosti slov. Vychází z půlky stejně jako TF-IDF z metody *term frequency* popsané výše. Pracuje s absolutní četností slov ve článku přepočtenou na pravděpodobnost [6]. Pro tuto metodu je zapotřebí mít k dispozici trénovací sadu slov. Pro každé slovo $w$ se jeho hodnota $x_w$ počítá následovně:

$$x_w = \frac{P(w|d)}{\sqrt{P(w)}} \tag{3.6}$$

$$P(w) = \frac{N_w + 1}{N_W + N_V} \tag{3.7}$$

Kde:

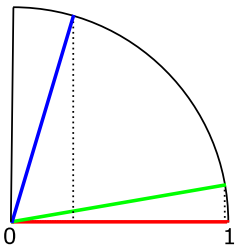
| | |
|---|---|
| $w$ | slovo |
| $d$ | článek |
| $P(w\|d)$ | pravděpodobnost při vybrání náhodného slova z $d$, že to bude $w$ |
| $N_w$ | absolutní četnost $w$ v trénovacím setu |
| $N_W$ | celkový počet slov v trénovacím setu |
| $N_V$ | celkový počet různých slov v trénovacím setu |
| $x_w$ | výsledná normalizovaná hodnota pro $w$ |

## 3.2 Hledání vazeb

Tento úkol (*link detection*) je povětšinou i základem ostatních úkolů. Jde o to poznat podobnost mezi dvěma články. Dělí to tento úkol na dvě hlavní části: 1) zajistit vhodnou reprezentaci článku (viz sekce 3.1) a 2) najít způsob, jak dvě instance zvolených reprezentací porovnat.

### 3.2.1 Cosine similarity

*Cosine similarity* je běžná metoda na porovnání podobnosti dvou vektorů. Vychází z matematického principu výpočtu cosinu úhlu mezi dvěma vektory. Čím si jsou vektory podobnější, tím budou svírat menší úhel. Čím menší úhel, tím větší cosinus. Účelem je zjistit míru podobnosti dvou článků přepočtenou na normalizovanou hodnotu.




Obrázek 3.2: Na obrázku lze vidět část jednotkové kružnice[2]. Pro transparentnost leží referenční (červený) vektor na ose x. Červený vektor a modrý vektor si nejsou skoro podobné, svírají velký úhel a jejich cosinus je malý. Na druhou stranu červený a zelený si jsou velmi blízké, a proto jejich výsledný cosinus leží skoro u jedničky.

$$cossim(A, B) = \frac{\sum_{i=1}^{n}(A_i \times B_i)}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3.8}$$

V čitateli je v podstatě skalární součin vektorů a ve jmenovateli je součin jejich velikostí. Výsledná hodnota $[0, 1]$ udává míru podobnosti. Vzhledem k nezáporným hodnotám získaných pomocí některých z popsaných metod (viz sekce 3.1) není možné získat úhel větší než $90\,^\circ$ (cosinus tedy nebude záporný). Dva vektory svírající úhel $0\,^\circ$ jsou totožné, podobnost je nejvyšší a cosinus úhlu, který svírají je 1. Naopak diametrálně odlišná dvojice (s úhlem kolem $90\,^\circ$) bude mít za výsledný cosinus 0.

### 3.2.2 Pearson product-moment correlation coefficient

Tato metoda (zkráceně PPMCC) je další způsob, kterým se dá měřit podobnost dvou objektů (vektorů) [4]. Vychází z velké části z *cosine similarity*, přičemž se liší právě pouze v normalizaci dat.

[2]Při porovnání dvou vektorů jednotkové délky lze získat výsledný cosinus jejich úhlu spuštěním kolmice z koncového bodu druhého vektoru. Kde protne osu x, tam leží výsledná hodnota.

$$ppmcc(A,B) = \frac{\sum_{i=1}^{n}(A_i - \bar{A}) \times (B_i - \bar{B})}{\sqrt{\sum_{i=1}^{n}(A_i - \bar{A})^2} \times \sqrt{\sum_{i=1}^{n}(B_i - \bar{B})^2}} \tag{3.9}$$

Každá hodnota v obou vektorech je normalizována průměrem celého vektoru. Tentokrát už je interval pro výsledky $[-1, 1]$ (může vzniknout záporná hodnota posunutím průměrem).

## 3.3 Přiřazování k tématu

Tento úkol (*topic tracking*) představuje situaci, kdy je k dispozici charakteristika nebo popis jednotlivých témat (např. několik cvičných článků referujících o jednom daném tématu). Pomocí toho je potřeba porovnávat nové, příchozí články, zdali také nepatří k tomuto tématu.

### 3.3.1 kNN

Použití metody $k$-Nearest Neighbors na přiřazování článků k tématům bylo v rámci studie předvedeno univerzitou Carnegie Mellon (dále CMU). Každý zpracovávaný článek je převeden na vektor a porovná se s uloženými naučenými články (naučenými tím způsobem, že ví, ke kterému tématu patří nebo nepatří). Výsledky porovnání se seřadí a vybere se $k$ nejpodobnějších (CMU vybírá nejpodobnější podle metody *cosine similarity* - viz sekce 3.2.1). Každý z nich poté určí, jestli je porovnávaný článek stejného tématu jako má on. Určuje se to podle předem určeného prahu (angl. *threshold*), kdy když je skóre porovnání nad ním, článek hlasuje pro stejné téma. Téma článku se poté určí podle většiny pozitivních hlasů [1].

Metoda klasifikace kNN patří mezi nejjednodušší z algoritmů pro zařazování objektu do tříd a používá se i v mnoha jiných odvětvích než jen v oblasti TDT. Vždy je jen potřeba najít vhodnou metodu, pomocí které se určí vzájemná vzdálenost porovnávaných objektů.

### 3.3.2 Bayesovská klasifikace

Použití *Naive Bayes Classifier* ukazuje T. J. Hazen ve své práci, kde zpracovává telefonní konverzace [6].

Každý článek může být popsán vektorem četností slov, které obsahuje. Některá však mají větší váhu jak ostatní (ať už globálně, či v rámci témat). Z naučených informací o tom, které trénovací články mají které téma je tedy možné určit, která slova jsou pro dané téma nejpodstatnější. Pro každé téma se vybere $n$ nejčastějších slov ($n$ se experimentálně zjistí), která se v dokumentech tohoto tématu objevují a nakonec se dají dohromady. Při tvoření vektorů se poté berou v potaz pouze tato slova. Dále je potřeba hodnoty normalizovat, aby se vyrovnal rozdíl mezi dlouhými a krátkými články. Po normalizaci se nasadí jedna z váhovacích metod viz sekce 3.1.

Po počátečních úpravách se bere každý článek a počítá se skóre proti každému tématu.

$$S(d,t) = \sum_{\forall w \in d} x_w \log \frac{P(w|t)}{P(w|\bar{t})} \tag{3.10}$$

$$P(w|t) = \frac{N_{w|t} + N_V P(w)}{N_{W|t} + N_V} \tag{3.11}$$

Kde:

| | |
|---|---|
| $w$ | slovo |
| $d$ | článek |
| $t$ | téma |
| $x_w$ | viz rovnice 3.6 |
| $P(w\|t)$ | pravděpodobnost pro $w$, že patří k $t$ |
| $P(w)$ | viz rovnice 3.7 |
| $N_{w\|t}$ | absolutní četnost $w$ v dokumentech patřících k $t$ |
| $N_{W\|t}$ | celkový počet slov v dokumentech patřících k $t$ |
| $N_V$ | celkový počet různých slov v trénovacím setu |

$P(w|\bar{t})$ se počítá obdobně jako $P(w|t)$, pouze s články nepatřícími k tématu $t$. Podle nejvyššího výsledného skóre se článek zařadí k příslušnému tématu.

## 3.4 Hledání tématu

Cílem tohoto úkolu (*topic detection*) je rozdělení vstupních dat v závislosti na jejich tématech. Nestačí porovnávání dvojic, které jsou si podobné. Také se tento úkol liší od předešlého podstatnou věcí a tou je absence trénovacích dat; pro potenciální klasifikátor tedy nejsou data, pomocí kterých by se naučil rysy a znaky jednotlivých témat.

Přesto se tento úkol může zdát jako nejpoužitelnější v praxi. Použití může být například filtrování emailových zpráv nebo i shlukování novinek podle událostí.

V praxi a postupu se využívá jiného TDT úkolu, detekce prvního článku (*first story detection*), který spočívá v rozhodnutí, zdali právě zpracovávaný článek patří do některé z již vygenerovaných skupin, nebo jestli tvoří nové, neznámé téma.

### 3.4.1 Rostoucí shlukování

Metoda zvaná anglicky *Incremental clustering* byla v pilotní TDT studii předvedena CMU. Používá vektorový model pro články: každý článek je reprezentován vektorem s jedním rozměrem, jehož indexy jsou ořezaná slova či fráze článku a složkami vektoru jsou TF-IDF (viz 3.1.2) hodnoty. Jednotlivé shluky představující témata jsou zastoupena prototypovými vektory, tzv. centroidy, což jsou normalizované součty všech vektorů spadajících do tohoto shluku.

Po vytvoření vektorů pro články nastupuje na řadu samotné shlukování. Porovnávání mezi články navzájem je v této metodě realizováno pomocí metody *cosine similarity* (viz 3.2.1). Algoritmus pro každý článek postupuje následovně:

1. Vezmi článek a porovnej ho se všemi centroidy.
2. Pokud je hodnota porovnání nejbližšího centroidu větší jak předem určená hodnota, jdi na 3., jinak na 4.
3. Článek patří do tohoto shluku, přepočítej hodnotu centroidu a vezmi další článek.

4. Článek nepatří do žádného ze shluků, je tedy potřeba udělat nový shluk a z článku jeho centroid.

### 3.4.2 Shlukování pomocí klíčových slov

Tato metoda má zjednodušeně dva kroky: nejdříve se z každého článku vyextrahují ta nejdůležitější klíčová slova, poté se hledají shluky slov a pro každý shluk se hledá střed reprezentující jedno téma [12].

Samotnému výběru klíčových slov předchází předzpracování textu. To zahrnuje lemmatizaci (ořezání slova až na jeho kořen), vyhledání víceslovných klíčových frází a rozpoznání entit. Výběr klíčových slov se rovná výběru těch nejčastějších slov (s vyfiltrovanými slovy viz výše).

Dalším krokem je porovnávání jednotlivých slov. Je nutné vybrat takové porovnávání, které dovoluje počítat vzdálenosti (pro shlukování - viz dále) a počítat středy mezi jednotlivými slovy. Ve svém článku autoři představují čtyři funkce, pomocí kterých získávají vzdálenosti mezi slovy:

- *Cosine similarity* s distribucemi klíčových slov. Zjednodušeně se pro každý článek počítá šance, že se slova nacházejí v něm a dělí se odmocninou součtu druhých mocnin těchto hodnot. Nejde tedy o klasickou vzdálenost dvou vektorů.
- *Cosine similarity* využívající vektory s TF-IDF hodnotami pro klíčová slova (podrobněji popsáno v 3.2.1).
- Jensen-Shannonova (dále jen J-S) divergence mezi distribucemi dokumentů.
- J-S divergence mezi distribucemi slov.

V obou případech J-S divergence se používá také relativní entropie nebo Kullback-Leiblerova divergence. Nebudeme zacházet do detailů, pro bližší pochopení přesného výpočtu J-S divergence viz celý článek [12].

Shlukování je realizováno pomocí modifikované metody *bisecting k-means*. Metoda postupuje následovně:

1. Označ dva články s největší vzdáleností (nejrozdílnější) jako středy.
2. Všechny ostatní články přiřaď k jejich nejbližšímu středu.
3. Spočítej nové středy shluků ze všech článků patřících do shluku.
4. Opakuj kroky 2. a 3., dokud se oba středy posouvají o určitou vzdálenost.
5. Pokud je velikost některého shluku větší než předem určená hodnota, aplikuj na něj celou metodu rekurzivně.

## 3.5 Segmentace textu

Cílem úkolu *story segmentation* je automatické rozdělení velkého vstupního textu na drobnější části, kde každá část by se dala označit jako jeden tematický blok. Tento úkol může předcházet všem dříve uvedeným úkolům a je velmi podstatný i k dosažení cíle této práce.

### 3.5.1 C99

Celý text je rozdělen na věty a pro každou je vytvořen vektor pomocí metody popsané v sekci 3.1.1 [3, 8, 10]. Vytvoří se velká matice $S$ obsahující hodnoty vzájemných porovnání (pomocí *cosine similarity* - viz 3.2.1) mezi větami. Dále se vytvoří matice $R$, která obsahuje pro každou hodnotu v $S$ počet jejích sousedů, kteří mají menší hodnotu než referenční hodnota. Toto je zavedeno kvůli zvětšení kontrastu mezi potenciálními částmi. Poté je nasazen hierarchický algoritmus, který pracuje shora dolů a na základě hodnot v maticích $S$ a $R$ postupně rozděluje celý článek (jeden segment) na více segmentů, dokud se nedosáhne koncových podmínek (počet segmentů nebo nepřesáhnutí prahu potřebného pro rozdělení jednoho segmentu).

Nevýhodou této metody může být, že nepracuje s širším kontextem, než je jedna věta a porovnává zbytečné dvojice (nesouvisející).

### 3.5.2 TextTiling

Tato metoda nerozděluje vstupní text po větách, ale po tzv. „tematických sekvencích“, které se skládají z $n$ výskytů nějakého tématu (výskytů slov patřících k tématu) za sebou (i přerušovaně) [8, 10]. Dále pro výpočet podobnosti mezi každými dvěma sousedícími sekvencemi se vezme $k$ dalších sekvencí vpravo i vlevo. Tím se vytvoří dva větší bloky a jejich vektory se porovnávají pomocí *cosine similarity* (viz sekce 3.2.1). Pro každou výslednou hodnotu se spočítá hloubkové skóre (angl. *depth score*), které určuje, o jak velké minimum se jedná.

$$d_i = 1/2 \times (hl(i) - s_i + hr(i) - s_i) \tag{3.12}$$

Kde $d_i$ je skóre, $s_i$ je hodnota podobnosti v daném místě a $hl$ a $hr$ jsou hodnoty nejbližších maxim nalevo a napravo. Tyto hodnoty jsou seřazeny a od nejlepšího se začíná segmentovat. Přestane se pokud se překročí daný počet segmentů, nebo když je skóre menší než zadaný parametr.

### 3.5.3 TopicTiling

Metoda vychází z předchozí (sekce 3.5.2), pouze s několika drobnými úpravami [10]. Jako základní blok bere věty a jejich vektory získává pomocí ID témat (viz sekce 3.1.1). Neuvažuje ale žádný kontext - porovnává pouze dvě sousedící věty. Když dojde na poslední krok, hledání předělů, TopicTiling vybere nejdříve pouze minima a pro ně počítá hloubkové skóre jako v TextTiling. Výsledné hodnoty se seřadí a dokud nejsou splněna koncová kritéria, tak se text segmentuje podle zjištěných indexů.

### 3.5.4 ClustSeg

Metoda ClustSeg rozvíjí zajímavou myšlenku - bere jako základní blok celý odstavec [8, 9]. Počítá s tím, že odstavec je tou nejmenší dostačující jednotkou pro vyjádření nějakého tématu. Zároveň předpokládá, že každý blok může patřit k více tématům (kvůli předpokladu, že v průběhu výkladu se autor může vrátit k jinému, už uzavřenému tématu). Pro každý blok se tedy vygeneruje příslušný vektor. Metoda má tři hlavní kroky:

1. Nejdříve se použije shlukovací algoritmus, který zvládá pracovat s překryvy (v práci autoři používají ICSD[3] [11]). Výstupem je množina shluků odstavců.

[3] ICSD - *Incremental Clustering by Strength Decision*

2. V druhém kroku se zpracovává výstup z prvního kroku. Zjišťuje se, jak moc si skutečně odstavce ve shlucích jsou podobné a jestli celý shluk splňuje vstupní podmínky.

3. V posledním kroku se zpracují právě získané segmenty. Kvůli tomu, že vznikly jako výstup z postupů, které pracují s překryvy, mohou být výsledné segmenty taky překryté. Každé dva segmenty, které mají alespoň jeden odstavec společný se spojí.

Slabé místo metody spočívá v nutnosti zvolit správně vstupní práh, který bude určovat, kdy si odstavce jsou podobné a kdy ne. Taktéž má metoda velmi úzké uplatnění - potřebuje pro svůj běh vstupní data, která jsou správně formátována a obsahují rozdělení na odstavce.

### 3.5.5 Zhodnocení metod

Každý přístup má své pro a proti. Metoda C99 nebere v potaz širší kontext, obsahuje zbytečné výpočty a má větší nároky než jiné metody. Metoda TextTiling představuje zajímavé vytváření základních bloků a pracuje s kontextem. Když ale dojde na zjišťování předělů, vypočítává zbytečné hodnoty a nezaměřuje se na podstatné výpočty. Metoda TopicTiling stejně jako TextTiling používá hloubkové skóre, což může u obyčejného porovnávání velmi pomoci. K výsledným předělům přistupuje lépe, ale zase nepoužívá širší kontext (pouze porovnání jedna a jedna věta). Poslední metoda, ClustSeg, má především velmi specifické uplatnění. Vzhledem k tomu ji ani nebudu testovat v práci.

# Kapitola 4

# Použitý přístup

V minulé kapitole byly ukázány a analyzovány různé existující metody pro řešení TDT problémů. Některé měly podobné části, ale všechny měly stejné předzpracování dat. V této kapitole budou uvedeny obecné potřebné kroky, které nechybí v žádném ze správných přístupů (a které zároveň byly použity v této práci) a bude podrobněji popsáno, jak jsem přesně postupoval při dosažení svého cíle.

Celý přístup jsem nejdříve testoval pomocí TDT datasetu (více o něm v sekci 4.1). Proto budou někde uvedeny dva přístupy, vzhledem k charakteru dat (u TDT datasetu jsem měl k dispozici trénovací data, u přednášek ne). Výsledky z TDT datasetu také šly objektivně posuzovat, určovat chyby a podle nich trénovat algoritmus. Postup tedy byl zjednodušeně: natrénovat algoritmus co nejlépe na datasetu, kdy je možné výsledky objektivně hodnotit a pak jej zkusit použít na přednášky a pouze subjektivně posoudit úspěšnost.

## 4.1 Testovací dataset

Dataset, který jsem dostal k dispozici od skupiny Speech@FIT se jmenuje *TDT5 Multilingual News Text*[1]. Tento dataset byl vytvořen v roce 2004 organizací *Linguistic Data Consortium* ve spolupráci s programem *DARPA TIDES*. Jeho obsahem je souhrn (zhruba 400 tisíc) krátkých zpráv ze světových novin rozdělených do 250 skupin podle toho, ke které události se vážou. Každá zpráva patří právě k jedné události. Texty jsou ve třech jazycích (angličtina, arabština, čínština), z nichž všechny neanglické jsou strojově přeloženy do angličtiny. Zprávy jsou různě dlouhé, od velmi krátkých (zhruba 30 slov) po velmi dlouhé (2500 slov). Lze si tedy ze široké škály vybrat, jaké se nejlépe hodí a co přesně člověk chce testovat.

Dataset dále obsahuje tři seznamy obsahující relevance. První seznam obsahuje zápis vztahu mezi dvojicí různých článků a výsledek jejich relavance vůči nějakému tématu (ano/ne). Druhý seznam obsahuje výpis témat a pro každé definuje články, které s ním nemají nic společného. Poslední seznam je asi šestnáctinový oproti předchozímu, ale obsahuje právě informace potřebné pro vhodné testování a experimentování s metodami - pro každé téma výčet článků, které se ho týkají.

Pro svou práci jsem si vybral kolem 50 různých témat a k nim příslušné články (množství záviselo na tom, co jsem zrovna testoval). Podstatnou věcí ale bylo, že vybrané články jsem rozdělil na dvě skupiny - trénovací a testovací sety. Před každým testováním jsem vybral poměr, kterým vybrané články budu dělit a pro každé téma jsem je rozdělil. Bylo podstatné,

[1] katalogové číslo LDC2006T1

aby se články rovnoměrně dělily pro každé téma, aby se dalo naučit něco o každém tématu. Např. by byla chyba, kdybych všechny články z tématu 1 zařadil na trénování a všechny články z t. 2 na testování. O t. 2 bych se nic nenaučil a naučené vědomosti by mi nebyly moc platné.

Pro testování segmentace textu jsem vzal několik článků a pospojoval je za sebe, aby vznikl jednolitý text. Pak jsem se pokoušel hledat předěly mezi jednotlivými články.

## 4.2 Předzpracování dat

Jak bylo zmíněno, většina metod nepracuje s textem takovým, jaký je, ale nejdříve si ho zpracuje. Pro zpracování je potřeba vyřešit několik věcí:

**Interpunkce a speciální znaky** Pro algoritmus zpracovávající slova by pomlčka, čárka nebo vykřičník působily jako normální slova. Je tedy potřeba text nejdříve oprostit od těchto značek.

**Ořezávač**[2] Člověk pochopí, že jedno slovo v jednotném čísle a množném nesou tu stejnou informaci (o tom, co je to za slovo). Stejně tak je velký problém se skloňováním v češtině. Pro stroj je potřeba tato všechna slova spojit do jednoho - vhodným algoritmem ořezat předpony a přípony, aby zbyl jen kořen. Například aby „počítač“ a „počítače“ byly namapovány do stejného objektu. Výsledkem bude jednak zmenšení výsledného vektoru, ale také větší přesnost porovnání. Pro angličtinu jsem používal Snowball stemmer[3], přesněji řečeno jeho odnož pro Python, PyStemmer[4]. Snowball obsahuje i český ořezávač, ale PyStemmer už ne, našel jsem tedy jiný, od autora jménem Luís Gomez[5].

**Seznam stop-slov** V každém jazyce se objevuje spousta slov, která pouze dotváří větu, spojují ji a případně zabarvují. Jsou to sice také podstatná slova, bez kterých by věta nedávala smysl, ale nijak více nerozvíjí to, o čem věta je. Jedná se především o zájmena, spojky, předložky, číslovky, členy (angličtina), ... Ale také i různé příslovce a přídavná jména, záleží na konkrétním datasetu. Všechna tato slova je potřeba před samotným zpracováním vyřadit, protože nás nijak blíže neposunou v hledání významu věty.

V rámci anglického jazyka jsem to řešil sesbíráním všech různých seznamů, co jsem našel na internetu a jejich sjednocením. Dále jsem seznam prošel, přidal, co by mě ještě napadlo a postupně s testováním jsem přidával další a další slova. V češtině to bylo podobné, ale tvary byly velmi rozdílné (ořezávač vždy není dokonalý), takže český seznam musel být daleko větší a musel počítat s různými tvary slov, která vystoupí z ořezávače. Po několika spuštěních samozřejmě následovala analýza a doplnění seznamu (objevily se speciální řetězce získané právě z přepisu řeči).

Po aplikování těchto úprav je text oproštěný od nepodstatných slov a pokud možno všechna slova se stejným kořenem brána jako jedno a totéž. Chronologicky následuje úkol segmentování textu na drobnější články, ale protože v samotné segmentaci jsou použity stejné postupy jako v porovnávání, bude popsáno nejdříve porovnávání dvou článků.

[2] angl. stemmer
[3] http://snowball.tartarus.org/
[4] https://github.com/snowballstem/pystemmer
[5] http://research.variancia.com/czech_stemmer/

## 4.3 Porovnávání dvou textů

Jak bylo zmíněno v sekci 3.1, pro porovnání článků je potřeba je umět reprezentovat. Vzhledem k dobrým výsledkům v článcích prezentujících metodu TF-IDF a svým výsledkům jsem se rozhodl právě pro ni (výsledky porovnání jednotlivých metod jsou v sekci 5.1.1). Pro přednášky mnoho možností nebylo, referenční (trénovací) data k dispozici nejsou, takže se musí vycházet pouze z četnosti slov v bloku a váhování pomocí IDF. Také bylo potřeba normalizovat dané hodnoty. Zvolil jsem relativní normalizaci pomocí celkového počtu pro TF (otestováno viz sekce 5.1.1).

Po sestavení vektorů je potřeba je porovnat. Znovu kvůli kladným ohlasům v článcích a faktu, že ve většině příkladů je nasazena metoda *cosine similarity* (popsána v sekci 3.2.1) jsem se rozhodl tuto metodu použít. Jediným problémem je určení potřebného prahu, který bude rozlišovat, která hodnota ještě označuje podobné články a která už ne. V případě testování s TDT datasetem jej lze získat při prvním běhu: při učení se najde práh, u kterého je nejmenší chyba a ten se použije u testovacího setu. Avšak během porovnávání částí přednášek jej nelze nijak objektivně získat. Šlo by to maximálně subjektivně podle odpovědí od lidí, kteří budou části ručně posuzovat (myšlenku rozvíjím v sekci 5.2). Naštěstí se bez něj lze relativně obejít. Cílem totiž není vyhledat všechny podobné části, ale jen několik nejbližších, takže stačí výsledky pro každou část seřadit a vybrat těch pár nejlepších.

## 4.4 Segmentace

Přístup k segmentaci textu obsahuje celý problém porovnávání dvou textů (jak bylo popsáno v sekci 4.3) v sobě. Pro získání jednotlivých segmentů je potřeba:

- rozdělit si vstupní text na pracovní bloky (a získat potenciální předěly),
- porovnat okolí možných předělů a určit užší výběr možných hranic mezi tématy,
- definitivně podle zvolené metriky určit, které hranice se vezmou a které nechají být.

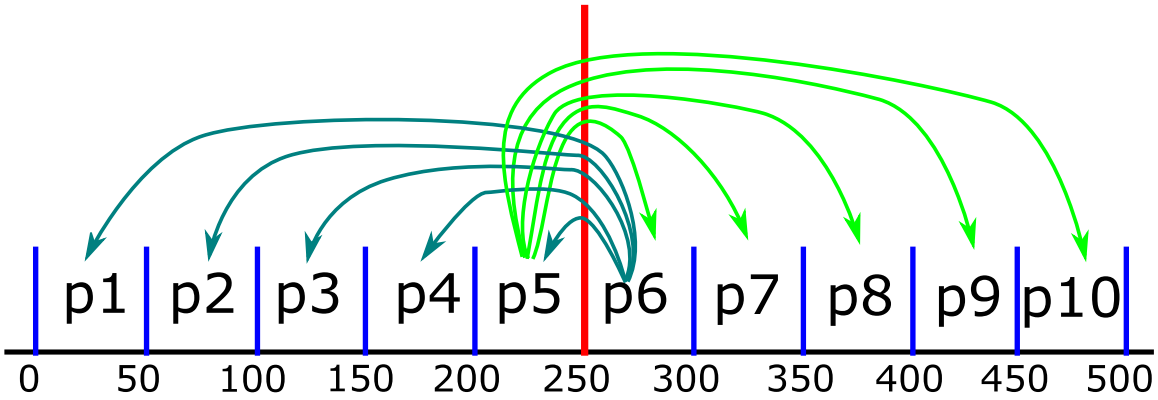
Můj přístup vychází ze spojení dvou metod, *TextTiling* a *TopicTiling* (viz sekce 3.5.2 a 3.5.3). Nejprve jsem rozdělil celý text na spoustu drobných částí - například pár delších vět (50 až 90 slov). Dělit pouze na věty se příliš neosvědčilo, protože věty v datasetu mohou být 1) různě dlouhé (v takovém případě se hůře porovnává podobnost) a 2) v přepisu přednášky žádné věty nejsou. Čím větší bloky, tím přesnější porovnávání.

Dále mezi každými dvěmi zjišťuji, jaká je jejich podobnost pomocí přístupu popsaného v minulé sekci 4.3 (i s větším kontextem - viz obrázek 4.1). Nicméně pokud bychom uvažovali délku základního bloku 90 slov (průměrná délka jedné zprávy v TDT5 je zhruba $5 \times 90$ slov), dostaneme sice vyšší a přesnější hodnoty, ale zároveň je šířka dostatečně velká, aby případná hranice mohla padnout právě doprostřed (tudíž zůstane nenalezena).

Proto uvažuji techniku posuvného okna[6], která zachovává větší délku základního bloku a zároveň nepřijdu o případné hranice uvnitř bloku. Technika funguje následovně:

1. Zvolím počet slov, po kterých chci testovat předěl témat (já zvolil 10 slov - označme tuto konstantu $shift$). Místo, odkud se postupně budou tvořit bloky (označme jej $start$) je na začátku textu. Velikost základního bloku budiž 90 slov (označíme $block$).

[6] angl. sliding window




Obrázek 4.1: Hranice (červená čára) se mezi dvěma bloky slov (slova - černá osa, jeden blok - část mezi dvěma modrými čarami) nevyhledává pouze porovnáváním dvou sousedních bloků. Část nalevo (p5) se porovnává až s pěti dalšími částmi za hranicí. Stejně tak část p6 v opačném směru. Výsledný součet se zprůměruje a uloží i s indexem předělu.

2. Spočítám počet aplikací posuvného okna ($N$): $N$ je nejmenší společný násobek posunutí a délky základního bloku. Následující kroky provedu $N$-krát.
3. Od místa *start* začnu postupně po *block* slovech vytvářet jednotlivé bloky.
4. Vypočítám pravděpodobnosti výskytu hranic mezi bloky a uložím i s indexy.
5. Místo *start* posunu o $shift$ slov a jdu na krok 3.

Postupně se zvětšující menší blok na začátku textu (vznikající posunem místa *start*) v popisu algoritmu neuvažuji - udělám z něj jen další blok.

Výsledkem tedy bude graf hodnot cosinů (viz např. graf 5.4), kde hodnota bude zaznamenaná po každých $shift$ znacích. Z grafu vyberu všechna minima a mám dvě možnosti podle čeho určovat, která vzít nebo ne:

- podle vypočtené hodnoty minima,
- podle hloubkového skóre minima (viz sekce 3.5.2).

V obou případech potřebuji nějaký práh, který bude určovat, která minima skutečně vezmu a která ne. Výsledky a úspěšnost obou možností vyhodnocuji v sekci 5.1.2.

## 4.5 Hodnocení

Jak už bylo popsáno v sekci 2.1, vzniklé chyby v obou úkolech jsou pouze dvě možné: něco nenalezeného (*missed detection*) a planý poplach (*false alarm*). Co která chyba znamená pro který z úkolů je shrnuto v tabulce 4.5.

Pro každý úkol se výsledná číselná úspěšnost/chybovost určuje jinak.

### 4.5.1 Metrika pro porovnávání

Výsledkem porovnávání je seskupení dvojic cosinus-výsledek, přičemž určený výsledek je buď „stejné“/„jiné“ (téma). Tyto hodnoty je potřeba přepočítat na jednu hodnotu, která bude jasně určovat úspěšnost. Chybu jsem počítal podle následujícího vzorce [5, 7]:

| | *missed detection* | *false alarm* |
|---|---|---|
| porovnávání | označení porovnání dvou textů stejného tématu jako rozdílné | označení porovnání dvou textů jiného tématu jako podobné |
| segmentace | nenalezená správná hranice | určení hranice tam, kde nemá být |

Tabulka 4.1: Shrnutí významu chyb pro dva TDT úkoly.

$$C_{Det} = (C_{Miss} \times P_{Miss} \times P_{Target} + C_{Fa} \times P_{Fa} \times (1 - P_{Target})) \tag{4.1}$$

Definujme si dvě hodnoty: celkový počet dvojic určených jako stejné jako *sameTotal* a celkový počet dvojic označených jako jiné jako *diffTotal*. Potom lze získat parametry do rovnice 4.1 následovně:

| | |
|---|---|
| $C_{Miss}$ a $C_{Fa}$ | váhy chyb jednotlivých typů |
| $P_{Miss}$ | pravděpodobnost výskytu *missed detection*: vypočteno #miss/*sameTotal* |
| $P_{Ffa}$ | pravděpodobnost výskytu *false alarm*: vypočteno #fa/*diffTotal* |
| $P_{Target}$ | šance pro článek na nalezení článku stejného tématu |

Čím blíže je hodnota k 0, tím je větší úspěšnost. Hodnoty $C_{Miss}$ a $C_{Fa}$ se pro většinu TDT úkolů určují v poměru 10:1, tedy $C_{Miss} = 10$ a $C_{Fa} = 1$ [5]. Hodnotu $P_{Target}$ pro každou testovanou skupinu vypočítávám zvlášť. Pro každé téma, které skupina obsahuje, vypočtu šanci výskytu a po získání hodnot od všech témat výslednou hodnotu zprůměruji. Výslednou hodnotu je ale ještě potřeba normalizovat menším ze dvou extrémních výsledků - když by se odpovídalo na každé porovnání a) „stejné téma“, nebo b) „jiné téma“:

$$(C_{Det})_{Norm} = C_{det}/MIN(C_{Miss} \times P_{Target}, C_{Fa} \times (1 - P_{Target})) \tag{4.2}$$

### 4.5.2 Metriky pro segmentaci

Pro zhodnocení výsledků segmentace existuje více metrik, vzhledem k horší objektivnosti při uchopení problému. Podstatnou věcí je si určit, co by mělo být výstupem ze systému. Záleží, jestli je cílem najít všechny hranice za cenu několika nadbytečných, nebo na druhou stranu nalezení co nejvíce správných, ale s žádnými nadbytečnými. Tohle vše jde také ladit pomocí váhování jednotlivých chyb.

#### F-measure

První metrika se nazývá *F-measure* a pracuje se dvěma dalšími hodnotami, *Precision* a *Recall* [8]. Používá se při přímočarém zjišťování, když chceme zjistit, jestli nalezené hranice přesně korelují s předlohou. Hodnoty se počítají následovně:

**Precision** určuje procento správně určených hranic ze všech určených.

$$precision = \frac{\text{počet správně určených hranic}}{\text{počet všech určených hranic}} \tag{4.3}$$

**Recall** určuje procentuálně kolik z referenčních hranic se našlo.

$$recall = \frac{\text{počet správně určených hranic}}{\text{počet všech definovaných hranic}} \tag{4.4}$$

**F-measure** je spojení dvou předchozích hodnot.

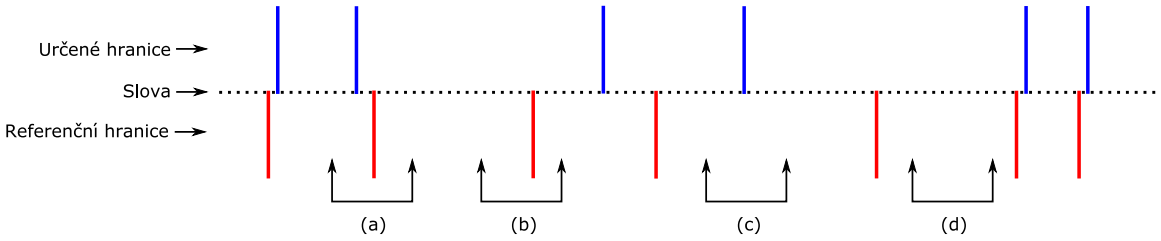
$$F\text{-}measure = 2 \times \frac{precision \times recall}{precision + recall} \tag{4.5}$$

Tato metrika má ovšem jeden nedostatek. Buď se vyhledávají skutečně pouze naprosto přesné hranice (pokud by se hranice určila byť o větu vedle, už je to špatně), nebo se určí jistá tolerance, při které se hranice vezme ještě jako validní. Při druhé možnosti ale metrika nerozlišuje mezi dvěma řešeními, např. když by první algoritmus našel hranici vždy naprosto přesně a druhý by se vždy velmi těsně dostal do tolerance. Obě řešení by tedy z pohledu *F-measure* byly stejně dobré.

Čím je tato hodnota vyšší, tím je algoritmus přesnější.

### $P_k$ metrika

Zajímavou metrikou je právě $P_k$, která nekontroluje ani tak absolutní přesnost určených hranic, jako spíše dvojice vět (jestli patří do stejného segmentu) [2, 8]. Výsledné číslo je ve skutečnosti pravděpodobnost, že když se náhodně zvolí dvě různé věty z článku, budou správně označeny jako patřící/nepatřící do stejného segmentu.




Obrázek 4.2: Obrázek ukazuje 4 možnosti, které mohou při testování nastat (porovnávají se dvojice, na které ukazují šipky stejné čáry): (a) systém rozpoznal správně, že vybrané věty nepatří do stejného segmentu, (b) systém určil špatně, že věty patří do stejného segmentu (miss), (c) systém určil špatně, že věty patří do jiného segmentu (false alarm) a (d) systém určil správně, že obě věty patří d stejného segmentu. (obrázek převzat z [2])

Podstatné je určit parametr $k$, který určuje statickou vzdálenost dvou určovaných vět. Tato hodnota se většinou stanovuje na poloviční velikost průměrné vzdálenosti dvou referenčních hranic.

# Kapitola 5

# Experimentování a výsledky

Veškeré testování a experimentování by se dalo rozdělit do dvou skupin, podle datasetu a cíle testů. Nejdříve přišlo na řadu testování s TDT5 datasetem, kdy bylo potřeba vyladit použitou metodu co nejlépe, aby se pak dala použít i na skutečné přednášky. Dále jsem u každého podkladu hodnotil úspěšnost dvou úkolů, hledání vazeb a segmentaci textu.

## 5.1 Testování s TDT5 datasetem

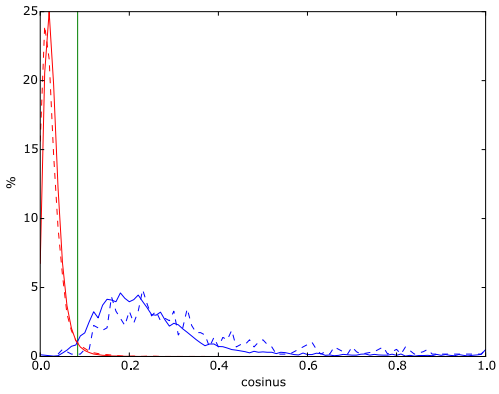
Při testování s tímto datasetem (více o něm v sekci 4.1) jsem nejdříve řešil problém hledání vazeb mezi jednotlivými články. Je jasné, že tento dataset obsahuje optimální data pro hledání tématu apod., protože text je psán stručně, výstižně a povětšinou obsahuje daleko více popisných a důležitých slov než mluvená řeč. Všechny výsledky jsem tedy musel brát s jakousi rezervou, že i když bude testování s TDT5 prokazovat dobré výsledky, u přednášek to nemusí být tak úplně stejné.

### 5.1.1 Hledání vazeb

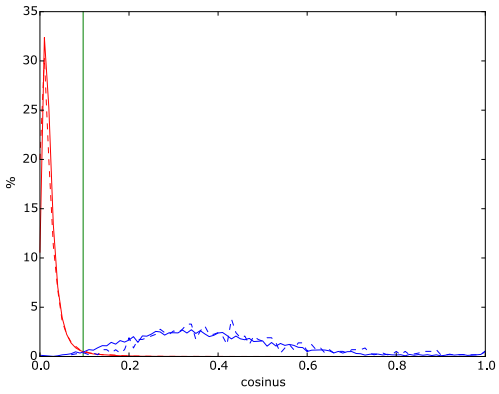
Pro testování jsem vybral dva podsety: 1) 260 článků z 10 témat s dělícím poměrem 1:1 a 2) 1300 článků z 50 témat s dělícím poměrem 1:3. Dělícím poměrem jsem články rozdělil na testovací a trénovací sety. U každého článku jsem si uložil ID jeho tématu a začal porovnávat každý s každým - začal jsem trénovacím setem. V rámci této fáze jsem měl k dispozici celý set, mohl jsem určovat vazby a kontext jednotlivých slov. Pomocí trénovacího setu jsem se naučil příslušné váhy (nebo témata - viz testované metody dále) slov a především určil práh, kdy byla nejmenší chyba: posunoval jsem jej od 0 do 1 s krokem 0.001 a v každém kroku zaznamenával počty *miss* a *fa*. Kde byla podle metriky nejlepší úspěšnost, ten jsem uložil. Aplikoval jsem jej poté pro testovací set následujícím způsobem: práh je hodnota mezi 0 a 1. Obdrženou hodnotu porovnávání dvou článků s ním porovnám a pokud je hodnota menší, klasifikuji články jako rozdílné. Pokud je hodnota větší jak práh, klasifikuji články jako podobné.

Po naučení informací jsem spustil porovnávání testovacího setu. Nyní jsem vycházel už pouze ze znalosti dvou porovnávaných článků a z naučených informací o slovech. Vzal jsem tedy oba články, zpracoval je, z jejich slov vytvořil dva vektory (když jsem narazil na neznámé slovo, které se neobjevilo v rámci učení, zahodil jsem jej) a porovnal. Výslednou hodnotu jsem si uložil do jedné ze dvou struktur podle výsledku porovnání a nakonec vykreslil dva grafy:

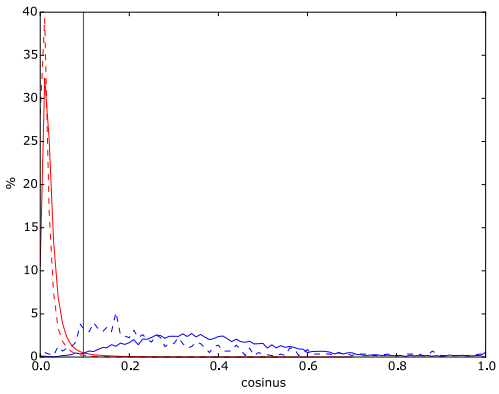
- Graf zobrazující výsledné hodnoty cosinů v závislosti na výsledku porovnání (viz grafy 5.1).
- DET (*detection error tradeoff*) křivku, která ukazuje závislosti dvou možných chyb, *missed detection* a *false alarm*. Více viz graf 5.2.



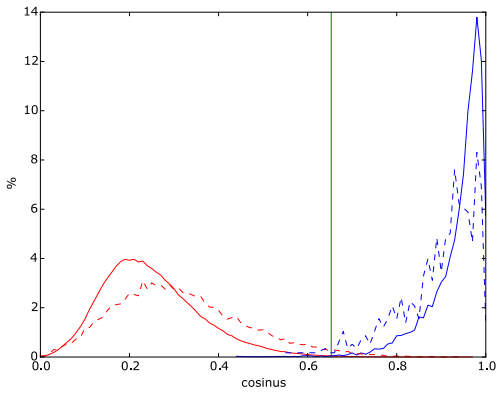

(a) TF-IDF #1




(b) TF-IDF #2




(c) TF-LLR




(d) TopicID

Obrázek 5.1: Na grafech lze vidět zaznačené výsledné hodnoty cosinů dle jednotlivých metod. Červená křivka představuje hodnoty porovnání článků různých témat a modrá hodnoty porovnání článků stejných témat. Plnou čarou jsou zaznačeny hodnoty získané trénováním a čárkovaně jsou hodnoty z testování. Jediná svislá čára je výsledný práh (je v místě, kde se střetávají červená s modrou (plné) - tam je také nejmenší chyba). Je vidět, že relativní normalizace velmi zlepší popisování článku vektorem. Za pozastavení stojí poslední graf, který vypadá podstatně jinak jak ostatní: hodnoty jsou lépe separované a hodnoty porovnání jsou daleko vyšší jak u ostatních metod. Přesné výsledky jsou k vidění v tabulce 5.1.1.

Porovnával jsem úspěšnosti čtyř různých přístupů k reprezentaci textu (popsaných podrobněji v sekci 3.1):

1. TF-IDF s logaritmickou normalizací složky TF (TF-IDF #1),
2. TF-IDF s normalizací pomocí celkového počtu slov ve článku (TF-IDF #2),

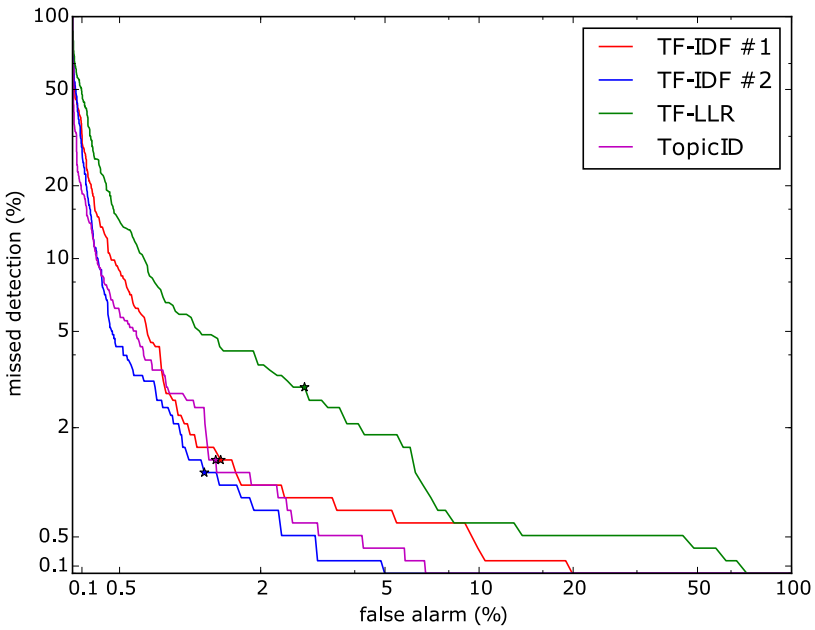
3. TF-LLR,

4. Zjednodušená verze přiřazování tématu ke každému slovu (zkráceně TopicID). V učící fázi jsem pro každé slovo podle toho, v jakém tématu se nejvíce objevovalo, přiřadil ID tohoto tématu.

Také jsem zkoušel dopad poměru testovacích - trénovacích dat a jejich celkové počty. Výsledné procentuální úspěšnosti počítané podle vzorce 4.1 lze vidět v tabulce 5.1.1.

| | | TF-IDF #1 | TF-IDF #2 | TF-LLR | TopicID |
|---|---|---|---|---|---|
| 1:1 | $C_{det}$ v % | 9.12 % | **2.86 %** | 8.14 % | 12.57 % |
| | práh | 0.086 | 0.113 | 0.089 | 0.815 |
| 1:3 | $C_{det}$ v % | 3.12 % | 2.66 % | 2.86 % | **2.34 %** |
| | práh | 0.084 | 0.097 | 0.097 | 0.653 |

Tabulka 5.1: V tabulce lze vidět výsledky pro jednotlivé metody. Všechny se při navýšení počtu trénovacích dat (změnění poměru mezi sety) a navýšení celkového počtu dat zlepšily.




Obrázek 5.2: DET křivka ukazující závislost *missed detection* a *false alarm* na posunujícím se prahu. Hodnoty jsou ukládány po 0.001 (posunující se práh). Čím blíže je křivka k osám, tím je její úspěšnost lepší. Dále má každá z křivek bod, který je označován EER - *equal error rate* (hvězdička), který označuje místo, kde si jsou obě hodnoty nejpodobnější (a zároveň nejnižší). Čím bude tento bod blíže k 0, tím lépe budou hodnoty porovnání článků stejných a jiných témat separované.

Z výsledků (tj. z tabulek a grafů) je vidět, že velký dopad na úspěch měl dostatečný počet trénovacích dat a větší rozmanitost témat. Zajímavé jsou výsledky metody TopicID, která přestože je velmi zjednodušená (nemělo smysl ji vyvíjet - potřebuje trénovací data), předvedla nejlepší zlepšení po zvětšení počtu trénovacích dat. Celkově ale vyšla nejlépe metoda TF-IDF s relativní normalizací, která zároveň ani nutně nepotřebuje trénovací data. Proto jsem ji vybral i pro další část úkolu.

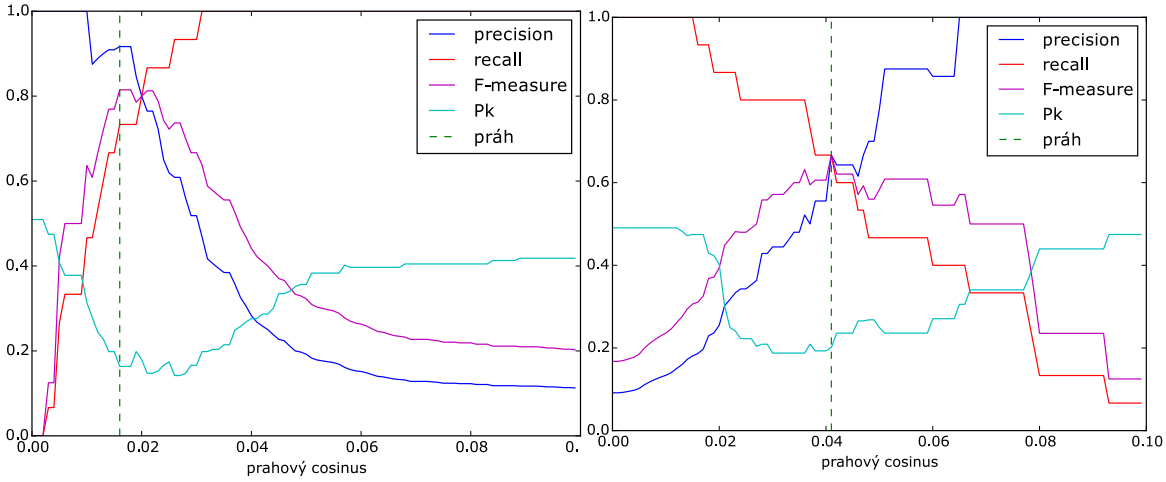
### 5.1.2 Segmentace

V rámci přípravy na testování segmentace jsem si připravil soubory se zhruba 6000 slovy (imitace hodinového záznamu řeči) - zprávy z TDT5 pospojované za sebe. V takto vytvořeném souboru jsem si nejdříve uložil definované hranice mezi jednotlivými články, a pak se je snažil najít.

Po nalezení všech potenciálních minim jsem jako v hledání vazeb bral práh a od 0 do 0.2 (nebylo potřeba výš) jej posunoval o 0.001. Pro určování hodnoty mám dvě možnosti (skutečná hodnota, nebo *depth score* - viz sekce 4.4). V každém případě pro každý posun prahu určuji počet zmeškaných a nadbytečných hranic, ze kterých poté vypočítám *F-measure*.

| | F-measure | $P_k$ |
|---|---|---|
| použití minim | 0.79 | 0.12 |
| použití depth score | 0.59 | 0.25 |

Tabulka 5.2: Byly porovnávány dvě metody posuzování minim. První bylo použití $n$ nejnižších minim, kde $n$ je optimální počet (metrika dávala nejlepší výsledky). Druhou možností bylo použít zmíněné *depth scores* (zase vzít $n$). Zde jsou k vidění číselné výsledky; pro F-measure platí vyšší je lepší, u $P_k$ je to naopak. Celý problém segmentace je ale velmi náročný na zhodnocení, proto pro lepší porovnání doporučuji obrázek 5.4.

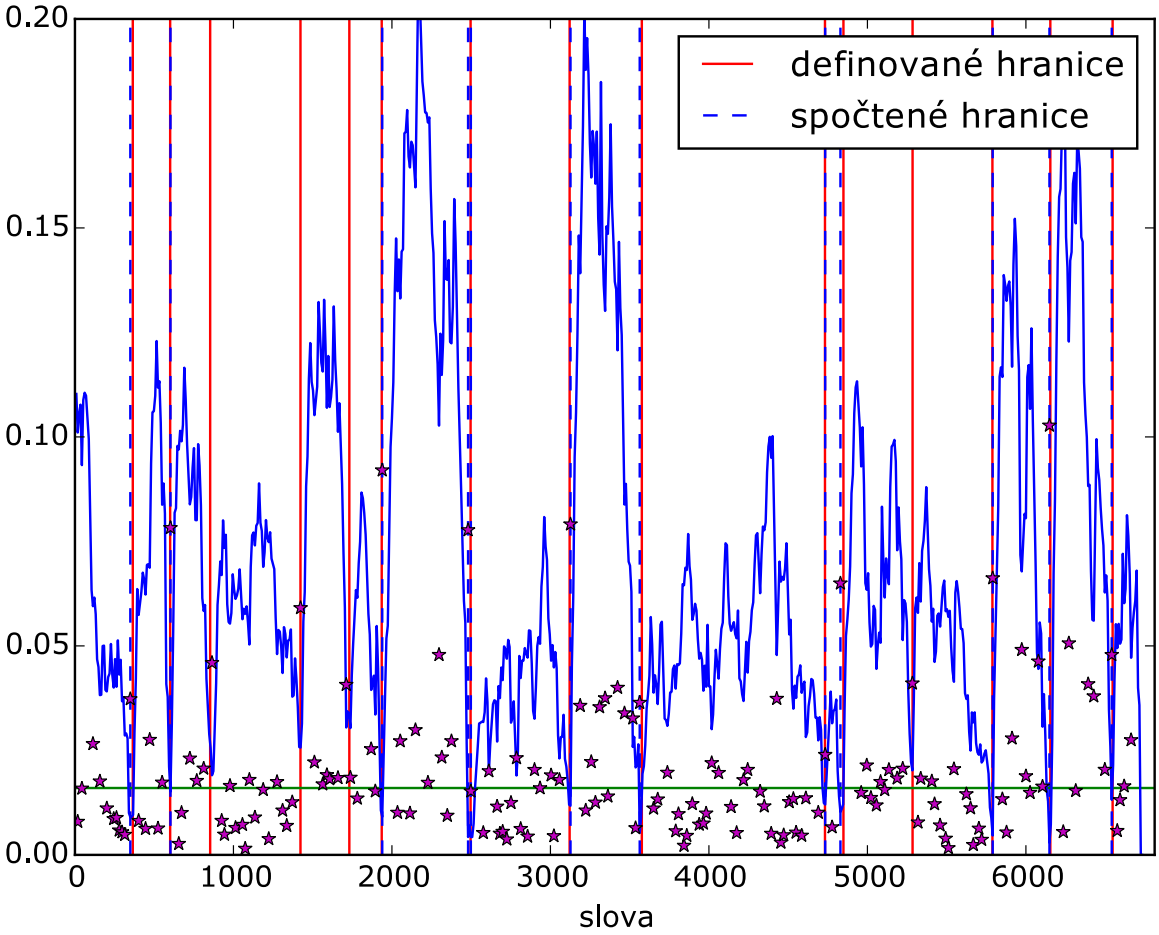



(a) použití pouze hodnot minim

(b) použití *depth score* pro minima

Obrázek 5.3: Na grafech jsou zaznamenány hodnoty jednotlivých metrik pro postupující práh. Na levém grafu se berou hodnoty **pod** prahem, vpravo **nad** prahem. Zaměřme se na levý graf. S rostoucí tolerancí je vidět narůstající hodnota *recall*, která značí nalezení referenčních hranic. *Precision* ale klesá, protože přibývají nadbytečné hranice. Pomocí toho je vykreslena *F-measure*, která má své maximum zhruba v místě minima $P_k$, obě metriky tedy relativně podobně určují přesnost. Z hlediska přesnosti je na tom podstatně lépe (a), hodnota *F-measure* je podstatně vyšší; *recall* také roste rychleji.

Z grafů a výsledků jsem vybral jako hlavní rozhodovací způsob pro segmentaci metriku *F-measure*. Dále také jako referenční hodnotu jsem vybral pravou hodnotu minima, vzhledem k lepší úspěšnosti. Navzdory dobré úspěšnosti se po skutečném experimentování s přednáškami objevilo několik nových problémů.




Obrázek 5.4: Graf hodnot cosinů v předělech. Všechna minima jsou potenciálními hranicemi, ale pouze ty pod svislou čarou skutečně akceptuji. Pro každé minimum je vykreslena hvězdička, ukazující *depth score* daného minima. Je vidět, že někde obě hodnoty silně korelují (křivka nízko, hvězdička vysoko), ale jinde je výpočet *depth score* narušen drobným pokleskem sousední hodnoty, a tak není hodnota přesná; jinde je tomu ale naopak. Je vidět také několik osamocených sloupců zřetelně oddělených nízkými minimy představující dílčí články.

## 5.2 Experimentování s přednáškami

K dispozici jsem měl transkripty spousty kurzů, ale bohužel pouze dva kurzy (Signály a systémy (ISS) a Zpracování řečových signálů (ZRE)) jsou aktuálně nasazeny na webu SuperLectures i s videem a slajdy, takže kvůli reálnému testování jsem zvolil pouze tyto dva. Metodu pro porovnání jsem využil TF-IDF # 2 (viz předchozí sekce) s výpočtem IDF mezi vygenerovanými segmenty a u segmentace jsem se řídil hodnotami minim.

Při segmentaci vzniklo ale několik problémů, které u testování nebyly:

**Nebylo jak určit práh** Zprvu jsem vycházel z toho, co mi vyšlo u testování, ale musel jsem jej podle výsledků operativně posunout, aby se přednášky rozdělovaly rovnoměrně.

**Vznikaly velmi krátké části** Podobnost v průběhu přednášky mnohdy hodně kolísala (bylo nestálé téma), takže vznikaly části např. minutu dlouhé. Rozhodl jsem se, že všechny hranice, které jsou vzdálené zhruba 2 minuty zprůměruji a spojím do jedné. Taktéž všechny výsledné části, které mají pod 3 minuty vyřazuji.

**Vznikaly příliš dlouhé části (45+ min)** Někdy se stalo, že graf podobnosti sice vytvářel sloupce oddělené dvěma minimy, ale nikdy minima neklesla pod práh - algoritmus tedy nic nedělil. Zavedení *depth scores* zase neuvažovalo minima tak dobře.

### 5.2.1 Průběh experimentů

Po zpracování přednášek jsem vytvořil dva výsledné seznamy vazeb. V každém jsem pro každou část přednášky hledal tři nejpodobnější části z jiných přednášek. V prvním jsem hledal mezi všemi ostatními částmi - mezikurzová vazba byla velmi slabá (jen v 1/10 případů se pro ISS část našla ZRE část). Když jsem nastavil, aby se vygenerovaly pouze vazby na části z druhého předmětu, výsledné dvojice měly ve většině případů velmi malé skóre. Je možné, že některé části nemají vůbec žádný protějšek v druhém kurzu.

Předal jsem seznamy (výsledných dvojic bylo kolem 1000) skupině Speech@FIT, která mi poskytla testovací stránku s nastříhanými částmi přednášek podle výsledků. Na stránce byla dvě videa (části z přednášek). U každého byla jedna otázka na správnost segmentace a jedna společná na správnost porovnání. Z porovnání jsem získával IP adresu respondenta (pro zjištění počtu unikátních respondentů), délky jednotlivých částí, kurzy, odkud části byly, cosinus a *ppmcc* jejich porovnání a odpovědi.

Dále byla stránka vypuštěna mezi studenty a některé pracovníky na FIT. Bohužel porovnávání je poněkud časově náročnější (člověk může dostat dvě části 25 min dlouhé - a to mohou být ještě ty kratší) a zvláště pokud člověk neměl jeden z kurzů (v mém případě ZRE), porovnávání se stává náročnějším co do energie, tak i času. Nelze se tedy příliš divit, že se mezi studenty nenašlo mnoho dobrovolníků.

### 5.2.2 Výsledky

Průzkumu se zúčastnilo celkem 30 lidí, od kterých jsem dostal 104 odpovědí na otázky (pro segmentaci jsem dostal dvojnásobek - od každého dvě). Následují grafy a tabulky hodnot a rozbor výsledků pro oba úkoly.

**Segmentace**

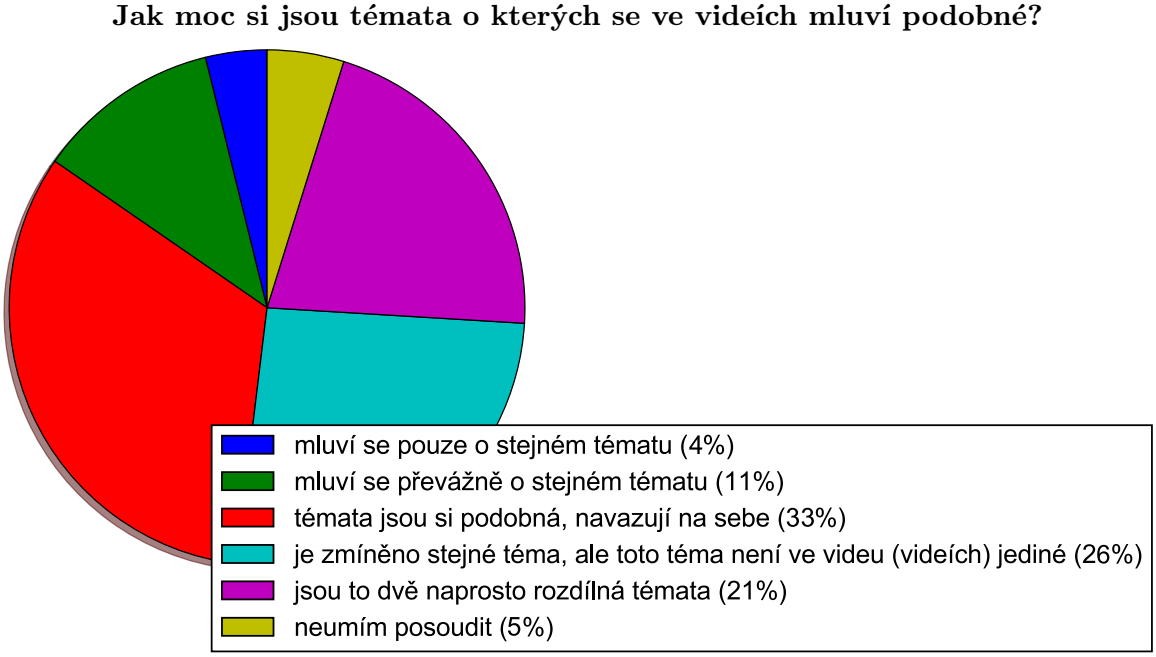

Odpovědi na tuto otázku byly relativně jednoznačné. Relativně proto, že každý může brát „téma“ v přednášce jinak. Podle výsledků sice vychází úspěšnost 80 % (pokud bychom brali první dvě odpovědi jako ty správné), ale úspěšnost samotné segmentace se dá lépe určit pouze společně s výsledky druhé otázky.

Podle tabulky níže je také vidět, že respondenti označovali třetí možnost především u částí velké délky. Do budoucna by bylo optimální buď algoritmus vylepšit, aby lépe hranice rozpoznal (ale co je to vlastně ta hranice?), nebo jej pustit rekurzivně na části, které mají délku větší než zvolené $x$. Případně je jednoduše rozdělit v nejpravděpodobnějším místě. Dlouhé části vytvářejí i jiné problémy - viz vyhodnocování porovnávání.

| | 1. odpověď | 2. odpověď | 3. odpověď |
|---|---|---|---|
| průměrná délka části | 17m 30s | 27m 19s | 43m 47s |

Tabulka 5.3: Tato tabulka ukazuje průměrné délky částí, které se vázaly k jednotlivým odpovědím. Je vidět, že čím delší část je, tím je skutečně větší šance, že bude obsahovat více témat.

**Porovnávání**




Vyhodnocování druhého úkolu bylo podstatně složitější. Už i sestavování otázek, aby vyhovovaly tomu, co si o částech člověk může myslet mělo svá úskalí. Dle prvního pohledu na graf může působit, že je velmi nízká úspěšnost (stejná témata to našlo jen v 15 %). Málokdy se ale v přednáškách opakují ty samé věci (výjimkou jsou např. opakování a příklady = ~15 % kurzu). Na druhou stranu se ale v nich objevuje spousta podobných věcí, které spolu nějak souvisí (33 %). Dále téměř čtvrtina (26 %) odpovědí tvrdila, že tam sice podobné téma je, ale není jediné. Z této čtvrtiny až 85 % respondentů odpovědělo na jednu z předchozích otázek „video obsahuje více hlavních témat“. Předpokládám, že tento problém by se dal vyřešit vylepšením segmentace. K předposlední odpovědi a předpokládaným důvodům k chybám se vyjádřím níže. Poslední odpověď netřeba rozebírat.

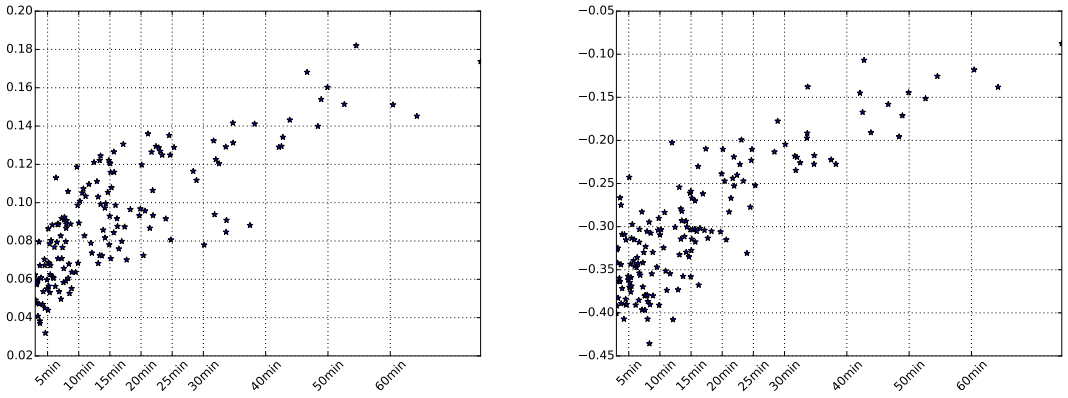
| | Odpovědi | | | | | |
|---|---|---|---|---|---|---|
| | 1. | 2. | 3. | 4. | 5. | 6. |
| průměrná *cossim* | 0.17 | 0.20 | 0.17 | 0.19 | 0.15 | 0.23 |
| průměrná *ppmcc* | -0.24 | -0.22 | -0.23 | -0.16 | -0.19 | -0,02 |
| % porovnávání stejných kurzů | 100 % | 92 % | 71 % | 85 % | 55 % | 80 % |

Tabulka 5.4: V tabulce lze vidět průměrné informace o částech a porovnáních, které vedly k jednotlivým odpovědím. V prvním a druhém řádku jsou průměrné hodnoty *cosine similarity* a *ppmcc* pro porovnání. V třetím řádku je údaj, který udává kolik procent ze všech porovnání bylo mezi stejnými kurzy.

V prvním řádku tabulky hodnoty cosinů souhlasí s předpokladem (čím vyšší cosinus, tím větší podobnost), i když ne úplně přesně (cosiny 1. a 5. odpovědi jsou blízko). Relativně vyšší hodnoty cosinů pro 5. odpověď zdůvodňuji tím, že se občas ve dvojici najde jedna

část, která je velmi dlouhá (45+ min). Tyto dlouhé části mají velkou tendenci produkovat velké skóre, přestože si nejsou nijak zvlášť podobné - obsahují hodně slov, mnoho z nich podobných jako u kratší části, ale také mnoho jiných, viz obrázek 5.5.

Hodnoty *ppmcc* v druhém řádku jsou podobně vyrovnané jako hodnoty v prvním řádku, výsledky této metody ale vůbec nekorespondují s předpokladem, že správné odpovědi budou mít vyšší hodnoty.




(a) použití *cosine similarity*

(b) použití *ppmcc*

Obrázek 5.5: Grafy ukazují výše zmíněnou chybu: části větších délek (osa x) dostávají v porovnáváních průměrně vyšší hodnoty (osa y). Je vidět, že skutečně hodnoty nejsou v jedné pomyslné vodorovné linii. V obou případech jsou hodnoty v některém místě vyrovnané, ale v nějakém bodě se začnou velmi odlišovat. Průměrná délka všech vygenerovaných částí je kolem 15 min a průměrná délka přiřazených podobných částí je kolem 43 min. Tato chyba by se dala odstranit zlepšením segmentace a případně lepší normalizací délky části.

V třetím řádku tabulky jsem vypsal kolik procent z porovnávaných dvojic obsahovalo obě části z jednoho kurzu. Potvrdila se domněnka z generování seznamů dvojic (sekce 5.2.1), a to, že vazby mezi oběma kurzy byly velmi slabé, přestože tyto dva kurzy spolu úzce souvisí. Je vidět, že ty nejlepší odpovědi (1. a 2.) se nacházejí převážně u dvojic ze stejného kurzu. Na druhou stranu nejhorší odpovědi (5.) byly z poloviny mezi dvěma kurzy. Vysvětluji to dvěma způsoby: 1) skóre mnoha mezikurzových dvojic bylo velmi malá, je možné, že mnoho z nich nemělo protějšek v druhém kurzu a 2) respondentům se hůře porovnala podobnost mezi různými kurzy (je potřeba hledat významové vazby a pokud tomu ne každý rozumí, je to těžké).

Když jsem pro každý z dvou typů porovnání (stejný kurz X jiný kurz) vygeneroval nové statistiky, pro dvojice ze stejného kurzu bylo 5. odpovědí pouze 15 %, kdežto u dvojic z rozdílného kurzu až 35 % (také proto, že algoritmus vygeneroval tři „nejlepší“ jiné části, nehledě na to, že všechny byly špatné - lepší nebyly).

### 5.2.3 Zhodnocení

Z vyhodnocení segmentace plyne navzdory mému prvnotnímu odhadu z grafů dobrá úspěšnost. V budoucnu by se mohlo zajistit rozdělení delších částí na menší a pokusit se tak separovat jednotlivá témata. Ale ze zhruba 75 % segmentace funguje správně a to beru jako dobrý výsledek.

S porovnáváním je to složitější. Jedním z cílů bylo provázat přednášky v jednom kurzu, což si myslím, že bylo splněno dobře. U porovnávání jiných kurzů už byla chyba větší, nicméně i tady byla 5. odpověď pořád v menšině. Podstatné úkony, které by porovnávání velmi zlepšily a separovaly dobré hodnoty od špatných jsou:

**Zlepšení segmentace** Tím se logicky i zlepší provázání. Odstraní se dlouhé části, které na sebe neprávem vážou mnoho jiných (jen protože jsou dlouhé a algoritmus pro delší části určuje větší skóre).

**Určení prahu** Zde je to složitější, určitě by to chtělo mnohem více výsledků, než zatím mám, aby se dal jednoznačně určit práh, pomocí kterého by se určovalo, jestli si jsou skutečně podobné. Každopádně je toto do budoucna nutnost, jak bylo vidět z výsledků (je lepší nabídnout podobnou část stejného kurzu než rozdílnou z jiného).

**Normalizace délky částí** Souvisí s prvním bodem. Podle seznamu se části mohou vázat na jiné dlouhé, přestože spolu nesouvisí. Je potřeba tento problém nějak vyřešit, optimálně najít vhodnou normalizaci délek částí.

# Kapitola 6

# Závěr

V této práci jsem uvedl problematiku *topic detection and tracking* a různé způsoby získávání tématu z textu. Porovnal jsem několik metod pro posuzování podobnosti mezi dvěma texty a vybral tu nejlepší, kterou jsem aplikoval dále. Také jsem předložil zpracování problému segmentace delšího textu na dílčí samostatné články. Toto vše jsem důkladně otestoval na testovacích datech a nasadil na přepisy z přednášek zaznamenaných na FIT.

Při experimentování s porovnáváním na datasetu TDT5 jsem dosáhl úspěšnosti kolem 90 %. S dostatkem dat pro naučení systém funguje skvěle. Poté byl systém nasazen na přednášky a otestován studenty a pracovníky na FIT. I přes fakt, že systém byl nasazen na neznámé vzorky, z 80 % zvládl správně detekovat téma v přednášce. U porovnávání vycházela úspěšnost nižší, ale z velké části dokázal systém správně určit dvojice příbuzných částí přednášek.

I přes dobré výsledky systém určitě není dokonalý. Vzhledem k tomu, že použití náročného subjektivního testování bylo nutností, nedal se algoritmus příliš zlepšovat podle uživatelských reakcí. Několik poznatků do budoucna bych ale přesto vyzdvihl. Pro zlepšení segmentace by pomohla analýza více transkriptů pro zjištění optimálního prahu pro dělení. Vzhledem k tomu, že stálost témat je v každé přednášce jiné, dalo by se uvažovat o víceúrovňovém dělení každé přednášky. Věřím, že zdokonalení segmentace by velmi pomohlo při zlepšování úspěšnosti porovnávání. U vyhledávání dvojic by nejvíce pomohlo daleko rozsáhlejší testování (několikanásobně více vzorků než jsem sehnal) a zjištění závislosti skutečné podobnosti na skóre porovnání - jinými slovy získání přesného prahu pro klasifikaci podobných a rozdílných částí.

# Literatura


[1] Allan, J.; Carbonell, J. G.; Doddington, G.; aj.: Topic detection and tracking pilot study final report. 1998.

[2] Beeferman, D.; Berger, A.; Lafferty, J.: Statistical models for text segmentation. *Machine learning*, ročník 34, č. 1-3, 1999: s. 177–210.

[3] Choi, F. Y.: Advances in domain independent linear text segmentation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, Association for Computational Linguistics, 2000, s. 26–33.

[4] Chung, M. K.: Correlation coefficient. *Encyclopedia of measurement and statistics*, ročník 1, 2007.

[5] Fiscus, J. G.; Doddington, G. R.: Topic detection and tracking evaluation overview. In *Topic detection and tracking*, Springer, 2002, s. 17–31.

[6] Hazen, T. J.: MCE training techniques for topic identification of spoken audio documents. *Audio, Speech, and Language Processing, IEEE Transactions on*, ročník 19, č. 8, 2011: s. 2451–2460.

[7] Lliu, X.; Ma, F.; Lin, H.: Topic Detection with Hypergraph Partition Algorithm. *Journal of Software*, ročník 6, č. 12, 2011: s. 2407–2415.

[8] Pérez, R. A.; Pagola, J. E. M.: Improving Text Segmentation with Clustering Cohesion.

[9] Pérez, R. A.; Pagola, J. E. M.: An incremental text segmentation by clustering cohesion. *HaCDAIS 2010*, 2010: str. 65.

[10] Riedl, M.; Biemann, C.: Text segmentation with topic models. *Journal for Language Technology and Computational Linguistics*, ročník 27, č. 1, 2012: s. 47–69.

[11] Suárez, A. P.; Trinidad, J. F. M.; Ochoa, J. A. C.; aj.: A new incremental algorithm for overlapped clustering. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, Springer, 2009, s. 497–504.

[12] Wartena, C.; Brussee, R.: Topic detection by clustering keywords. In *Database and Expert Systems Application, 2008. DEXA'08. 19th International Workshop on*, IEEE, 2008, s. 54–58.

# Příloha A

# Obsah CD

Přiložené CD obsahuje:

- `src/`: obsahuje zdrojové kódy v jazyce Python3.4
- `doc/`: obsahuje zdrojové kódy v LaTeXu pro tuto práci
- `results/`: obsahuje výsledky a grafy:
  - `lectures_segmentation/`: obsahuje grafy hodnot cosinů jednotlivých přednášek, podle kterých se segmentovaly
  - `links/`: obsahuje seznamy vygenerovaných spojení mezi částmi přednášek
  - `measures/`: obsahuje grafy výsledků testování podle metrik
  - `responses/`: obsahuje grafy odpovědí od lidí
  - `segmentation/`: obsahuje grafy segmentace testovacích dat
- `transcripts/`: obsahuje přepisy přednášek a další vstupy
- `poster.pdf`: demonstrační plakát
- `video.mp4`: demonstrační video
- `README.txt`: informační soubor obsahující dodatečný popis skriptů, popis instalace a další detaily