**Mendelova univerzita v Brně**
**Provozně ekonomická fakulta**

# Dávkový automatický korektor pravopisu slov rozsáhlých textových souborů

## Diplomová práce

**Vedoucí práce:**
**doc. Ing. Jan Žižka, CSc.**

**Bc. Martin Péchal**

**Brno 2014**

**Poděkování**

Zde je možné psát text poděkování. Pokud bude poděkování vynecháno a používáte dvoustranný dokument, pouze smažte text, ale nechte prázdnou stránku v dokumentu.

**Čestné prohlášení**

Prohlašuji, že jsem tuto práci: **Dávkový automatický korektor pravopisu rozsáhlých textových souborů** vypracoval/a samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*

Jsem si vědom/a, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 24. května 2014 _____

***Abstract***

*Jandová, M., Rybička, J. The document template for bachelor/diploma thesis. Bachelor thesis. Brno: Mendel University, 2009.*
*Example of document template for bachelor/diploma thesis is presented here. An abstract is in (British) English.*

***Keywords***

*Thesis, template, document, thesis example. Here are key words in (British) English.*

**Abstrakt**

Jandová, M., Rybička, J. Šablona pro závěrečné práce. Bakalářská práce. Brno: Mendelova univerzita v Brně, 2009.
V textu je popsána šablona pro závěrečné práce a je uvedena v příkladech.

**Klíčová slova**

Závěrečná práce, šablona, dokument, příklad závěrečné práce.

# Obsah

# Seznam obrázků

# Seznam tabulek

# 1 Úvod a cíl práce

## 1.1 Úvod

Dnešní doba s sebou nese značný nárůst objemu dat ve světe. Jednou z hlavních tažných sil tohoto nárůstu je internet, který dnes pokrývá velkou část populace. Navíc se dále zlepšuje rychlost, spolehlivost a další aspekty, které vedou k jeho dalšímu rozšíření a větší míře užívání. Na rozšířenost internetu navazují i další technologie, jako jsou například chytré telefony apod., pomocí kterých můžeme produkovat další a další data. Je tedy zcela zřejmé, že se nárůst objemu dat ve světe jen tak nezastaví a zdá se, že se bude rychlost nárůstu spíše ještě zvyšovat.

Objem dat už dosáhl takových rozměrů, že je často nelze zpracovávat ručně, a proto se také mění role člověka v procesu zpracovávání dat. Člověk spíše vytváří nástroje, které použije na zpracování dat a poté kontroluje výsledky. Objemy dat, které je potřeba zpracovávat, navíc dnes v některých případech dosahují dokonce takových velikostí, že je problém je zpracovávat i automatizovaně. V poslední době se v souvislosti s touto problematikou objevuje pojem *Big data*, kterým je tato problematika označována. Tento pojem ale zachycuje širší pohled na problematiku dat a nezahrnuje pouze velké množství dat, ale i další faktory, jako například rychlost vzniku a různorodost dat (Russom, 2011, s. 6).

Jeden z typů dat, který tvoří velkou část objemu dat ve světě, jsou textová data. Textová data mají spoustu úskalí, která komplikují jejich zpracování, jako je například mnoho různých jazyků, problémy s kódováním apod. Navíc informace v textové podobě nemusí být úplně přesné nebo spíše nejasné. Pokud je například použito spojení „koupil jsem to“, tak není jasné, co bylo myšleno pod slovem „to“. Pokud se na to tedy podíváme z pohledu definice pojmu *Big data* zmíněného výše, tak je zřejmé, že textová data souvisí hlavně s faktorem různorodosti, která je zde zjevně velká.

S textovými daty souvisí problematika chyb v textu. Chyby mohou být způsobeny navíc různými důvody, jako je například překlep, přeřeknutí při diktování či neznalost správného pravopisu slova. Snaha omezit tyto chyby vedla k tomu, že spousta textových editorů a programů obecně, přes které lze zadávat text, obsahuje součást na vyhledávání chyb, na které upozorňuje uživatele. Pokročilejší implementace této kontroly se snaží navíc rovnou chybu opravit nebo alespoň nabídnout co nejlepší možnou opravu ke špatně napsanému slovu.

Jak vyplývá z textu výše, tak převážná část aplikací, které danou problematiku řeší, pracuje interaktivně přímo při práci uživatele. Situace se tedy komplikuje v případě, že potřebujeme provést korekci u textu bez pomoci uživatele, který lépe rozumí kontextu celého textu a dokáže lépe zvolit vhodné slovo. Pokud by tedy bylo potřeba zkontrolovat a opravit větší množství textu, tak by nemusela být ruční korekce vhodným přístupem (skutečnost bude záviset na množství a požadované přesnosti). Příkladem je situace, kdy budeme chtít pomocí nějaké metody dolování znalosti z dat získávat nějaké informace z většího množství textových dat. V takových případech je velikost zdrojových dat již tak rozsáhlá, že korekci

v podstatě ani ručně nelze provést. Vzhledem k tomu, jak roste rychlost nárůstu objemu dat, lze předpokládat, že se bude v budoucnu provádět dolování znalostí z dat stále častěji a postupně nad většími objemy. Proto by bylo vhodné mít k dispozici nějaký nástroj, který by dokázal korekci provést automaticky, i když třeba nebude výstup tak kvalitní, jako kdyby prováděl korekci uživatel.

## 1.2 Cíl práce

Cílem práce je navržení a následná implementace softwarového nástroje, který bude provádět korekci textu automaticky bez nutného významnějšího zásahu uživatele. Textové vstupy jsou předpokládány v plain text formátu. Primární účel, pro který by měla být aplikace optimalizována, je korekce textu před procesem dolování znalostí. Z toho vyplývá, že aplikace bude muset být schopna pracovat s velkými objemy textových dat (v řádech stovek MB).

Dílčí částí práce bude příprava slovníků a jejich součástí, pomocí kterých bude prováděna korekce nalezených chyb. Slovníky by se přitom měly dát měnit či aktualizovat nezávisle na aplikaci, aby byla ponechána možnost jednoduchého rozšíření v budoucnu.

Na závěr bude provedeno zhodnocení dosažených výsledků z hlediska přesnosti korekce a následně i z hlediska paměťové a časové náročnosti. Hodnocení bude provedeno jak z hlediska testovacích dat, tak z hlediska reálných dat.

# 2 Literární přehled

Ačkoliv by se to nemuselo na první pohled zdát, tak se chyby v textu vyskytují v poměrně velké míře, jak uvádí například Wang et al. (2003, s. 754), při zadávání dotazů do vyhledávačů v anglickém jazyce dosahuje dokonce míra překlepů 26%. Takto velké procento nicméně může být způsobeno tím, že lidé již počítají s tím, že je opraví funkce „měli jste na mysli" a neopravují tedy chyby i v případě, že si jich všimnou. U delšího textu psaného běžně na počítači bude chybovost značně nižší oproti příkladu uvedenému výše. Kukich (1992) uvádí několik studií[1], které se zabývaly podrobněji frekvencí výskytu chyb v delších textech a dané studie uvádí chybovost v rozsahu 1-3%. V praxi tedy bude značně záležet na původu textu, který bude kontrolován.

Role programů na korekci textů je dnes zřejmá také v případě mobilních zařízení, ta se dnes totiž rozšířila natolik, že je jasné, že musely být implementovány nástroje pro korekci chyb, vzhledem k tomu, jak jednoduché je vytvořit na daných zařízeních chybu, díky jejich malým velikostem (klávesnic). Nabízí se tedy otázka, zda nástroje používané k odstranění či minimalizaci chyb nemohou vést i k určité deformaci, nicméně tato práce se touto částí problematiky nezabývá.

Je nutné zmínit, že problematiku korekce textu komplikuje celá řada skutečností. Hlavní z nich je skutečnost, že ve světě existuje nepřeberné množství jazyků, přičemž každý má nějaká specifika. Tato specifika komplikují implementaci korektoru. Jako konkrétní příklad pro anglický jazyk lze zmínit skutečnost, že existuje spousta slov s velmi podobnou výslovnosti, ale s jiným významem. Podobný problém je například u arabštiny, která obsahuje pro změnu znaky, které jsou si velmi podobné.

## 2.1 Typy chyb

Chyby ve slovech lze rozdělit na dva typy[2], prvním typem jsou situace, kdy díky překlepu vznikne slovo, které vůbec neexistuje (Kukich, 1992). Příkladem takové chyby je situace, kdy je chybně napsáno „autonobil" a ve skutečnosti mělo být napsáno „automobil". Ve slově bylo mylně prohozeno písmeno „n" za písmeno „m", přičemž tento konkrétní překlep je navíc poměrně pravděpodobný vzhledem k tomu, že obě písmena znějí stejně a jsou na klávesnici vedle sebe.

Druhým typem chyby je situace, kdy je použito jiné slovo, které v daném jazyce normálně existuje. Tato situace nastává například při přeslechnutí diktovaného slova, jelikož se některá slova vyslovují podobně, ale jinak se píší a mají jiný význam. Situace s přeslechnutím je problémem zejména v anglickém jazyce. V českém jazyce dojde pravděpodobněji k tomuto typu chyby stejným způsobem jako v případě prvního typu a to překlepem.

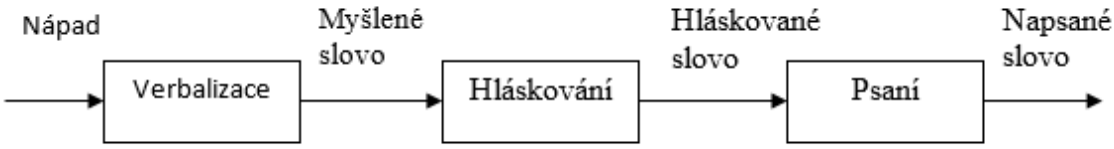
[1] Studie byly prováděny s textem v anglickém jazyce

[2] Typy chyb jsou v cizí literatuře označovány jako non-word (neexistující slova) a real-word errors

Kukich (1992) zmiňuje, že testy, které analyzují podíl druhého zmiňovaného typu chyb (slova co jsou chybně vzhledem ke kontextu, ale jsou ve slovníku) na celkovém počtu, je okolo 30-40%. Skutečný podíl se ale bude pravděpodobně lišit dle jazyka textu. V českém jazyce by mohl být podíl menší, protože čeština neobsahuje tolik podobně znějících slov.

## 2.2 Vznik chyb

Předtím, než bude rozebrána samotná detekce a korekce chyb, je dobré mít přehled o tom, jak vlastně typy chyb popsané v předchozí kapitole vznikají. Tuto znalost totiž lze později využít v návrhu procesu korekce chyb. Modely, které korekci řeší, totiž mohou být díky těmto znalostem přizpůsobeny konkrétní aplikaci (původu textových dat). Danou problematiku dobře ilustruje obrázek z (Deorowicz a Ciura, 2005, str. 276):




Obr. 1 Proces tvorby slova

Na daném obrázku lze dobře ilustrovat, kdy nejčastěji vznikají jednotlivé typy chyb. Typ chyb, kdy je špatné slovo normálně ve slovníku, vzniká nejčastěji v subprocesu verbalizace či hláskování. U verbalizace tento problém často pramení z neznalosti jazyka a nevědomosti o tom, jak má slovo vypadat, když je použito v jiné části řeči či kontextu. Stejný typ chyby může vzniknout při hláskování slova, pokud má slovo podobnou výslovnost jako jiné, či se jedná o jinou podobně znějící variantu stejného slova. Příkladem může být slovo „mně“, kdy v závislosti na konkrétní situaci (pádu) může být správné hláskování „mě“ a v jiné „mně“. Chyby prvního typu, kdy slovo neexistuje ve slovníku, nastávají zejména v poslední části procesu, kdy dojde například k překlepnutí při psaní na klávesnici.

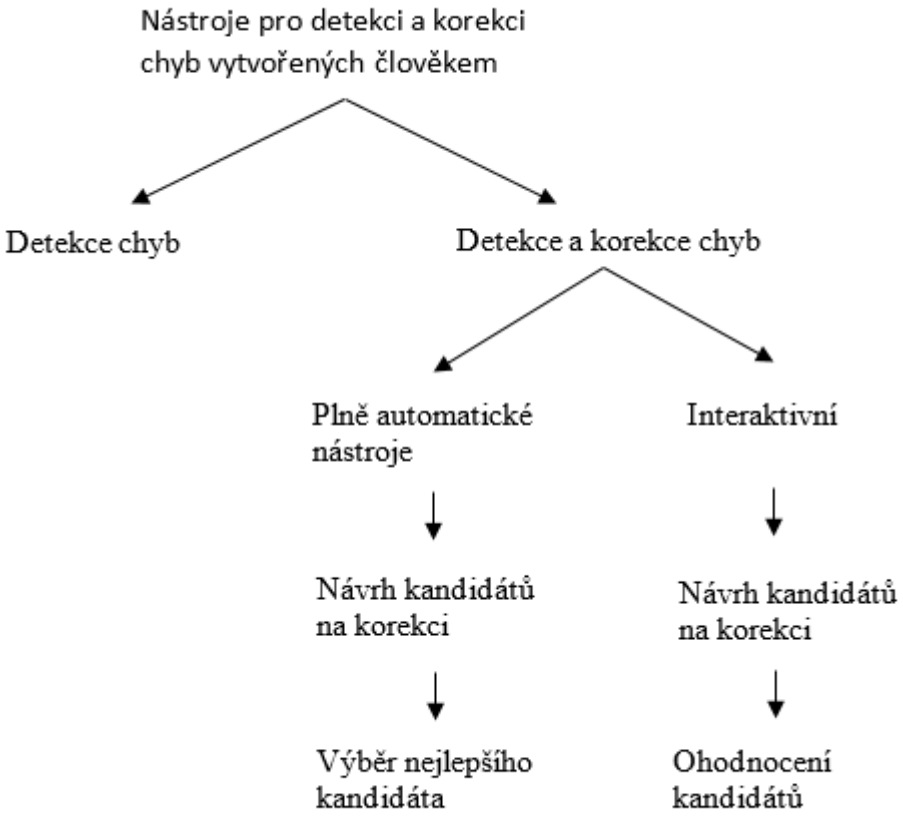
Kukich (1992) uvádí, že lze chyby v podstatě rozdělit na dvě třídy, kognitivní chyby a typografické. Pokud je daná definice vztáhnuta k obrázku výše, tak lze říci, že kognitivní chyby nastávají v prvních dvou částech procesu a typografické v poslední části (při psaní na klávesnici).

Do procesu tvorby textu mohou ještě vstupovat další proměnné, jako je například situace, kdy text vymýšlí někdo jiný a diktuje jej osobě, co jej zapisuje. Nebo může být celý proces zcela jiný. Příkladem takové situace je převod obrazu do tex-

tu pomocí OCR[3] nebo převod mluveného slova do textu. V těchto případech chyby vznikají často kvůli nedokonalosti daných technologií (Deorowicz a Ciura, 2005).

## 2.3 Typy aplikací

Předtím, než budou podrobněji rozebrány jednotlivé části procesu korekce chyb, je nutné zmínit, jaké typy korektorů textu existují. Implementace některých subprocesů korekce je totiž závislá na tom, jaký typ aplikace je vyvíjen a k čemu bude aplikace sloužit. Základní dva procesy, které definoval Kukich (1992) jsou detekce a korekce chyb. Některé softwarové nástroje korekci neřeší a nechávají ji pouze na uživateli. Situaci na tomto poli dále dobře ilustruje diagram níže (Verberne, 2002).




Obr. 2 Typy přístupů ke korekci textu

Z diagramu je zřejmé, že spousta aplikací implementuje pouze detekci chyb. Většinou se jedná o aplikace, které se touto problematikou přímo nezabývají, ale jsou v nich tvořena nějaká textová data (např. nástroje na programování), proto je im-

[3] Optical character recognition

plementována alespoň základní kontrola, zda jsou slova korektně napsaná. Tento typ lze považovat za interaktivní korekci až v momentě, kdy budou nabízeni kandidáti pro opravu. Nejsofistikovanější přístup vyžaduje automatická korekce, kde na rozdíl od interaktivní korekce musí být způsob hodnocení jednotlivých kandidátů podstatně přesnější, protože je automaticky volen nejlepší kandidát. U tohoto typu se tedy nabízí využití okolních slov, kterými je chyba obklopena (kontext). Daná problematika bude podrobněji rozebírána později.

Na typy nástrojů se lze podívat ještě z pohledu typů chyb. Většina nástrojů totiž řeší pouze chyby u slov, které nenajde ve slovníku. Důvodem je skutečnost, že pro detekci chyb kdy je použito jiné slovo, ale je ve slovníku, je nutné použít velmi sofistikovaný přístup, který má hodně společného se samotnou automatickou korekcí textu.

## 2.4 Detekce chyb

Jak již bylo naznačeno v předchozí části, tak korekci textu lze rozdělit na více částí. Nyní bude rozebírána první část a to detekce chyb. Detekce chyb spočívá v tom, že se postupně řetězce (slova) na vstupu vyhledávají ve slovníku (Kukich, 1992). Aby bylo možné identifikovat slova, která jsou chybně, tak je nutné využít slovník slov daného jazyka. Tento slovník by měl být pokud možno co největší, aby nebyla označována slova, která jsou dobře jako špatná. To by totiž mohlo vést naopak k tomu, že by byl počet chyb zvýšen. Daná slova by totiž byla nahrazena za jiná, která sice existují ve slovníku, nicméně nehodí se do aktuálního kontextu. Značnou nevýhodou tohoto přístupu je skutečnost, že je nutné udržovat slovník pokud možno co nejvíce aktuální a průběžně jej rozšiřovat o nově vznikající slova (Whitelaw et al., 2009).

I přesto, že se jedná o jednoduší část problematiky, zde mohou nastat problémy. Jedním z nich je právě velikost slovníku, kdy v případě velmi rozsáhlého slovníku může být výpočetně složitější vyhledávat požadovaná slova, nicméně díky výkonu současného HW a možnosti přizpůsobení struktury pro ukládání slovníku není tato část výraznějším problémem.

Ještě než budou rozebrány další potenciální problémy prvního přístupu, je nutné zmínit druhý možný přístup, který pracuje na odlišném principu s využitím n-gramů. Za n-gramy v tomto přístupu[4] považujeme sekvenci písmen o různých délkách. Unigram je jednopísmenná sekvence, digram je sekvence dvou písmen a trigram třech. V různých jazycích se jednotlivé sekvence vyskytují s různou frekvencí, což lze využít při analýze chyb. Pokud se ve slově vyskytuje nějaký n-gram, který má malou frekvenci výskytu, tak dané slovo může být chyba. Výhodou tohoto řešení je, že není potřeba slovník, nicméně příprava statistických informací o n-gramech je naopak relativně náročná. V praxi se tedy pro korekci textu používá spíše první zmiňovaný způsob s vyhledáváním ve slovníku (Kukich, 1992).

[4] Označení n-gramy se používá i v souvislostí s výskytem slovních spojení (např. dvě slova – digram atd.)

Horší situace ale nastává v případě, že text na vstupu je ve více jazycích. Pokud by byl použit pouze jeden slovník, tak budou daná slova z jiného jazyka označena jako špatná. Daný problém by šel teoreticky řešit použitím více slovníků, nicméně to přináší spoustu úskalí. Jedním z nich je například situace, kdy je slovo napsáno špatně, ale shodou okolností se nachází ve slovníku jiného jazyka, takže je označeno jako dobře napsané. Proto by bylo nutné implementovat funkčnost, která na základě okolí aktuálního slova rozpozná jazyk a použije podle toho daný slovník. V praxi ale není funkčnost rozpoznávání jazyka implementována, protože tento proces je poměrně komplikovaný. Existuje několik metod, které lze použít pro rozpoznávání jazyka, přičemž dané metody jsou postaveny na pravděpodobnostech výskytu n-gramů, markových modelech a dalších přístupech (Řehůřek, Kolkus, 2009). Funkčnost rozpoznávání jazyků je často suplována jinými přístupy, jako je přepínání slovníku na základě toho jaká klávesnice (jazyk) je aktuálně zvolena (MS Word).

Dalším ještě komplikovanějším problémem je detekce chyb ve slovech, která jsou napsána správným pravopisem, ale do věty ve skutečnosti nepatří. Jedná se tedy o druhý typ chyby popsané výše. V tomto případě nelze použít detekci na základě vyhledávání slova ve slovníků, ale je nutné přistoupit k mnohem sofistikovanějšímu přístupu, který vychází z přístupů ke korekci chyb. Z toho vyplývá, že detekce těchto chyb není většinou implementována v případě, že řešíme pouze detekci chyb a uživatelé si je poté opravují sami. Problematika oprav tohoto typu chyb bude rozebírána později.

Jedním z důvodů proč různé aplikace, které nějakým způsobem řeší kontrolu či korekci textu stejným přístupem, je skutečnost, že dané aplikace používají stejné externí knihovny. Ty často řeší pouze některé části problematiky (například pouze detekci). Důvodem je zejména to, že pokročilejší funkce jsou značně náročné na implementaci.

## 2.5 Korekce chyb

Většina současných aplikací pro korekci je postavena pouze na detekci chyb a následné opravě chyb uživatelem. Za korekci v tomto případě lze považovat pouze část, kdy se generují slova jako možní kandidáti pro nahrazení špatného slova. Tyto aplikace tedy mohou být jednodušší a mohou využívat pouze základní přístupy ke korekci. Použití pokročilejších přístupů, které budou rozebírány dále, ale není vyloučeno.

Situace se komplikuje v případě, kdy uživatel nemůže do procesu korekce zasahovat ať už z jakéhokoliv důvodu (množství textu atd.). Zcela automatickou korekci lze provádět s velkou přesností a automaticky víceméně pouze v situacích, kdy je korektor přizpůsoben nějakému konkrétnímu použití, kde je počet možných slov (slovní zásoba) omezená. Zástupcem této kategorie je například korekce příkazů pro nějaký program. Dalším podstatně složitějším příkladem je situace, kdy se má korekce provádět automaticky nad velkým objemem dat, který už nelze zpracovávat ručně. Této části problematiky se primárně věnuje tato práce. Dalším ty-

pickým příkladem, kdy nelze provádět korekci ručně je převod mluveného slova do textu. V rámci daného procesu se používá korekce textu, aby byla zajištěna co nejpřesnější detekce toho, co člověk vyslovil (Kukich, 1992). Automatická korekce bývá také často součásti OCR procesu.

Dalším aspektem, který ovlivňuje přístup k opravě chyb je skutečnost, zda jsou opravována jednotlivá slova nebo zda je k dispozici i kontext (předchozí a následující slova po chybném slově). V případě, že by bylo nutno provádět i kontrolu slov, která jsou ve slovníku, ale byla pouze zaměněna za jiné slovo, tak je okolní kontext nutností. Jinak by nebyla k dispozici žádná informace, na základě které by se dali vygenerovat kandidáti na korekci. Kontext také velice napomáhá s volbou opravy v případě, kdy ke slovům co nejsou ve slovníku, vygenerujeme víc než jednoho kandidáta.

Korekce se skládá z několika navazujících procesů. Nejprve se vytvoří seznam kandidátů, poté se daní kandidáti nějakým způsobem ohodnotí, vybere se nejvhodnější kandidát a nakonec se rozhodne, zda se oprava provede. Zde lze ilustrovat rozdíl mezi interaktivní korekcí a automatickými řešeními – proces generování kandidátů je víceméně stejný, hodnocení kandidátů ale nemusí být u interaktivní korekce natolik přesné, protože ještě prochází kontrolou od uživatele. Pokud je ale k dispozici řešení, které se používá pro automatickou korekci, tak je samozřejmě vhodné použít je pro hodnocení (řazení kandidátů) i u interaktivní korekce (pokud nejsme omezeni třeba výpočetním výkonem).

### 2.5.1 Typy korekce

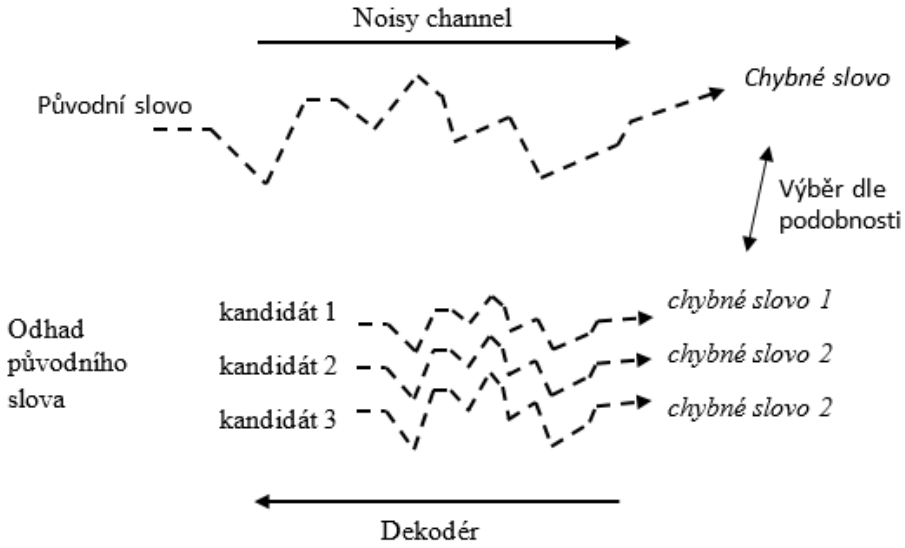
Pod pojmem typ korekce se rozumí způsob, kterým je hodnocena vhodnost jednotlivých kandidátů. Korekci chyb lze rozdělit na dva základní typy (Kukich, 1992). Jedním typem je situace, kdy je prováděna korekce slova bez informace o jeho okolních slovech. Slova, která jsou vlevo a vpravo od vybraného slova (chyby) jsou označována jako kontext. Druhý typ korekce je opak prvního a okolní kontext využívá k tomu, aby byla oprava chyb co nejpřesnější. Za třetí typ korekce je někdy považována pouhá detekce chyby, nicméně z našeho pohledu se o korekci nejedná, protože cílem této práce je implementace automatického korektoru pro velké objemy textu, pouhá detekce tedy není řešením. Typy korekce tedy jdou ruku v ruce s typy aplikací, které byly popsány výše.

Pokud jsou typy korekce vztaženy k typům chyb popsaných v kapitole 2.1, tak je zřejmé, že chyby ve slovech, která jsou součásti jazyka (jsou ve slovníku) lze opravovat pouze druhým způsobem s využitím kontextu, protože není k dispozici žádná jiná informace o tom, jakým směrem by se korekce měla vydat.

## 2.6 Noisy channel model

Tento model navrhl Shannon (2001) při matematické analýze komunikace. Model lze vysvětlit při pohledu z hlediska korekce textu následovně: chybné slovo je výsledkem poškození původního slova při přechodu přes komunikační kanál, který není dokonalý (obsahuje šum). Jurafsky (2009) popisuje šum jako formu substitucí

a dalších změn v písmenech slova, které komplikují rozpoznání pravého slova. Autor podrobněji vysvětluje princip modelu na následujícím obrázku.




Obr. 3 Princip Noisy channel modelu

Jurafsky (2009) dále uvádí, že cílem je vytvořit model přenosové cesty. Tento model je poté využit tak, že se vezme každé slovo daného jazyka a nechá se projít daným modelem. Následně se poté vybere slovo, které je nejbližší chybnému slovu. Noisy channel model je postaven na pravděpodobnosti a je dán rovnicí níže.

$$\hat{w} = \arg\max_{w \in V} P(w \mid x) \tag{1}$$

Znak ŵ zastupuje slovo, která je nejlepším odhadem modelu na původní slovo. Hledáno je tedy slovo w ze slovníku V, které má maximální pravděpodobnost vzhledem ke slovu x s chybou, které je opravováno. Jelikož je problém pracovat s touto pravděpodobností, tak je dále aplikována tzv. Bayesova věta. Tuto problematiku podrobněji rozebírá (George, 1992).

$$\hat{w} = \arg\max_{w \in V} \frac{P(w \mid x)P(w)}{P(x)} \tag{2}$$

Rovnici nicméně lze dále zjednodušit tím, že je odstraněn jmenovatel. To lze provést díky tomu, že P(x) je konstanta. Výsledná pravděpodobnost bude tedy maximalizována jak v případě předchozí rovnice, tak i v případě následující zjednodušené verze.

$$\hat{w} = \arg\max_{w \in V} P(w \mid x) P(w) \tag{3}$$

Díky tomuto zjednodušení lze říci, že výsledná pravděpodobnost se rovná součinu P(x|w), což je tzv. channel model a P(w) neboli prior model (Kernighan et al., 1991). V literatuře se dále objevují i jiná označení pro dané modely. Například Jurafsky (2009) užívá pro *channel model* název *error model* (chybový model) a pro prior model uvádí název *language model* (model jazyka). Dále tedy budou používány tyto výrazy, protože lépe vystihují, co se pod nimi skrývá.

### 2.6.1 Generování kandidátů

Prvním krokem při aplikaci přístupu definovaném výše je generování kandidátů slov. Z nich bude později vybrán nejpravděpodobnější kandidát na základě vypočítané pravděpodobnosti. Kernighan et al. (1990) navrhl použít metodu, kterou definoval Damerau (1964). Tato metoda identifikuje čtyři základní operace: vložení znaku, nahrazení znaku, smazání znaku a prohození dvou sousedních znaků. Na základě těchto operací dále definuje tzv. edit distance (editační vzdálenost řetězců). Pod tímto pojmem je myšlen minimální počet operací (ze zmiňovaných čtyř), který vede k tomu, že se z jednoho řetězce stane druhý.

Získané poznatky lze tedy využít při generování kandidátů, kdy se vezme slovo s chybou a pomocí možných operací definovaných výše jsou vygenerovány všechny možné varianty řetězců. To znamená, že se například vloží všechna možná písmena použité abecedy na každou pozici v textu. Dále se jednotlivá písmena ve slově postupně vymažou, prohodí či nahradí. Takto nám vznikne množina řetězců, které jsou ve vzdálenosti jedna od původního řetězce. Z této množiny poté vybereme pouze slova, která existují ve slovníku, čímž se jednak značně sníží počet kandidátů na opravu, ale hlavně zůstanou pouze relevantní kandidáti. Následující tabulka demonstruje generování kandidátů nad anglickým slovníkem. Tento příklad byl zvolen zejména z toho důvodu, že jsou zde vidět všechny možné druhy operací, které vedou ke vzniku kandidáta na korekci ve vzdálenosti jedna od původního slova.

Tab. 1 Generování kandidátů pro korekci

| Chyba | Korekce | Špatný znak | Správný znak | Pozice | Typ operace |
|---|---|---|---|---|---|
| acress | actress | - | t | 2 | mazání |
| acress | cress | a | - | 0 | vložení |
| acress | caress | ac | ca | 0 | prohození |
| acress | access | r | c | 2 | nahrazení |
| acress | across | e | o | 3 | nahrazení |
| acress | acres | s | - | 4 | vložení |
| acress | acres | s | - | 5 | vložení |

Zdroj: Kernighan et al., 1990

Znak „-“ reprezentuje nulový (prázdný) znak. Vygenerovaní kandidáti jsou poté dále předáni k ohodnocení použitým modelem.

Z příkladu je zřejmé, že správné slovo bude pravděpodobně skutečně ve vzdálenosti jedna od slova s chybou, protože kandidáti jsou relevantní. Damerau (1964), který navrhl tuto problematiku, dále uvádí, že 80% překlepů je ve vzdálenosti jedné operace od zamýšleného slova a téměř všechny chyby jsou do vzdálenosti dvě. Analýzou chyb se dále zabývali například Pollock a Zamora (1984), kteří dokonce uvádí, že chyby se vzdáleností jedna od správných řetězců (slov) dokonce tvoří 90-95% chyb. Skutečnost nicméně může být rozdílná, vzhledem k tomu, že autoři prováděli analýzu nad rozdílnými daty. V praxi bude tedy záviset na tom, jaký text bude opravován.

Pokud by bylo nutné provádět korekci s co největší přesností, tak je možné použít metodiku generování kandidátů i opakovaně a získat tak i varianty, které jsou vzdáleny více. Při zvažování do jaké vzdálenosti generovat kandidáty by se mělo vzít v potaz, jaké množství textu bude zpracováváno a jaký je požadovaný výkon, protože s opakovaným spouštěním roste poměrně rychle výpočetní náročnost.

### 2.6.2 Model jazyka (language model)

Jakmile jsou vygenerováni kandidáti na korekci, tak se přechází k jejich hodnocení. To vychází z modelu definovaného výše, který lze, jak již bylo řečeno, rozdělit na dvě části. Jednou z částí je tzv. language model, který ve zmíněné rovnici reprezentuje část P(w). Tato pravděpodobnost zachycuje, jak často se dané slovo vyskytuje v daném jazyce v případě, že ignorujeme kontext slova. V případě, že se bere v úvahu kontext slova, pak pravděpodobnost reprezentuje, jak moc často se dané slovo vyskytuje, obklopeno danými slovy. Ohodnocení pravděpodobnosti demonstruje následující tabulka, která zachycuje ohodnocení kandidátů z tabulky č. 1.

Tab. 2 Ohodnocení pomocí language modelu

| c (kandidát) | frekvence c | P(c) |
|---|---|---|
| actress | 1343 | 0.0000315 |
| cress | 0 | 0.000000014 |
| caress | 4 | 0.0000001 |
| access | 2280 | 0.000058 |
| across | 8436 | 0.00019 |
| acres | 2879 | 0.000065 |

Zdroj: Kernighan et al. (1990)

V tabulce je vidět, že momentálně je nejpravděpodobnější korekce slovem „across“. Je důležité zmínit, že konkrétní hodnoty vzešly z analýzy korpusu AP newswire z roku 1988, který obsahoval 44 miliónů slov. Kvůli zjednodušení navíc nebyl v úvahu brán kontext slova (byla tedy použita pouze frekvence slov v daném korpusu). Pokud má tedy slovo frekvenci 8436, tak se přesně tolikrát vyskytovalo v daném korpusu. Výsledná pravděpodobnost se tedy podělí celkovým počtem slov daného korpusu. V praxi by se spíše použily frekvence spojení dvou, třech či více slov (tzv. n-gramy). Zde ale v rámci zjednodušení autoři ilustrují model na unigramech. N-gramy budou podrobněji popsány v jedné z následujících kapitol.

Dále si všimněte, že ačkoliv slovo „cress“ nemá žádný výskyt, tak jeho pravděpodobnost není rovna nule. Důvodem je skutečnost, že nemáme vždy k dispozici informaci o frekvenci každého slova (protože ta záleží na použitém korpusu). V tom případě se používají různé techniky (např. vyhlazování), které tomuto problému předcházejí. Daná problematika bude podrobněji rozebírána později.

### 2.6.3 Chybový model (error model)

Podstatně složitější situace je v případě evaluace druhé části modelu a to tzv. chybový model (*error model*) neboli P(w|x). Tento model totiž musí nějakým způsobem modelovat aspekty vedoucí k tvorbě chyby v textu. Modelování chyb je poměrně složité, protože to může záležet na celé řadě faktorů. Jurafsky (2009) uvádí, že mezi ně patří například rozložení klávesnice, skutečnost zda je uživatel pravák či levák a dále na spoustě dalších faktorů, mezi které patří i proces vzniku chyb popsaný v kapitole 2.2.

Kernighan et al. (1990) navrhl metodu, která využívá podobného přístupu, jako při generování kandidátů. U daných slov tedy sleduje jaká operace a jaký konkrétní znak byl zaměněn, smazán, vložen či prohozen. Zároveň také sleduje, jaký znak dané editaci předcházel (sleduje se tedy kontext, tentokrát ale na úrovni znaků). Na základě tohoto přístupu lze poměrně dobře vytvořit způsob k hodnocení kandidátů.

Hodnoty pro jednotlivé operace a znaky ukládal Kernighan et al. (1990) do matic záměn (*confusion matrices*). Celkem byly tedy vytvořeny čtyři matice:

- del[x,y] - reprezentuje, kolikrát byl řetězec „xy“ napsán jako „x“
- ins[x,y] - reprezentuje, kolikrát byl řetězec „xy“ napsán jako „z“

- sub[x,y] - reprezentuje, kolikrát bylo „x“ napsáno jako „y“
- trans[x,y] - reprezentuje, kolikrát bylo „xy“ napsáno jako „yx“

Jurafsky (2009) doplňuje, že tyto matice byly vytvořeny (pro operace mazání a inzerci) se závislostí na předchozím znaku, nicméně lze je udělat i se závislostí na znaku následujícím. Autor ale dále nerozebírá, který z přístupů je výhodnější.

Následně bude proveden výpočet celkové pravděpodobnosti chybového modelu pomocí vzorců, které hodnoty z matic vztahují k velikosti korpusu, ze kterých byly matice generovány.

$$P(w \mid x) = \begin{cases} \dfrac{del[c_{p-1}, c_p]}{chars[c_{p-1}, c_p]} & \text{- mazání} \\ \dfrac{ins[c_{p-1}, t_p]}{chars[c_{p-1}]} & \text{- vložení znaku} \\ \dfrac{sub[t_p, c_p]}{chars[c_p]} & \text{- nahrazení znaku} \\ \dfrac{trn[c_p, c_{p+1}]}{chars[c_p, c_{p+1}]} & \text{- prohození znaků} \end{cases} \tag{4}$$

Následující tabulka zachycuje vypočtené pravděpodobnosti spolu s pravděpodobnostmi modelu jazyka a chybového modelu.

Tab. 3 Ohodnocení kandidátů pomocí Noisy channel modelu

| slovo c | frekvence(c) | p(c) | p(t\|c) | p(c) p(t\|c) | % |
|---|---|---|---|---|---|
| actress | 1343 | 0.0000315 | 0.000117 | $3.69 * 10^{-9}$ | 37% |
| cress | 0 | 0.000000014 | 0.00000144 | $2.02 * 10^{-14}$ | 0% |
| caress | 4 | 0.0000001 | 0.00000164 | $1.64 * 10^{-13}$ | 0% |
| access | 2280 | 0.000058 | 0.000000209 | $1.21 * 10^{-11}$ | 0% |
| across | 8436 | 0.00019 | 0.0000093 | $1.77 * 10^{-9}$ | 18% |
| acres | 2879 | 0.000065 | 0.0000321 | $2.09 * 10^{-9}$ | 21% |
| acres | 2879 | 0.000065 | 0.0000342 | $2.22 * 10^{-9}$ | 23% |

Zdroj: Jurafsky (2009)

Ačkoliv má největší pravděpodobnost, že chybné slovo bylo „actress“, tak se použije jako náhrada chyby slovo „acres“, protože je ve výsledcích dvakrát (byly totiž provedeny dvě různé operace mazání, postupně předposlední a poslední znak „s“). Z hlediska chybového modelu ale stačilo slovo „acres“ vyhodnotit pouze jednou, protože chybový model není závislý na kontextu.

### 2.6.4 Generování matic pro chybový model

Výše již bylo zmíněno, že se v rámci chybového modelu používají matice, které zachycují, jak moc často se daná operace s daným znakem provedla, aby bylo dosa-

ženo správného slova. Jinak řečeno zachycují, jak často došlo k chybě v závislosti na chybové znaku a dle typu chyby případně i na předchozím znaku. Mimo tyto matice ještě ve výpočtu figurují frekvence jednotlivých znaků a frekvence sekvencí dvou znaků. Všechny tyto informace (matice a výskyty) jsou nutné k tomu, aby bylo možné použít daný model.

Jurafsky (2009) uvádí, že lze matice získat vygenerováním ze seznamu chyb a jejich korekcí. Tento přístup má ale nevýhodu v tom, že bude i tak nutné získat nějakým způsobem informace o frekvenci znaků a sekvencí znaků. To lze vygenerovat z nějakého korpusu v daném jazyce. Kernighan et al. (1990) nicméně původně navrhl způsob, který využívá korpus rovnou i ke generování matic. To realizováno tak, že se matice nejprve inicializují na stejnou hodnotu a poté se spustí hledání chyb nad daným korpusem. Získané informace o chybách a korekcích se poté použijí pro aktualizaci matic a takto se spustí daná činnost opakovaně. Autor nicméně neuvádí, kolik iterací by se mělo provést. Druhé řešení má i přes to dvě výhody: nevyžaduje tolik ruční práce při přípravě listu známých chyb a korekcí a frekvence výskytů znaků a digramů jsou více relevantní vzhledem k maticím.

## 2.7 Slovníky a korpusy

Přesnost metod pro korekci textu je také značně závislá slovnících a korpusech, které využívají dané metody k provedení korekce. Slovníky a korpusy mají v procesu korekce textu rozdílnou roli. Slovníky se primárně používají v rámci detekce chyb, kdy se kontroluje, zda jsou slova na vstupu obsažena ve slovníku. Naproti tomu korpusy se používají ke generování dodatečných informací nebo samotných slovníků. Pod generováním informací jsou myšleny informace, které jsou dále využívány v různých aplikacích v rámci zpracování přirozeného jazyka. Konkrétním příkladem jsou frekvence různých řetězců (n-gramů). Problematika n-gramů je podrobněji popsána v následující kapitole. Korpusy jsou dále využívány i pro generování samotných slovníků. Použití slovníků a korpusů lze dále vztáhnout k jednotlivým typům aplikacím popsaným dříve. Pokud je cílem aplikace provádět pouze korekci textu, tak není nutné používat korpus (maximálně pro vygenerování slovníku) a generovat informace pro část korektoru.

### 2.7.1 N-gramy

V rámci *noisy channel* modelu definovaném výše byla využívána pravděpodobnost výskytu unigramů, což je v podstatě pravděpodobnost jednotlivých slov. Model tuto informaci využíval k tomu, aby mohl ohodnotit a zvolit nejvhodnějšího kandidáta na korekci. Pokud se tedy nějaké slovo vyskytovalo v trénovacích datech častěji, tak mělo větší pravděpodobnost výskytu. Tím pádem byl kandidát ohodnocen lépe. Ve skutečnosti se ale pouze na pravděpodobnosti výskytu daného slova nedá natolik stavět. To platí obzvlášť v případě plně automatické korekce, nebo pokud

se provádí i kontrola slov, které jsou ve slovníku (viz kapitola 2.1) se tedy používají n-gramy vyššího stupně[5]. Tím jsou myšleny slovní spojení dvou a více slov.

Jurafsky (2009) uvádí, že intuice na pozadí n-gramů vychází z toho, že je potřeba vypočítat pravděpodobnost slova *w* pomocí jeho historie *h*, neboli P(w|h). Příkladem může být například situace, kdy je prováděn výpočet pravděpodobnosti slova *„auto“*. Pravděpodobnost P(auto|nehodu způsobilo rychle jedoucí auto) lze tedy spočítat tak, že se vezme počet výskytů řetězce „nehodu způsobilo rychle jedoucí auto“ a počet výskytů řetězce „nehodu způsobilo rychle jedoucí“ a dané počty se vydělí. Jurafsky (2009) dále zmiňuje, že danou pravděpodobnost bude problém spočítat i kdyby byl jako korpus použit celý web, protože dané výskyty v konkrétní příkladu uvedeném výše nebudou natolik velké, aby dobře vypovídali o skutečné pravděpodobnosti.

Jelikož tedy nelze přistupovat k výpočtu pravděpodobností touto cestou, tak Jurafsky (2009) dále zmiňuje, že lze použít tzv. Markovův předpoklad (*Markov assumption*), který předpokládá, že se lze předpovědět pravděpodobnost výskytu slova bez toho, aby bylo zkoumán větší počet předchozí slov. Z tohoto předpokladu tedy vychází následující jednotky:

- bigram (pohled o jedno slovo zpět)
- trigram (pohled o dvě slova zpět)
- n-gram (pohled o n-1 slov zpět).

Základní rovnice pro výpočet pravděpodobnosti n-gramu je tedy následující:

$$P(w_i \mid w_1^{i-1}) \approx P(w_i \mid w_{i-N+1}^{i-1}) \tag{5}$$

Autor dále vzorec rozšiřuje pro výpočet pravděpodobnosti bigramů. Následující vzorec už je zjednodušen a neobsahuje ve jmenovateli sumu všech bigramů, které začínali slovem $w_{i-1}$, protože tato skutečnost je zachycena v počtu unigramů z pohledu slova $w_{i-1}$.

$$P(w_i \mid w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})} \tag{6}$$

Tento způsob výpočtu se nazývá *Maximum likelihood estimate* (dále MLE), protože maximalizuje pravděpodobnost výskytu n-gramu vzhledem ke zdrojovému korpusu. Daný přístup dobře ilustruje příklad na malém korpusu. Uvažujme tedy následující věty, které tvoří korpus (<s> a </s> značí začátek a konec věty, respektive kontextu).

```
<s>Moje auto je červené</s>
<s>Tvoje auto je modré</s>
<s>Vidím modré auto</s>
```

[5] Jako n-gramy jsou někdy označovány i sekvence n-znaků

Následující výpočty demonstrují výpočet pravděpodobnosti vybraných bigramů dle definovaného korpusu (vět) výše:

$$P(je|auto) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})} = \frac{2}{3} \qquad P(červené|\ je) = \frac{1}{2} \qquad P(Moje|<s>) = \frac{1}{3}$$

Je nutné zmínit, že jeden z důvodů, proč nemusí být n-gramy vnímány jako vhodná technika, k využití pro zpracování textu je zdání, že nezachycují jazyk z gramatického hlediska. Ve skutečnosti, ale informaci o gramatice do určité míry obsahují, což lze dobře demonstrovat na následujícím příkladu, podobnému tomu co uvádí Jurafsky (2009). Bude-li tedy uvažována např. věta „Byl jsem tam nakoupit suroviny“, tak při výpočtu pravděpodobností jednotlivých digramů ve větě dojít k těmto výsledkům.

$$P(jsem|byl) = 0.30 \qquad P(suroviny|nakoupit) = 0.0015$$

Dané pravděpodobnosti jsou dány pouze jako příklad jak by výpočet mohl dopadnout. Důležité tedy nejsou konkrétní hodnoty, nýbrž rozdíl v řádech výsledné pravděpodobnosti. Slova, která se v rámci jazyka používají často (např. jsem) totiž hrají výraznou roli v gramatice. Na hodnotách je tedy zřejmé, že digram obsahující tyto slova má řádově větší pravděpodobnost oproti slovům, které nejsou moc závislé na okolních slovech a které mohou být použity v různých kontextech.

### 2.7.2 Vyhlazování (smoothing)

V předchozí kapitole byly definovány n-gramy. Jejich použití v praxi je ale komplikovanější než by se mohlo na první pohled zdát. Korpusy, z kterých jsou generovány, mají totiž omezenou velikost. Díky tomu často nastává situace, že některé n-gramy se nevyskytují v korpusu ani v jedné instanci. To nicméně neznamená, že se v rámci daného jazyka n-gramy chybějící v korpusu nevyskytují. Tento problém byl nastíněn již v kapitole **Chyba! Nenalezen zdroj odkazů.**, nyní tedy bude představen podrobněji a doplněn o možné přístupy k jeho řešení.

Situaci lze nejlépe demonstrovat příkladem, je-li tedy uvažována věta „Moje avto má růžovou barvu“ s chybou ve slově „auto“, tak lze identifikovat pravý kontext chyby s kandidátem „auto“ jako „auto má růžovou“. Jelikož se ve skutečnosti moc často nevyskytuje auto v růžové barvě, tak je možné, že se daný trigram nevyskytuje ani v korpusu na kterém je model trénován, to ale neznamená, že dané větné spojení nemůže nastat. Ve větším korpusu by se dané spojení vyskytovat mohlo. Pokud by tedy byla tato chyba detekována, tak by byla pravděpodobnost daného trigramu rovna nule. Předpokládá se výpočet pomocí metodiky MLE, která byla popsána v předchozí kapitole 2.7.1.

Aby se tomuto problému předešlo, tak se používají techniky, které provádí tzv. vyhlazování (*smoothing*). Tyto techniky přiřazují pravděpodobnost i entitám, které se v korpusu vůbec nevyskytují. Díky tomu, bude korekce textu více spolehlivá a nevypadnout tak ze zvažování mohli žádní kandidáti kvůli nulové pravděpodobnosti jejich výskytu. Konkrétních metod, které tento problém řeší, existuje celá řada.

Zmínit lze například metody *Laplace*, *Good-Turing* či *Kneyser-Ney smoothing*, které se vyskytují často v literatuře, která se zabývá korekcí textu.

Za základní přístup je považována metoda *Laplace smoothing*, která je také nazývána *add-one smoothing* (přidání jedné). Jurafsky (2009) popisuje dané vyhlazování následovně. Nejprve definuje vzorec, který používá MLE a následně jej rozšíří:

$$P(w_i) = \frac{c_i}{N} \tag{7}$$

Kde $c_i$ je počet výskytů daného slova a N je počet všech výskytů slov v rámci slovníku. Následně je do vzorce doplněno tzv. přidání jedné.

$$P_{Laplace}(w_i) = \frac{c_i + 1}{N + V} \tag{8}$$

Zde je vidět, že ke každému výskytu slova c byla připočten jeden výskyt. To se tedy musí zohlednit i ve jmenovateli a tak se zde přičítá V, což je počet unikátních slov ve slovníku (protože je jednička přičtena ke každému slovu ze slovníku). Jurafsky (2009) dále zmiňuje, že se lze na danou situaci podívat i z jiného pohledu, kdy dochází ke snižování pravděpodobnosti u slov, které mají nenulový výskyt, a naopak zvyšování pravděpodobnosti u těch, co doteď měli nulový výskyt.

Vzorce popsané výše jsou pro určeny pro unigramy, v případě použití s bigramy, je nutné vzorec dále upravit:

$$P^*_{Laplace}(w_i \mid w_{i-1}) = \frac{C(w_{i-1}w_i) + 1}{C(w_{i-1}) + V} \tag{9}$$

Jurafsky (2009) demonstruje využití tohoto přístupu na testovacím korpusu, který obsahuje 1446 unikátních slov a celkem 9332 vět. Autor vybral několik slov a vytvořil z nich tabulku, která obsahuje počty výskytu jednotlivých digramů v korpusu. V převzaté tabulce níže jsou již počty všech výskytů inkrementovány o jedničku, jak definuje tento model vyhlazování.

Tab. 4 Výskyt digramů po přidání jednoho výskytu

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 6 | 828 | 1 | 10 | 1 | 1 | 1 | 3 |
| want | 3 | 1 | 609 | 2 | 7 | 7 | 6 | 2 |
| to | 3 | 1 | 5 | 687 | 3 | 1 | 7 | 212 |
| eat | 1 | 1 | 3 | 1 | 17 | 3 | 43 | 1 |
| chinese | 2 | 1 | 1 | 1 | 1 | 83 | 2 | 1 |
| food | 16 | 1 | 16 | 1 | 2 | 5 | 1 | 1 |
| lunch | 3 | 1 | 1 | 1 | 1 | 2 | 1 | 1 |
| spend | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 1 |

Zdroj: Jurafsky (2009)

Aby bylo možné provést výpočet pravděpodobností pomocí definovaných vzorců, je nutné uvést ještě výskyty daných unigramů v testovacím korpusu.

Tab. 5 Výskyty unigramů

| i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|
| 2533 | 927 | 2417 | 746 | 158 | 1093 | 341 | 278 |

Zdroj: Jurafsky (2009)

Následně mohou být vypočítány pravděpodobnosti digramů dle vzorce (9).

Tab. 6 Vypočtené pravděpodobnosti digramů

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 0.0015 | 0.21 | 0.00025 | 0.0025 | 0.00025 | 0.00025 | 0.00025 | 0.00075 |
| want | 0.0013 | 0.00042 | 0.26 | 0.00084 | 0.0029 | 0.0029 | 0.0025 | 0.00084 |
| to | 0.00078 | 0.00026 | 0.0013 | 0.18 | 0.00078 | 0.00026 | 0.0018 | 0.055 |
| eat | 0.00046 | 0.00046 | 0.0014 | 0.00046 | 0.0078 | 0.0014 | 0.02 | 0.00046 |
| chinese | 0.0012 | 0.00062 | 0.00062 | 0.00062 | 0.00062 | 0.052 | 0.0012 | 0.00062 |
| food | 0.0063 | 0.00039 | 0.0063 | 0.00039 | 0.00079 | 0.002 | 0.00039 | 0.00039 |
| lunch | 0.0017 | 0.00056 | 0.00056 | 0.00056 | 0.00056 | 0.0011 | 0.00056 | 0.00056 |
| spend | 0.0012 | 0.00058 | 0.0012 | 0.00058 | 0.00058 | 0.00058 | 0.00058 | 0.00058 |

Zdroj: Jurafsky (2009)

V tabulce jsou červenou barvou vyznačeny hodnoty, které nyní mají díky vyhlazování alespoň nějakou pravděpodobnost. Důležité je také zmínit, že hodnoty pravděpodobností nebyly rozloženy mezi jednotlivé digramy, které měli předtím nulovou pravděpodobnost rovnoměrně, protože v rámci vzorce pro normalizaci figuruje i hodnota unigramů (jmenovatel). To je výhoda, protože takto dané pravděpodobnosti přesněji reprezentuji skutečnost.

Aby bylo možné provést zhodnocení přínosu této metodiky vyhlazování, je nutné provést přepočet na výskyty po provedení vyhlazování. Díky tomu bude

možné porovnat výskyty před vyhlazení a po vyhlazení. Pro zpětný výpočet výskytů lze použít vzorec, který také uvádí Jurafsky (2009).

$$c^*(w_{i-1}w_i) = \frac{[C(w_{i-1}w_i)+1]\times C(w_{n-1})}{C(w_{i-1})+V} \tag{10}$$

Následně je pomocí daného vzorce vypočítáno rozložení výskytu po vyhlazení.

Tab. 7 Zpětně rekonstruované výskyt po vyhlazení

| | i | want | to | eat | chinese | food | lunch | spend |
|---|---|---|---|---|---|---|---|---|
| i | 3.8 | 527 | 0.64 | 6.4 | 0.64 | 0.64 | 0.64 | 1.9 |
| want | 1.2 | 0.39 | 238 | 0.78 | 2.7 | 2.7 | 2.3 | 0.78 |
| to | 1.9 | 0.63 | 3.1 | 430 | 1.9 | 0.63 | 4.4 | 133 |
| eat | 0.34 | 0.34 | 1 | 0.34 | 5.8 | 1 | 15 | 0.34 |
| chinese | 0.2 | 0.098 | 0.098 | 0.098 | 0.098 | 8.2 | 0.2 | 0.098 |
| food | 6.9 | 0.43 | 6.9 | 0.43 | 0.86 | 2.2 | 0.43 | 0.43 |
| lunch | 0.57 | 0.19 | 0.19 | 0.19 | 0.19 | 0.38 | 0.19 | 0.19 |
| spend | 0.32 | 0.16 | 0.32 | 0.16 | 0.16 | 0.16 | 0.16 | 0.16 |

Zdroj: Jurafsky (2009)

Z tabulky je zřejmé, že tento typ vyhlazování má značný vliv na výskyty slov, které měli před vyhlazením značně vyšší výskyty. Počet výskytů pro digram „want to“ se dokonce zmenšil z 608 před vyhlazením na 238 po vyhlazení. Na základě toho lze tedy dojít k závěru, že daná metodika není moc vhodná, protože provádí poměrně velké změny u hodnot s původním vyšším výskytem. V praxi lze tento dopad mírně zmírnit tím, že se nebude přičítat jednička k jednotlivým výskytům digramů, ale číslo o něco menší. I tak je ale vhodné spíše použít jinou metodiku řešící tento problém.

Jako alternativní metodika může být zvoleno Good-Turing vyhlazování. Jurafsky (2009) uvádí, že tato metoda je postavena stejně jako některé další metody na myšlence, že se pro stanovení pravděpodobnosti využijí entity (n-gramy), které byly zpozorovány v trénovacích datech pouze jednou. Jinak řečeno, frekvence singletonů (entity co se vyskytují pouze jednou) se využijí pro výpočet frekvence n-gramů, které nebyly zpozorovány ani jednou. Tato metodika je již o něco komplikovanější než metodika přidání jedné, která je zmíněna výše, z tohoto důvodu tedy nebude dále rozebírána a budou místo toho představeny alternativní metody k vyhlazování. Metodu Good-Turing případně podrobněji rozebírá Gale (1995). Dalším vhodným zdrojem zabývajícím se vyhlazování a analýzou vhodnosti jednotlivých metod je práce, kterou vytvořili Chen a Goodman (1996).

### 2.7.3 Alternativy k vyhlazování

Vyhlazování (*smoothing*) není jedinou cestou, jak řešit problém s nulovým výskytem n-gramů v trénovacích datech (korpusu). Jurafsky (2009) uvádí, že lze použít znalosti i znalosti z nižších n-gramů. Jednou z metodik, která je na tomto přístupu postavena, je interpolace. Tato metodika bere v potaz všechny informace (o n-gramech nižšího stupně) a kombinuje je dohromady tak, aby lépe vypovídali o skutečné pravděpodobnosti výskytu daného n-gramu. Pro interpolaci trigramů lze tedy použít následující vzorec.

$$\hat{P}(w_i \mid w_{i-1}w_{i-2}) = \lambda_1 P(w_i \mid w_{i-2}w_{i-1}) + \lambda_2 P(w_i \mid w_{i-1}) + \lambda_3 P(w_i) \qquad (11)$$

Kde součet $\lambda_{1\text{-}3}$ je roven jedné. Hodnoty $\lambda$ jsou váhy a určují, jak moc se lze na danou část modelu spoléhat. V případě, že by například data ohledně bigramů byly hodně kvalitní, tak je dobré mít danou váhu nastavenu tak, že aby tuto část modelu posílila. Tuto metodu lze dále vylepšit pomocí závislosti vah na kontextu.

Další alternativní metodou je tzv. couvání (*backoff*). Princip této metody je zřejmý již z názvu. Metoda vždy používá nejvyšší dostupný n-gram a v případě, že daný n-gram není nalezen, tak se použije n-gram nižšího stupně. Konkrétní metody postavené na tomto přístupu je např. *Katz smoothing* (Cheb a Goodman, 1996). Součástí metodik couvání je často i vyhlazování popsané v předchozí kapitole.

## 2.8 Alternativní přístupy ke korekci

Vývoj pokračoval dále.

### 2.8.1 Soundex

Doplnit

### 2.8.2 Vylepšení Noisy channel modelu

Doplnit

# 3 Literatura

BRILL, Eric; MOORE, Robert C. An improved error model for noisy channel spelling correction. In: *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics.* Association for Computational Linguistics, 2000. p. 286-293.

DAMERAU, Fred J. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 1964, 7.3: 171-176.

DEOROWICZ, Sebastian; CIURA, Marcin G. Correcting spelling errors by modelling their causes. *International journal of applied mathematics and computer science*, 2005, 15: 275-285.

GALE, William; SAMPSON, Geoffrey. Good-Turing smoothing without tears. *Journal of Quantitative Linguistics*, 1995, 2.3: 217-237.

GEORGE E.P. BOX, George E.P.George C. *Bayesian inference in statistical analysis*. Wiley classics library ed. New York: Wiley, 1992. ISBN 04-715-7428-7.

CHEN, Stanley F.; GOODMAN, Joshua. An empirical study of smoothing techniques for language modeling. In: *Proceedings of the 34th annual meeting on Association for Computational Linguistics.* Association for Computational Linguistics, 1996. p. 310-318.

JURAFSKY, Daniel. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition.* 2nd ed. Upper Saddle River: Prentice Hall, 2009, 1024 s. ISBN 01-350-4196-1.

KERNIGHAN, Mark D.; CHURCH, Kenneth W.; GALE, William A. A spelling correction program based on a noisy channel model. In: *Proceedings of the 13th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, 1990. p. 205-210.

KUKICH, Karen. Techniques for automatically correcting words in text. *ACM Computing Surveys (CSUR)*, 1992, 24.4: 377-439.

POLLOCK, Joseph J.; ZAMORA, Antonio. Automatic spelling correction in scientific and scholarly text. *Communications of the ACM*, 1984, 27.4: 358-368.

RUSSOM, Philip. Big data analytics. *TDWI Best Practices Report, Fourth Quarter*, 2011.

ŘEHŮŘEK, Radim; KOLKUS, Milan. Language identification on the web: Extending the dictionary method. In: *Computational Linguistics and Intelligent Text Processing*. Springer Berlin Heidelberg, 2009. p. 357-368.

SHANNON, Claude Elwood. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 2001, 5.1: 3-55.

VERBERNE, Suzan. Context-sensitive spell checking based on word trigram probabilities., University of Nijmegen, 2002.

WANG, Peiling; BERRY, Michael W.; YANG, Yiheng. Mining longitudinal Web queries: Trends and patterns. *Journal of the American Society for Information Science and Technology*, 2003, 54.8: 743-758.

WHITELAW, Casey, et al. Using the web for language independent spellchecking and autocorrection. In: *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*. Association for Computational Linguistics, 2009. p. 890-899.

# Přílohy

# A Matice

Test