TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: N 2612 – Elektrotechnika a informatika
Studijní obor: Informační technologie

Softwarová podpora přepisu přednášek z videozáznamů

Software support for transcription of lectures from video recordings

Diplomová práce

Autor: **Bc. Jan Rameš**
Vedoucí práce: prof. Ing. Jan Nouza, CSc.

V Liberci 21.5.2010

Stránka vyhrazena pro zadání práce!

Zadání je dvoustránkové, další stránka bude mít číslo 4!

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum 21.5.2010

Podpis

Děkuji svému vedoucímu prof. Ing. Janu Nouzovi, CSc. za jeho čas, materiály a pomoc poskytnuté během zpracování této práce. Dále bych chtěl poděkovat Ing. Jindřichovi Žďánskému, Ph.D. a Ing. Petru Červovi, Ph.D. za poskytnuté rady a konzultace.

Abstrakt

Tato práce se zabývá automatickým přepisem spontánní řeči především pro oblast přepisu přednášek s možností jejich oprav, nutnosti adaptace slovníků (gramatických modelů) a odlišností od diktovacích systémů. Součástí práce je také ukázka přístupu, jak nakládat s přepsanými texty s využitím webových technologií s důrazem na možnost spolupráce více osob při opravě přepisů. Výsledná aplikace je realizována jako server-klient, kdy klientská část využívá HTML a JavaScriptu společně s přehrávačem Flash k realizaci kompletního uživatelského rozhraní pro opravu a kontrolu přepisu, zobrazení výsledných titulků uvnitř videa pro snadné přehrávání. Dále nastiňuje způsob využití indexace pro nalezení relevantního obsahu v zaznamenaných přednáškách.

V první části je rozebrána problematika rozpoznávání řeči používaná v přepisovači a dalších použitých nástrojích.

Druhá část obsahuje popis konkrétních metod, knihoven a frameworků, které byly při tvorbě aplikace použity. V závěru této části jsou popsány podobné technologie, které se přepisu spontánní řeči také věnují.

V dalších kapitolách je nejprve nastíněn přístup řešení a následně jsou podrobně popsány přístupy k řešení daného problému s důrazem na jejich využití na webu. Závěr této části je věnován způsobům vyhodnocení úspěšnosti přepisů využívajícím různá měřítka pro vhodné zhodnocení úsilí, jaké musí být vynaloženo k opravě přepsaného textu.

Závěr práce je věnován vyhodnocení výsledků pomocí metod popsaných v části předchozí. Je zde také nastíněna řada oblastí, ve kterých je možné v řešení tohoto problému pokračovat, především je pak zdůrazněna nutnost adaptace slovníků pro daný obor přednášky.

Klíčová slova: skryté markovské modely, přepis spontánní řeči, webové technologie

Abstract

This work describes use of automatic transcription of spontaneous speech specifically for transcription of lectures. It also describes means for correction of such transcriptions together with the necessity of language model adaptation and other diversities from dictating systems. There is also shown an approach of how to deal with such transcriptions using web technologies with emphasis on collaboration of multiple people while correcting the transcriptions. Resulting application is server-client based, where the client side uses HTML and JavaScript with Flash based player to create a full featured user interface for correcting and administering the transcriptions. Another part of client application inserts the subtitles directly into the video image and shows a way how transcriptions can be used for indexing which allows users to find a specific and most relevant part of information they seek inside of the lectures.

First part of the work introduces speech recognition algorithms used in the transcription software and other tools used in this work.

Second part contains descriptions of individual methods, libraries and frameworks that were used for creating the application. At the end of this section there are described similar technologies that also deal with spontaneous speech recognition.

In several next chapters there is first shown an approach of how the resulting software could be implemented and then there are described all the methods and algorithms in detail with emphasis on web technologies. Finally there is also shown which methods were used for comparison of transcription success rate by using several different approaches which better describe the effort that needs to be made to correct the transcribed text.

Final chapters of the work show the actual results of tests (which algorithms were described in previous section) that were performed. There are also shown numerous areas that could further be developed in the future, and most importantly there is also noted that specific language models need to be created for each of the lecture fields.

Keywords: hidden Markov model, transcription of spontaneous speech, web technologies

Obsah

Seznam obrázků

Úvod

Cílem diplomové práce je zhodnocení možností hlasových technologií při využití k přepisu přednášek z video (či audio) záznamů a navržení dalších podpůrných prostředků k jejich správě a korekci. V průběhu zpracování práce byl vytvořen software, který bude možné využít ke správě, přehrávání a úpravě zachycených přepisů.

S rozvojem internetových (a síťových) technologií společně se zvyšováním výkonu počítačů rostou i možnosti využití hlasových technologií. Použití pro přepis přednášek je dalším krokem v jejich rozvoji. Přepsaný záznam je možné použít k podpoře studia přes internet (tzv. E-learning) s využitím vhodného uživatelského rozhraní, v prostředí internetu pak lze hovořit o webové aplikaci. E-learning je dnes velmi populární. Ve spojení s vysokorychlostním připojením a moderními technologiemi lze tak studentům nabídnout stažení přednášek či jejich sledování z pohodlí domova. Při zapojení hlasových technologií je navíc možné v přepsaném obsahu vyhledávat a nalézat přímo místo, které studenta na konkrétní přednášce zajímá. Vytvořením těchto možností může studenty motivovat k lepším výsledkům při jejich studiu i k úspoře času. S využitím webových a síťových technologií je navíc možné všechny materiály ukládat přehledně na jednom místě, kde budou snadno dostupné a umožní studentům najít požadovaný obsah snadno a rychle bez nutnosti procházet množství jiných webových stránek.

Nutnost realizace aplikace pro opravu přepisů plyne z toho, že přepis není stoprocentně úspěšný. Ani člověk vždy nerozumí přesně tomu, co slyší, ale na rozdíl od počítače má schopnost uvažovat tvůrčím způsobem a tak významu řeči porozumět. Tento systém nabídne přednášejícím (či jejich asistentům a dalším subjektům, kteří se na opravě chtějí podílet) možnost provádět opravy na přepsaném záznamu. Touto formou pak lze celý text validovat, aby neobsahoval přeřeknutí či drobné nepřesnosti vzniklé během výkladu. Vzhledem k tomu, že oprava se provádí v systému online, může probíhat i na jedné přednášce paralelně. Pomocí systému verzování je přitom vyloučeno přepsání již opraveného odstavce. Díky verzování a ukládání revizí je navíc možné sledovat kompletní vývoj každého odstavce.

1 Představení problematiky

1.1 Úvod do rozpoznání řeči

Rozpoznání řeči je komplexní úloha, jež sestává z pořízení zvukového záznamu, jeho konverzi do příznakových vektorů, které následně slouží detektoru pro určení výsledné textové (fonémové) reprezentace.

Již samo pořízení zvukového záznamu je velmi kritickou částí. Čím kvalitnější záznam je, tím úspěšnější bude i práce rozpoznávače. Zašuměné nahrávky, či nahrávky, kde přes sebe mluví více osob nebo se ozývá hluk pozadí, jsou pro detektor fatální. Při pořizování záznamu se také často provádí normalizace hlasitosti nahrávky, snahou je přiblížit hlasitost rozpoznávané nahrávky zvukovým datům použitým k trénování. Dalším krokem bývá využití tzv. preemfázového filtru, který má za úkol odstranit stejnosměrnou složku ze signálu.

Následně je signál rozdělen na okna (která se v polovině překrývají) a pro tato okna jsou spočítány příznaky. Jako jednoduché příznaky mohou sloužit energie signálu v daném okně, počet průchodů nulou nebo frekvenční spektrum součtované opět přes určitá okna. V modernějších rozpoznávačích se ale místo spektra využívá kepstrum. Kepstrum je logaritmované frekvenční spektrum přenesené zpět inverzní Fourierovou transformací do časové oblasti. Zde se již nemluví o čase ale o tzv. kefrenci. Ukazuje se, že kepstrum mnohem lépe charakterizuje řečový signál než frekvenční spektrum. Z těchto příznaků bývá často spočítána jejich první a druhá derivace (diference), které jsou použité jako další příznaky.

Dále je možné rozdělit rozpoznávače do dvou kategorií:

1. S pevným slovníkem – jako trénovací data jsou použita slova, která budou následně rozpoznávána, využívá se například v mobilních telefonech pro hlasové vytáčení.
2. S proměnným slovníkem – slova, která detektor bude umět rozpoznat, jsou sestavena z natrénovaných jazykových modelů a je tak možné do slovníku přidat téměř jakékoliv slovo.

Pro první kategorii se často používá značně omezený slovník a je tak možné použít jednodušší metodu vybírání nejlepšího slova, nazývanou DTW (dynamic time warping) neboli dynamické borcení času. Její hlavní nevýhodou je nutnost použití velkého množství

oken na jedno slovo a pro větší slovníky je takřka nepoužitelná (má příliš velkou výpočetní náročnost). Z tohoto důvodu se pro větší slovníky využívají rozpoznávače využívající HMM.

1.2 HMM rozpoznávače

Zkratka hidden Markov model neboli skryté Markovské modely. Využívá k popisu slova automat, který má předem definovaný počet stavů (například tři na jeden foném). Pracuje se přitom s pravděpodobností setrvaní v daném stavu, nebo přechodu na stav následující. Výrazně tak klesá počet operací, které je nutno provést k nalezení nejlepšího kandidáta pro dané slovo na rozdíl od DTW.

Při rozpoznávání plynulé řeči pak stačí propojit výstup automatu pro každé slovo s vstupem do slova následujícího. Při rozpoznávání se následně využívají slovníky či komplikovanější jazykové (řečové) modely.

1.3 Trénování HMM

Je potřeba si uvědomit, že člověk nemluví tak, jak je to napsáno v textu (v případě českého jazyka se tomu sice blíží, ale u jazyka anglického je to již zcela patrné). Základní jednotka řeči je foném, ten charakterizuje jedno či více písmen v psaném textu a je to i základní jednotka, s kterou rozpoznávače pracují.

Vlastní učení probíhá tak, že skupina osob vytvoří určitý počet nahrávek, z nichž je pořízen přesný textový a následně fonetický přepis (viz kap. 1.5) přesně anotovaný tak, aby bylo pevně dané, kde každý foném začíná a končí. Z těchto zdrojových dat je pak učící algoritmus schopen vytvořit model pro každý foném, případně další celky běžné řeči jako jsou pomlky (ticho), rušivé vlivy (zakašlání, hláska „aaa“ při odmlce apod.), tj. jejich akustický model. Akustický model je následně použit k sestavení stavů automatu.

Pro lepší funkci rozpoznávače (především pak u plynulé řeči) je vhodné přizpůsobit akustický model dané osobě a místu, kde je nahrávka pořizována. Výhodou tohoto postupu je, že není nutné, aby konkrétní osoba pořizovala velké množství nahrávek. Jako výchozí stav se použije obecný model, který je pouze adaptován na konkrétního mluvčího. Kvalita výchozího modelu je tím vyšší, čím více nahrávek máme k dispozici a čím více osob je namluvilo.

1.4 Rozpoznávání plynulé řeči

Rozpoznávání plynulé řeči[5] je ještě komplikovanější, protože není možné určit, kde každé slovo začíná a končí. Uvedu příklad, u kterého vynechám veškerou interpunkci a mezery:

Textový přepis: Strč prst skrz krk.

Fonetický přepis: Strčprskrskrk

Z příkladu je vidět, že jsou-li slova následována určitými slovy, může to zásadním způsobem ovlivnit jejich výslovnost a některé fonémy mohou být dokonce vynechány.

Úloha rozpoznávání spojité řeči využívá opět fonémové modely, kde jsou ale modely všech slov vzájemně zacykleny. Pokud bychom uvažovali slovník o 1000 slovech a větu o 10 slovech, existuje 1000^{10} možností jak tuto větu sestavit. Během rozpoznávání je proto potřeba vyhodnotit, jaká z kombinací slov je nejpravděpodobnější. Vyhodnocení všech kombinací je výpočetně nereálné, existují ale metody, které umožňují nezabývat se všemi kombinacemi a zaměřit se pouze na několik nejpravděpodobnějších. Jedná se o princip dynamického prohledávání s prořezáváním, kde se postupně nejslibnější kombinace prohledávají dál a ostatní se zahazují. Kromě výše uvedeného akustického modelu se tak navíc používá ještě model jazykový (gramatický nebo též slovník). Tento model postihuje strukturu jazyka a určuje pravděpodobnosti následností slov.

1.5 Tvorba slovníků

Pokud je tedy potřeba přidat slovo do slovníku, musíme napřed pořídit jeho fonetický přepis. Pro český jazyk lze využít sadu jednoduchých pravidel. Bohužel existují výjimky, které je třeba vzít při fonetickém přepisu v úvahu. Tyto výjimky je nutné přepisovacímu algoritmu explicitně vyjmenovat. Například

Slovo → fonetický přepis dle pravidel → korektní fonetický přepis

Diplomat → ~~Ďiplomat~~ → **Diplomat**

Takto vytvořený slovník nebere v úvahu kontext daného slova, jinak řečeno všechna slova mají stejnou pravděpodobnost výskytu. V realitě tomu tak ale není. Při snaze vytvořit co nejlepší přepisovací software, se pak používají ještě pokročilejší metody, kde se využívá tzv. bigramů (či trigramů například pro angličtinu, která má menší počet slov a je tedy možné z výkonnostních důvodů trigramy použít). Slovník tvořený bigramy nám říká, jaká je pravděpodobnost, že po slově A bude následovat slovo B. Zcela jistě je pravděpodobnější, že se v řeči vyskytne dvojsloví „Ahoj Honzo“ než „Ahoj psát“. K sestavení takovéhoto jazykového modelu je za potřebí velké množství dat.

1.6 Synchronizace času

Další částí úloh rozpoznávání řeči je metoda „forced alignment“. Jejím úkolem není nalézt přepis daného úseku nahrávky, ale časově zarovnat existující přepis dle daného záznamu. Výsledkem je označkovaný text, kde je každému slovu (případně fonému) přiřazen čas jeho výskytu v nahrávce. Použitím „forced alignmentu“ nezískáme přesnou pozici daného slova či věty v rozsáhlém záznamu, ale pokud mu jako vstup dáme jen část nahrávky, o které víme, že se v daném místě očekávaný text bude nacházet, je synchronizace velmi přesná.

Vstupem algoritmu bývá jednak zvukový záznam a také gramatika[1], která je tvořena seznamem slov tak, jak jsou za sebou. Tento postup je platný pro metodu použitou v HTK (viz kap. 2.3)

[1] Gramatika je mnohem obecnější a může do jisté míry sloužit k vytvoření jednoduchého jazykového modelu (modelu bez použití pravděpodobností ale s možností větvení)

2 Použité technologie

2.1 Newton Dictate

Newton Dictate je aplikace vyvíjena na TUL ve spolupráci s firmou Newton media, a.s. Jedná se o implementaci vlastního rozpoznávače s širokou podporou pro použití slovníků a adaptaci hlasových modelů. Program dále nabízí možnost přidání uživatelských maker, která v omezené míře umožňují rozšíření slovníku o další výrazy.

Aplikaci Newton Dictate jsem v průběhu práce použil k pořízení přepisů přednášek a k pokusům s přepisem spojených.

2.2 GWT

Google web toolkit (GWT) je vývojový kit a platforma pro tvorbu složitých optimalizovaných webových aplikací se zaměřením na aplikace běžící v internetových prohlížečích ve formě JavaScriptu bez nutnosti udržování různých verzí zdrojových kódů v závislosti na typu a verzi prohlížeče. GWT využívá pro psaní kódu jazyk Java, který při kompilaci převádí na sadu zdrojových kódu JavaScriptu, který pak umí prohlížeč interpretovat. GWT vytvoří několik permutací výsledného kódu zvlášť pro každý podporovaný prohlížeč. Výsledkem je výrazně optimalizovaná a kapacitně méně náročná aplikace.

GWT dále nabízí řadu rozmanitých widgetů, komponent webové stránky, které je možno použít bez nutnosti vlastní implementace. Tato galerie tak vývojáři značně šetří čas. Jedná se například o widgety typu menu, záložky, dialogy, butony, stromové struktury, a další.

GWT navíc umožňuje aplikace ladit přímo v Java kódu, využívá k tomu speciální plugin (zásuvný modul) do prohlížeče. Jako ladící nástroj může být použito prostředí Eclipse, pro které Google nabízí opět plugin, který ještě zvýší efektivitu vývoje.

GWT se neomezuje pouze na klientskou část. Je možné využít i interní mechanismy pro RPC (remote procedure call, vzdálené volání procedur) a pomocí nich velmi efektivně napsat i serverovou část, opět v jazyce Java. To umožňuje ladit serverovou i klientskou část v jednom integrovaném prostředí. Využití této serverové technologie ale není povinné. Další možností je využití například serveru PHP a technologie AJAX, pro kterou má opět GWT zabudovanou podporu.

Pro vývoj webových aplikací pomocí JavaScriptu lze také použít čistě JavaScriptové frameworky (které nepotřebují křížový překlad), jako například jQuery nebo Prototype. Ty ovšem nenabízejí takové pohodlí během vývoje. Především je jejich ladění v prohlížečích problematičtější. Jejich výhodou může naopak být snadnost použití u jednodušších aplikací.

2.2.1 RPC

Vzdálené volání procedur (Remote Procedure Call) je metoda, umožňující požádat vzdálený počítač o data nebo o provedení operace. Klient i server musejí sdílet stejný prototyp (hlavičku) funkce. Jakmile klient zavolá danou funkci, neprovede se na klientovi, ale místo toho se parametry odešlou na server (například s využitím technologie AJAX či JSON). Server požadavek přijme, vyřídí a obdobnou metodou vrátí data klientovi. Klient může buď přímo čekat na odpověď (tzv. blokující a méně používaná metoda) nebo nastavit callback (metodu, která se zavolá automaticky a asynchronně po vyřízení požadavku). Následně klient může reagovat na výsledek. Výhodou tohoto řešení je možnost „probublávání“ výjimek ze serveru na klienta a jejich efektivnější zpracování a zobrazení uživateli. Programátor se v tomto případě nestará o to, jak jsou data posílána mezi serverem a klientem, do jisté míry ani nemusí znát konkrétní nastavení serveru, pouze zavolá metodu a čeká na výsledek.

2.2.2 AJAX

Asynchronous JavaScript and XML (AJAX) je webová technologie specifikující, jakým způsobem se vyměňují data mezi klientem (resp. webovou aplikací) a serverem. V tomto případě je použit dokument XML, do kterého jsou uložena data a ty pak asynchronně odeslána na server. XML formát je z principu strukturovaný, a tak velmi snadno použitelný. Drobná nevýhoda této technologie spočívá v tom, že programátor musí mít znalost o URL adrese, na kterou má požadavek zaslat, aby byl serverem správně vyřízen.

2.2.3 JSON

JavaScript object notation (JSON) je také webová technologie sloužící k specifikaci způsobu výměny dat mezi klientem a serverem. Nemá tak volnou strukturu jako XML, ale umožňuje přenášet jednoduché datové typy, pole a objekty. Její výhodou je oproti AJAXu rychlost zpracování na klientovi. Zpravidla je možné použít nativní funkci prohlížečů k jejich převedení zpět do objektů a struktur.

2.3 HTK

Hidden Markov model toolkit (HTK), je multiplatformní soubor knihoven a stand-alone aplikací, které implementují funkce pro učení zvukových modelů, práci s příznaky a audio daty, umožňují i vlastní rozpoznávání.

V této práci jsem jej použil k „forced alignmentu“ na straně serveru a k vyhodnocení výsledků přesnosti rozpoznávače.

2.4 Hunspell

Hunspell je open-sourcová knihovna pro kontrolu pravopisu. Její služby využívá například balík OpenOffice.org. Na začátku práce jsem se pokoušel použít knihovnu Jazzy, která je plně implementována v Javě. Problém Jazzy ale spočívá v nutnosti použití značně omezeného slovníku. Jakmile jsem se pokoušel použít slovník s 3 000 000 výrazy, Jazzy se nepodařilo ani načíst a knihovna se pak stala nepoužitelnou.

Hunspell oproti tomu zvládá značně větší slovníky a na rozdíl od Jazzy používá formu regulárních výrazů a značkovaného textu namísto přímého vyjmenování všech slov. Díky tomu je možné načíst slovníky, které jsou pro český jazyk lépe použitelné. Rozdíl oproti Jazzy je v možnosti načtení slovníku s téměř stonásobkem výrazů.

2.5 Flowplayer

Flowplayer je open-sourcový flashový[2] přehrávač audio a video souborů s možností rozšíření o další moduly. Při softwarové realizaci jsem použil především modul pro přidávání textového obsahu do videa a tímto způsobem do něj vložit titulky. Dalšími použitými moduly jsou tzv. streamovací rozšíření.

2.6 Streamovací server

Streamování je způsob přenosu videa či audia, jeho výhoda oproti klasickému stažení celého souboru spočívá v tom, že je možné spustit video v kterémkoli čase bez nutnosti čekat, než se stáhne předešlá část.

Streamovací server je aplikace zodpovědná za dodání správných dat klientovy dle požadavku na přehrávání. V práci jsem použil dva typy serveru, které rozeberu podrobněji v následujících kapitolách.

[2] Adobe Flash je modul do internetového prohlížeče umožňující mnohem lépe využívat systémové zdroje, a tak rozšířit možnosti webových stránek. Ve Flashi jsou na internetu často k vidění hry případně reklamní či navigační panely a také výše zmíněné přehrávače (např. přehrávač na stránce YouTube).

2.6.1 HTTP pseudo-streaming

HTTP pseudo-streaming je jednodušší forma přenosu využívající HTTP protokol k přenosu dat. V principu se nejedná o klasický streaming, neboť server neposílá jen ta data, která klient aktuálně potřebuje. Namísto toho pouze zkopíruje hlavičky souboru (které popisují například pozici ve videu, od které se přehrává, jeho rozměry, použitý kodek[3] apod.), následně zahájí přenos a chová se stejně jako klasický HTTP přenos.

Jeho výhodou je snadná implementace a nasazení téměř na libovolný webový server bez větších požadavků na výkon a také možnost využití vyrovnávací paměti prohlížeče k uložení přehrávaných dat pro budoucí použití.

2.6.2 Red5 stream server – RTMP streaming

Server Red5 je „klasický“ steamovací server, v práci je pak využita jeho část sdílející data přes RTMP protokol. Jeho výhodou je možnost detailnějšího nastavení. Podporuje kontrolu zátěže a je vhodnější pro nasazení tam, kde je očekáváno větší vytížení. Umožňuje také rozložení zátěže na více serverů a práci v tzv. cluster[4] režimu. Další jeho výhodou je také možnost přehrávání živých přenosů, využití sdílených objektů mezi uživateli a lepší zabezpečení sdílených médií.

2.7 Podobné technologie

Odvětví rozpoznávání textu z volně mluvené řeči (resp. řeči často nespisovné, hovorové apod.) je relativně nové. Dosavadní nasazení rozpoznávačů bylo tam, kde uživatel věděl, že bude nahrávka určená pro přepis nebo v médiích, kde je kvalita záznamu i řeči na velmi vysoké úrovni.

Jedinou veřejnou technologií, která se podobá zadání této práce, využívá firma Google u služby YouTube. Google nejprve uveřejnil službu, která umožnila přidávat do videa titulky. Přidávání titulků je na YouTube realizováno pomocí nahrání speciálního souboru s časovými značkami.

Následně přibyla i funkce umožňující přepsat audio do textové formy. Tato funkce je ale zatím v testovací (Beta) verzi. Současně s tím již není nutné nahrávat soubor s titulky obsahující časové značky, YouTube provede synchronizaci („forced alignment”) sám. Možnosti přepsání řeči na text mohou využít nyní všichni uživatelé přes menu u videa. Tato

[3] Komprimační formát, ve kterém je video či audio zakódováno.
[4] Cluster je několik počítačů tvářících se na venek jako jeden celek.

funkce není ale povolena pro všechna videa. YouTube službu povoluje podle vlastního uvážení, přesný klíč mi znám není. Dle Googlu je funkce experimentální a ne vždy přesná, co se přepisu týká, ale má za úkol přiblížit videa sluchově postiženým návštěvníkům stránek.

Pokud video obsahuje titulky, je možné jej nalézt nejen pouze na základě komentářů ale i dle obsahu. Google dne 14. 7. 2008 uveřejnil službu „Google Elections Video Search gadget“, pomocí které jsou všechna videa z Politického kanálu YouTube automaticky přepisována a indexována, následně je pak umožněno v nich vyhledávat. V září téhož roku došlo k rozšíření služby a vzniku nové nazvané „Google Audio Indexing“ (GAudi) na Google Labs. Obě tyto služby ale nejsou v aktuální době k dispozici. Google zároveň podporuje pouze anglicky mluvené záznamy.

3 Cíle práce

Prvním cílem práce je zhodnocení hlasových technologií a možnosti jejich nasazení na úkol automatického přepisu přednášek. V tomto bodě se jedná především o možnosti aplikace stávajících slovníků na netradiční úlohu (úlohu, pro kterou slovníky nebyly navrženy) a přizpůsobení hlasových modelů tak, aby vyhovovaly pro konkrétního mluvčího a místnost a také danou záznamovou techniku. Pro zjednodušení práce je uvažován pouze jeden mluvčí nahrávaný v jedné místnosti při přednášení jednoho předmětu (v tomto případě jsem dostal k dispozici záznamy doc. RNDr. Pavla Satrapy, Ph.D. z předmětu Alternativní metody programování pořízené v učebně A11 budovy A Technické university v Liberci).

Druhým cílem je vybudovat počítačovou aplikaci sloužící k správě zdrojových záznamu a opravě přepsaných titulků. Při řešení tohoto úkolu je kladen důraz také na zhodnocení aktuálních IT technologií, které je možné použít při vytváření optimálního uživatelského prostředí jak z pohledu koncového uživatele, tak i z pohledu správce či vlastníka zdrojů dat, která tato aplikace využívá. Dále je třeba zvážit, jak bude uživatel tuto aplikaci používat a jaké prostředky mu budou vyhovovat při vlastní opravě přepsaného záznamu.

4 Návrh řešení

4.1 Pořízení textového přepisu

Nejprve jsem se zabýval pořízením textového přepisu jedné konkrétní přednášky. V prvním kroku jsem přepsal zhruba pět minut trénovacích dat. Tyto přepisy je nutné dělat ručně a do přepisu uvádět všechna slova tak, jak byla řečena i za podmínky, že byla špatně vyslovena či ukončena v půlce. Tato trénovací data jsem následně zaslal Ing. Petru Červovi, Ph.D., který mi na základě nich vytvořil adaptovaný hlasový profil vyžadovaný programem Newton Dictate. Z původních pěti minut, bylo možné použít přibližně pouze tři minuty kvalitnějšího záznamu tak, aby neutrpěla výsledná kvalita adaptovaného profilu. Za pomoci speciálního programu CSStudio jsem pokračoval v přepisu většího množství hlasových dat určených k adaptaci. Současně jsem dával větší pozor na kvalitu záznamu i srozumitelnosti dat v nahrávce. Vybíral jsem jen určité části záznamů tak, aby bylo možné je k adaptaci využít všechny.

Adaptovaný profil jsem následně využil pro přepis celé přednášky. V této době jsem měl k dispozici dvě verze hlasového profilu. Výsledek přepisu se u obou verzí na první pohled příliš neměnil (přesnější vyhodnocení v kap. 6).

Pro přepis přednášek jsem navíc používal slovník určený pro přepis novinových článků. Zde se ukázal první problém. Velké množství výrazů používaných ve značně specifickém oboru, jakým je programování (ale zajisté by se ukázalo, že tento problém bude vznikat i u dalších vysokoškolských oborů), nejsou ve slovníku obsaženy (např. slova jako lambda, epsilon, funkcionální programovací jazyk, dx, …). V tomto případě dojde ke zmatení přepisovače, který provádí porovnání rozpoznaných fonémů se statistickými daty obsaženými ve slovníku, z nichž pak rekonstruuje výsledný přepis. Následně jsou oblasti, ve kterých se tyto speciální výrazy vyskytují, chybně přepsány a navíc za nimi vzniknou další chyby, než se přepisovač zotaví. Pro ilustraci problému uvedu příklad:

Přepisovač přepisuje: … všichni že lambda je součástí …

Výsledek přepisu: … všichni že nám dá pět součástí …

Popis práce přepisovače:

- Přepisovač správně rozpozná slovní spojení „všichni že“

- Následně narazí na slovo „lambda“, to ale nezná a vybere v souladu se slovníkem nejlepšího kandidáta, který odpovídá zaznamenaným příznakům a současně se nejpravděpodobněji vyskytuje za slovem „že“, v tomto případě je to spojení „nám dá“
- Dále následuje slovo je, jelikož dle statistiky není pravděpodobné, aby se za slovem „dá“ vyskytovalo slovo „je“, dosadí opět v souladu se slovníkem slovo „pět“
- V tomto bodě už je přepisovač zotaven a správně rozpozná slovo „součástí“, protože dle slovníku je pravděpodobnost jeho výskytu za slovem „pět“ dostatečně vysoká

Je možné, že přepisovači zotavení trvá déle, než opět najde slovní spojení tak, aby se dle příznakových vektorů z nahraného záznamu a pravděpodobností ve slovníku shodovaly s tím, co bylo vysloveno.

Vytvoření kvalitního slovníku je náročný proces vyžadující desítky megabytů zdrojových textů. Pokoušel jsem se pořídit textové materiály vztahující se k oboru přednášek, ten je ale natolik specifický, že nebylo možné v době realizace práce sehnat dostatečné množství dat k natrénování kvalitního slovníku (viz kap. 8).

4.2 Přístupy k řešení software

Nejprve jsem si položil otázku, jaký model pro program zvolit. Nabízely se tři postupy:

1. Stand-alone aplikace – rozumí se tím aplikace běžící na PC obsahující veškeré programové vybavení potřebné k splnění všech úkolů, které na ní budou kladeny. Zároveň musí mít přístup k datům potřebným k jejímu běhu, konkrétně pak k jazykovým a hlasovým modelům.
2. Server-klient – část aplikace je spuštěna na serveru, části programového vybavení mohou být pouze na serveru a stejně tak i části dat. K realizaci je nutné mít výkonnější počítač (server) a méně výkonný klient. Mezi serverem a klientem musí navíc existovat síťové spojení s dostatečnou propustností k přenosu multimediálních proudů (streamů). Přístupy k řešení klientů jsou dva:
    a. Standardní klient – klientská, spustitelná přímo na klientu, jako běžná aplikace (například pod Windows či multiplatformní využívající virtuálního stroje Java apod.)

b. Webová aplikace – klientská část je spouštěna přes prohlížeč formou webové stránky s využitím skriptovacích jazyků (nejrozšířenější je v dnešní době JavaScript). Webová aplikace je z principu multiplatformní, vzniká zde problém nekompatibility webových prohlížečů.

Pro konkrétní realizaci jsem zvolil model server-klient. Pomocí tohoto přístupu je možné sdílet data mezi uživateli a umožnit tak jejich spolupráci při úpravě titulků. Zároveň jsou všechny důležité technologie a data uloženy na serveru a je tak znemožněno třetí straně, aby je získala bez souhlasu vlastníka (či držitele autorských práv). Na straně klienta běží webová aplikace. S využitím GWT (kap. 2.2) se do značné míry minimalizují problémy spojené s kompatibilitou prohlížečů. Vzhledem k tomu, že GWT využívá Javu jako výchozí programovací jazyk, je možné převzít i další množství zdrojových kódů, které nebyly původně pro web vůbec určeny. Webové technologie jsou v dnešní době na vzestupu a stávají se pro programátory progresivnějšími a má-li výsledná aplikace sloužit studentům, je mnohem výhodnější realizovat ji na webu. Stane se tak mnohem dostupnější, je možné ji lehce začlenit do informačních systémů školy a efektivně ji využít pro E-learning. Další nespornou výhodou webové aplikace je absence nutnosti aplikaci instalovat. Běží na každém počítači (v dnešní době již i mobilním telefonu) a je k ní možné přistoupit i z internetových kaváren po celém světě.

Malou nevýhodou tohoto řešení je nutnost přítomnosti přehrávače, který podporuje zobrazování titulků. Zvolil jsem přehrávač FlowPlayer (kap. 2.5), který se pro tento typ úlohy jevil vhodný vzhledem k možnostem jeho rozšíření pomocí zásuvných modulů. Nevýhoda spočívá v tom, že řada PC nedisponuje přehrávačem Flash, který je nutný pro spuštění přehrávače. Drobnou nadějí v tomto směru je webový standard HTML5, který by měl umožnit přehrávání videa přímo v HTML stránce bez nutnosti přehrávače. Tento standard ale momentálně není schválen, a tak i jeho podpora ze strany prohlížečů je značně omezená. Vývojáři prohlížečů, které tento standard podporují, se navíc nemohou shodnout na formátu (kodeku), který bude pro přehrávání použit. V novém standardu navíc chybějí metody na přidávání obsahu do videa, konkrétně například možnosti do něj vkládat titulky. Jakmile bude

standard přijat, vzroste množství podporovaných zařízení i o tablety či smartphony[5] (které prozatím Flash nepodporují) a tím i univerzálnost celé aplikace.

Dále jsem se zabýval způsobem uložení dat. Vzhledem k tomu, že je pro aplikaci použit model server-klient, je nejvhodnější volbou použít relační databázi. Její výhodou je snadná přenositelnost a snadná a efektivní manipulace s daty. Pro přenos dat lze využít mapování databáze do XML a následně je zpracovat na klientu, případně je transformovat do podoby JSON a použít JavaScriptový engine prohlížeče k jejich převodu na datové struktury. Oba tyto přístupy jsou v dnešní době hojně využívány, nicméně GWT nabízí vlastní metodu přenosu pomocí RPC (kap. 2.2.1) a do značné míry tak abstrahuje programátora od znalosti konkrétní formy přenosu (GWT nabízí dvě implementace RPC a je možné mezi nimi přepínat bez změny jediného řádku kódu). Data jsou tak přenášena ve formě objektů, které mají stejnou implementaci na straně klienta i serveru. Drobnou nevýhodou je značná komplikace při implementaci RPC bez použití GWT, ale přínos k urychlení vývoje je natolik značný, že jsem se rozhodl využít tohoto přístupu.

4.3 Realizace aplikace

Ačkoli použití frameworku GWT vede ke značnému zjednodušení, byl jsem nucen přijmout ještě jedno omezení v podobě podpory jediného prohlížeče (pro testovací účely). I přesto, že GWT usnadňuje vývoj pro více prohlížečů, nevede jeho použití k úplné nezávislosti na platformě. V tomto konkrétním případě hovořím především o zcela zásadní komponentě, kterou je „rich text box“ neboli textový editor s širší podporou formátování a editace textu WYSIWYG[6] formou. Implementace ze strany prohlížečů se často liší a já jsem byl nucen použít funkce, které nejsou předdefinovány v GWT a nejsou tak ani přenositelné mezi prohlížeči.

Jako cílovou platformu jsem vybral prohlížeč Mozilla Firefox. K výběru právě této platformy jsem měl dva důvody. Zaprvé je Firefox nejrozšířenějším prohlížečem na trhu (podle w3schools.com ho používá 46,2% uživatelů na Internetu, je tedy více než třikrát oblíbenější než Microsoft Internet Explorer 8 s podílem 15,3%) a je zároveň i multiplatformní. Druhým důvodem, proč jsem vybral právě tento prohlížeč, je počet

[5] Chytré mobilní telefony s přístupem na internet možností spouštění aplikací, velkou (často dotykovou) obrazovkou a výkonným procesorem.

[6] Zkratka „what you see is what you get“ neboli „co vidíš je to, co dostaneš“ označuje formu editace u textových editorů ať už přímého textu (Microsoft Word, OpenOffice.org Writer) nebo třeba HTML (Microsoft Office, Macromedia Dreamweaver)

vývojových nástrojů usnadňujících vývoj webových aplikací (např. Firebug, JavaScript debugger a další).

Další otázkou, kterou jsem si při řešení problému položil, bylo, co všechno by daná aplikace měla ve svém základu umět, aby bylo možné její budoucí praktické nasazení. Prvním z úkolů aplikace je správa videí, tj. pohled na všechna videa (přednášky), které jsou obsaženy v databázi a slouží k základní navigaci mezi nimi. Videa se dají zobrazit ve dvou dalších pohledech:

1. Editační pohled – v tomto zobrazení je možné upravovat titulky a provádět úkony s tím spojené. Navíc zde fungují klávesové zkratky, kterými je možné rychleji se orientovat ve videu během oprav.
2. Zobrazovací pohled – video je zobrazeno jako v klasickém přehrávači s titulky vloženými přímo do něj. Tento pohled se stane hlavním pro všechny, kdo chtějí obsah konzumovat, nikoliv se podílet na jeho úpravě a tvorbě. V tomto pohledu je také možné využít speciální funkce a přepis přeložit do jiného jazyka. Pohled navíc slouží i k zobrazení videa z výsledku hledání.

Posledním ze základních pohledů je hledání, kde je uživateli umožněno nalézt relevantní obsah ze všech přítomných videí a automaticky spustit přehrávání od nalezeného odstavce.

5 Realizace řešení

Tato kapitola se věnuje jednotlivým částem vybudované aplikace a jejich konkrétní implementaci.

5.1 Přihlášení

Pro práci s aplikací je nutné, aby byl uživatel přihlášen. Vzhledem k přijatému modelu výměny informací pomocí RPC (viz kap. 2.2.1) je ale systém náchylný na řadu útoků. Zmíním dva nejčastější, které útočníci často kombinují, jedná se o tzv. „cross-site scripting“ (XSS) neboli skriptování napříč doménami a „session hijacking“ neboli ukradnutí relace.

Výše zmíněné útoky lze provést dvěma způsoby[1]:

1. Neperzistentní útok:
    a. Mějme dva uživatele Petra a Janu. Jana často navštěvuje konkrétní webovou stránku, která vyžaduje, aby se přihlásila, a ukládá citlivé informace.
    b. Petr zjistí, že danou stránku lze napadnout pomocí XSS.
    c. Petr vytvoří odkaz, který zneužívá možnosti napadení, a odešle ho Janě E-mailem.
    d. Důvěřivá Jana klikne na odkaz, který spustí na stránce nebezpečný kód a odešle informace o Janině relaci Petrovi (toto je XSS konkrétní útok). Petr tuto informaci může použít k zjištění informací o Janě bez jejího vědomí.
2. Perzistentní útok:
    a. Mějme opět dva uživatele Petra a Janu.
    b. Petr odešle na server příspěvek (například do diskuze, knihy návštěv apod.) obsahující nebezpečný kód.
    c. Jakmile Jana otevře danou stránku, kód se provede a odešle Petrovi informace o Janině relaci.

Navíc je možné zcizit relaci i pomocí přímého sledování provozu v síti. Tomuto typu útoků lze nejlépe zamezit šifrováním (SSL přes protokol HTTPS).

V aplikaci je implementována standardní metoda přihlášení s využitím relací. K zamezení obou typů XSS útoků jsem využil funkce pro filtrování obsahu. Aplikace

nevyužívá vůbec parametry stránky a není možné první útok vůbec provést. Díky využití zpracování na úrovni RPC volání není možné do stránky vložit obsah v přímé podobě HTML kódu a tak provést druhý typ útoku.

Pro dodatečné zabezpečení uživatele a jako další ochranu proti útokům pomocí zcizení relace jsem využil svázání konkrétní relace s adresou počítače. V určitých případech, kdy útočník a oběť jsou na stejné lokální síti a adresa tedy není jedinečná, řeší tento problém navíc dodatečná informace tzv. „logon seed“. Jedná se o číslo, které je jedinečné pro aktivní relaci a není na klientovi uchováno v podobě cookie[7], ale jako proměnná v JavaScriptu. Toto číslo je regenerováno pokaždé, když uživatel obnoví stránku a zamezí útočníkovy přístup k jakýmkoli funkcím systému bez znalosti jeho aktuální hodnoty. Relace má navíc časově omezenou platnost při neaktivitě přihlášeného uživatele.

5.2 Způsob uložení dat (ORM – POJO)

Jako způsob uložení dat jsem zvolil relační databázi, konkrétně serverovou platformu MySQL. V aplikaci nevyužívám příliš specifika konkrétního SQL jazyka. V případě větší zátěže je možné celou databázi přesunout na výkonnější server a to jak z hlediska jeho hardwaru, tak i možnosti využití jiné platformy.

Vzhledem k tomu, že klient i server sdílejí společné datové objekty (v Javě nazývány POJO neboli „Plain old Java object“), které jsou velmi jednoduché a odrážejí strukturu tabulek (relací) v databázi, lze si práci s databází usnadnit pomocí mechanismu zvaného ORM (Object-relational mapping). ORM je mechanismus, kde se využívá specifika jazyka (v tomto konkrétním případě jazyka Java, ale ORM je možné použít ve většině objektových programovacích jazyků, které podporují typové informace o objektech případně jeho rozšíření nazývaném reflexe) a na základě informací o členských proměnných objektů je možné sestavit buď automaticky SQL dotaz na přidání, smazání, nebo úpravu záznamu, případně vyplnění proměnných v objektu, na základě dat stažených ze serveru. Není pak nutné pro každý konkrétní dotaz psát příslušný obslužný kód, což značně urychluje vývoj a zvyšuje čitelnost a abstrakci kódu. Jistou nevýhodou tohoto řešení jsou vyšší nároky na čas zpracování pomocí ORM, než programového zpracování výsledků.

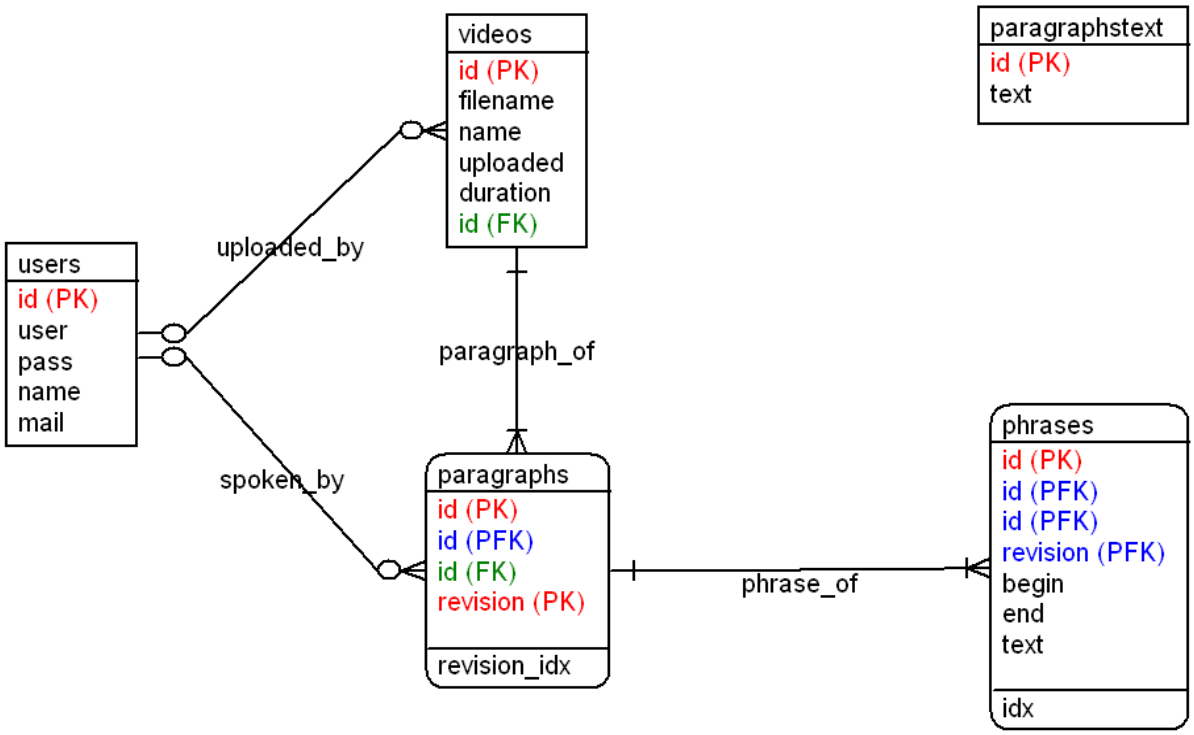
Ačkoliv existují knihovny pro Javu implementující ORM, nemohl jsem žádnou z nich použít z důvodů jejich nekompatibility s modelem, který využívá GWT. Většina těchto

[7] Informace uložená v prohlížeči, která je předávána serveru s každým požadavkem (běžný uživatel o existenci cookies nemusí být informován)

knihoven je určená pro model, kdy daná aplikace má neustálé spojení s databází a může tak přistupovat k datům až v okamžiku, kdy je aplikace potřebuje. Naproti tomu model server-klient v GWT vyžaduje, aby programátor rozhodl, které hodnoty je potřeba za databáze získat a všechny jsou následně ze serveru přeneseny na klienta. Z tohoto důvodu jsem vytvořil vlastní jednoduchou knihovnu implementující ORM za použití standardních anotací, které využívá i Java Persistence API (knihovna Javy, která se používá pro mapování objektů do XML nebo relačních databází), takže je možné použít stejné objekty s jinými frameworky pokud to bude do budoucna potřeba.

5.3 Struktura dat, odstavce, fráze

Strukturu uložení dat v databázi jsem navrhl na základě schématu XML (viz příloha B), který ve své aplikaci použil Ing. Martin Čičkán a která se stává platformou pro výměnu textových přepisů mezi různými aplikacemi vyvíjenými na Ústavu Informačních technologií a elektroniky TUL.



Obr. 1 Struktura databáze

Databáze obsahuje čtyři hlavní tabulky a její struktura je zobrazena na Obr. 1. Struktura databáze obsahuje čtyři hlavní tabulky. V tabulce **users** jsou uloženy záznamy o uživatelích, je jí využito během přihlašování a i všechna videa jsou s ní spojena tak, že každé video musí patřit nějakému platnému uživateli a současně i každý odstavec (titulek) musí být řečen platným uživatelem. V aktuální verzi programu ale možnost, že by v jedné nahrávce

hovořilo více osob, není implementována a u přepisů přednášek lze pro jednoduchost uvažovat pouze jednoho mluvčího. Vzhledem ke kompatibilitě se strukturou XML byla tato vazba zachována. Tabulka **videos** pak uchovává informace o nahraných videosouborech, jejich délce a umístění na disku (které je potřeba pro streamovací službu).

Další dvě tabulky (**paragraphs** a **phrases**) slouží k uložení vlastního přepsaného textu. Každé video má skupinu odstavců (titulků) a každý odstavec obsahuje několik frází. Fráze mohou být jedno i víceslovné, jejich pozice uvnitř odstavce je určena časem začátku a konce, přičemž není povoleno, aby se fráze časově překrývaly. Toto je zaručeno způsobem parsování (viz kap. 5.6) a synchronizací času po opravě (viz kap. 5.8).

Uložení jsem realizoval způsobem verzování, to znamená, že při každé úpravě se neztratí předchozí informace, ale přepíše se novější revizí. Tento způsob uložení má několik výhod:

- Správce nebo vlastník videa si může prohlédnout veškeré změny provedené ostatními uživateli a případně se vrátit k některé z předchozích verzí odstavce.
- Je možné automaticky vypočítat úspěšnost přepisu mezi prvotní verzí od přepisovače a koncovou verzí opravenou člověkem. Změny lze sledovat i opticky mezi jednotlivými revizemi včetně zvýraznění odebraných a přidaných částí textu.
- Lze snadno rozlišit opravené a neopravené odstavce.

Z důvodu přidání revizí jsem zavedl ještě dva pomocné databázové pohledy. Pohled **paragraphrev** vybírá z tabulky odstavců nejnovější revizi a pohled **paragraphlist** navíc k předchozímu pohledu přidává informaci o začátku a konci každého odstavce a počet frází, které daný odstavec obsahuje.

Nevýhoda řešení uložení jednotlivých frází v odstavcích s několika revizemi spočívá v rychlém nárůstu počtů záznamů v tabulce frází (pro ilustraci po opravení přednášky o trvání 65 minut bylo vytvořeno 15 362 záznamů). Pokud jsou ale pro vyhledávání v této tabulce použity vhodné indexy, je dopad na výkonnost systému minimální.

Struktura databáze navíc obsahuje pomocnou tabulku **paragraphstext**, která je použita jako fulltextový index. Tomuto tématu se budu blíže věnovat v kapitole 5.11.

5.4 Struktura aplikace

Pro vytvoření aplikace jsem použil modulární strukturu. Činnosti aplikace jsou rozděleny do jednotlivých pohledů. Tyto pohledy je možné registrovat a zakazovat pomocí konfigurace před kompilací. Aplikaci lze tak velmi snadno rozšířit o další funkce případně libovolnou stávající funkci odebrat. Zároveň s použitím dědičnosti objektů lze stávající pohledy rozšiřovat o další funkce bez nutnosti duplikovat zdrojový kód.

Výsledná aplikace obsahuje čtyři hlavní pohledy:

1. Seznam videí
2. Pohled zobrazovací
3. Pohled editační
4. Pohled vyhledávací

Funkce a realizace těchto pohledů bude popsána podrobněji v následujících kapitolách.

5.5 Správa videí

Seznam videí je základní pohled, odkud uživatel může přecházet k jejich editaci nebo zobrazení. V aplikaci je zobrazen formou tabulky obsahující snímek videa a jeho jméno, obdobně jak videa zobrazuje například průzkumník Windows. Po najetí myší jsou zobrazeny další údaje jako délka trvání, jméno přednášejícího atd.

Součástí tohoto pohledu by do budoucna měla být i možnost video na server nahrát, smazat, upravit název a další administrační akce. Ve stávajícím řešení je použita zjednodušená varianta, kdy aplikace očekává, že registraci videa příslušnému přednášejícímu provede buď uživatel ručně v databázi anebo jiný systém video automaticky zaregistruje po jeho nahrání například po dokončení přednášky.

Nahrání videa uživatelem současně nese problém kvůli velikosti videa. Pokud by se počítalo s nahráním přes internet, může to být časově náročná operace, proto vnímám možnost registrace videa aplikací třetí strany jako výhodnější speciálně v případě přepisů přednášek, kdy je možné využít lokální počítačové sítě zbudované v infrastruktuře univerzity.

5.6 Zobrazení videí

Pro zobrazení videa jsem použil přehrávač FlowPlayer (viz kap. 2.5). Součástí FlowPlayeru je i JavaScriptové API, které jsem adaptoval pro využití s GWT. Pomocí tohoto

API je možné přehrávač začlenit do webové stránky a dále s ním manipulovat a přidávat zpětné notifikace o provedených akcích, například v případě, kdy uživatel spustí, zastaví nebo přesune pozici ve videu, je aplikace upozorněna. Toho se využívá pro zobrazování titulků ať už přímo ve videu, kde je využito rozšíření přehrávače o modul kreslení textu do videa, nebo mimo video v editačním pohledu.

Pro zobrazování titulků je použit časovač, který je spuštěn při začátku přehrávání videa a v časových intervalech kontroluje čas ve videu. V příslušných časech je pak provedena výměna titulků. V editačním okně jsou navíc zvýrazňovány jednotlivé fráze v konkrétních časech, jak byly rozpoznány nebo zarovnány.

5.7 Parsování přepisu, dělení na odstavce

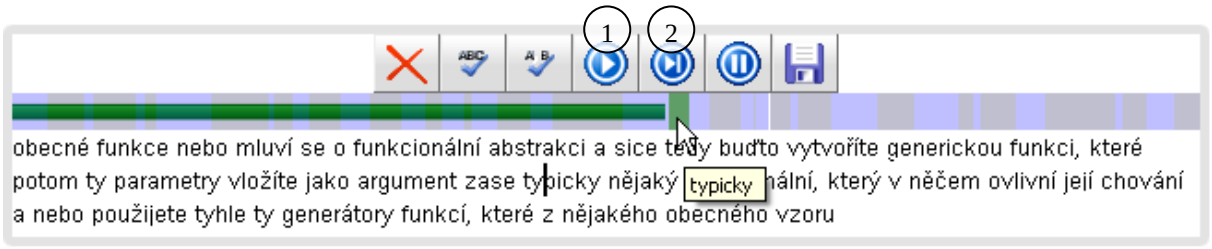
Po získání přepisu přednášky pomocí Newton Dictate jsem se zabýval rozdělením zdrojového souboru ve formátu TXZ (viz příloha C), který označuje fráze velmi jednoduchým způsobem pomocí časových značek začátku a konce každé fráze, na odstavce. Formát TXZ podporuje odstavce, ale uživatel je musí explicitně rozdělit v programu Newton Dictate.

Pro tento účel jsem vytvořil třídu `TXZParser`, která vrací postupně jednotlivé fráze, dokud nenarazí na konec TXZ souboru. Fráze jsou vraceny jako POJO (viz kap. 5.2) a je tak možné je přímo ukládat do databáze. Pro stanovení zalomení odstavců (titulků) jsem použil čas mezi frázemi. Vždy se porovnává čas předchozí fráze a nové fráze, pokud přesáhne určitou hranici, je vložen nový odstavec. Pokud je čas stanoven příliš krátký, je vygenerováno velké množství odstavců, pokud je naopak příliš dlouhý, jsou odstavce zbytečně dlouhé a je pak obtížné je zobrazit uvnitř videa. Zavedl jsem proto dynamické omezení času v závislosti na počtu frází v titulku. Začíná se na omezení 5s pauza pro odstavce s méně jak 20 frázemi, pokud počet frází přesáhne 20, je třeba už jen 1s pauza, pokud se počet frází zvýší na 30, je pauza omezena na 0,5s. Tyto časy a počet pravidel jsou jednoduše uživatelsky modifikovatelné a je tak možné vyladit systém tak, aby co nejlépe zalamoval titulky dle požadavků.

Uživatel nahraje z editačního pohledu soubor TXZ, ten je automaticky rozdělen a vložen do databáze, uživatel pak může ihned začít s opravou textu nebo pomocí aplikace rozdělit příliš dlouhé titulky na menší celky. Uživatel vybere frázi, před kterou se má odstavec zalomit, a aplikace jej rozdělí bez ztráty zarovnání textu a řeči. Rozdělení titulků je ale možné provést i po provedení opravy bez ztráty synchronizace s řečí.

5.8 Způsob editace

Po nahrání přepisu lze aktivovat režim úprav v editačním pohledu. Realizoval jsem ho pomocí textového editačního prvku „rich text box“. Jedná se v podstatě o další webovou stránku uvnitř editačního okna, kterou lze upravovat WYSIWYG cestou obdobně jako v Microsoft Wordu. Režim úprav formátování jsem ale nepoužil, namísto toho jsem využil funkce, kdy je možné do editoru vkládat neviditelné položky a následně jich využít k určení pozice v textu. K vlastnímu editoru je navíc připojen panel nástrojů sloužící k ovládání přehrávače a provedení dalších akcí a také ukazatel pozice (progress bar), který zobrazuje pozici videa vůči aktuálně upravovanému odstavci a pozice slov před editací. Všechny tyto funkce mají uživateli sloužit pro snadnější orientaci v textu během oprav.



Obr. 2 Režim úprav

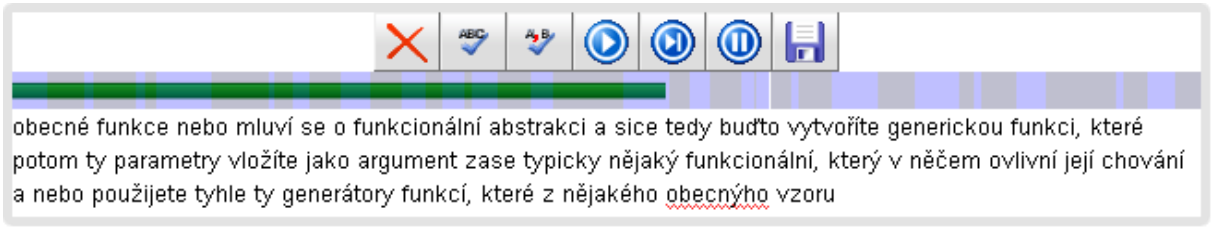
Obr. 2 zobrazuje vzhled celkového řešení. Zelenou barvou v ukazateli pozice je vyznačena aktuální pozice v textu, na které stojí kurzor vkládávání, zatímco kurzor myši (pokud jím uživatel ukáže na pozici slova) zobrazí „bublinu“ se slovem na dané pozici. Slova jsou oddělena barevně, aby bylo lépe vidět, kde jsou jejich začátky a konce. Zelený pruh pak označuje pozici videa v rámci aktuálně upravovaného odstavce. Pokud uživatel klikne na ukazatel pozice, začne se od této pozice přehrávat video a pozastaví se, jakmile narazí na konec odstavce. Kombinací těchto přístupů pak uživatel přesně ví, v jakém místě opravy se nachází a může tak snadno koordinovat opravy s poslechem mluvené řeči. Tlačítkem 1 je možno přehrát celý titulek od začátku, případně od místa označeného tlačítkem dva, uživatel pak může přehrávat video až od místa, kam došel s opravou a nemusí poslouchat řeč pokaždé od začátku. Přehrávač je navíc možné ovládat klávesovými zkratkami a řídit jimi skoky dopředu a zpět a pozastavovat nebo spouštět video.

Jak jsem se již výše zmínil, aby bylo možné zobrazovat aktuální pozici v textu, musí v něm být zobrazeny neviditelné značky. Nejsou přitom označena slova, ale mezery mezi nimi. To má tu výhodu, že pokud uživatel smaže slovo, značka v mezeře zůstane. V nejhorším případě uživatel smaže skupinu slov a mezer, takže aktuální pozice bude

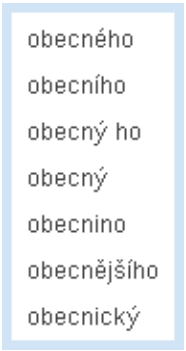
roztažena přes více původních slov, ale stále zůstane informace o přibližné aktuální pozici. Zřídkakdy se stává, že je třeba nahradit celý odstavec novým textem, a pokud zůstane několik slov na původních místech, stačí to k tomu, aby byla informace o pozici přijatelně přesná. Pro znovuobnovení přesné pozice je nutné provést synchronizaci (viz kap. 5.9).

5.8.1 Kontrola pravopisu

Jakmile uživatel dokončí opravu, může si nechat text zkontrolovat pomocí zabudované kontroly pravopisu využívající knihovny Hunspell (viz kap. 2.4). Touto kontrolou lze snadno odhalit překlepy a další pravopisné nedostatky, které mohly během opravy vzniknout. Systém odešle na server text ke kontrole. Server vrátí chybné výrazy, aplikace je v textu vyhledá a jsou zvýrazněny uživateli (Obr. 3). Uživatel může buď chybu ručně opravit, nebo na ni kliknout a aplikace opět požádá server o seznam přijatelných náhrad (Obr. 4).



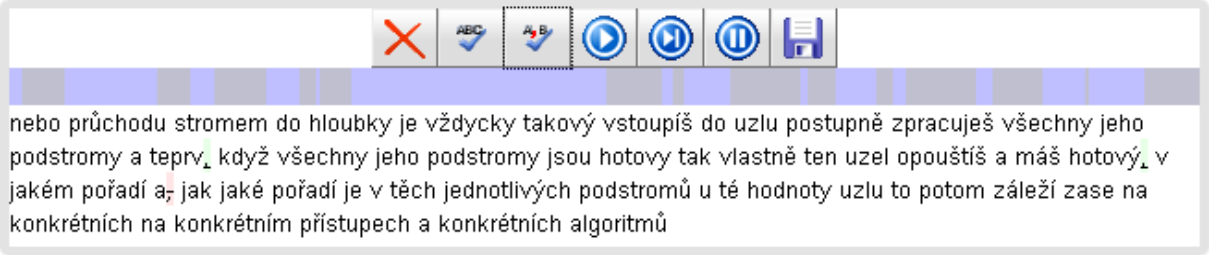
Obr. 3 Kontrola pravopisu - zvýraznění chyb



Obr. 4 Kontrola pravopisu - náhrady

5.8.2 Doplnění čárek

Pro lepší čitelnost textu je v systému zabudována možnost automatického doplnění čárek pro nejčastější spojky. Pokud uživatel této možnosti využije, je text opět zaslán na server a jsou do něj doplněny čárky, výsledek se uživateli následně zobrazí formou zvýraznění rozdílů (Obr. 5). Je pak zřetelně vidět, kde byly čárky doplněny a naopak, kde by měly být odebrány, jsou kontrolovány i spojky, před kterými by spojka být neměla.



Obr. 5 Doplnění čárek

Pro kontrolu jsem použil metodu, jejímž vstupem jsou dva seznamy, jeden pro spojky, před kterými by mněla být čárka a druhý pro spojky, kde čárka naopak být nesmí. Celý text se projde po slovech a kontroluje se, jestli slovo (případně dvojice slov) jsou spojkou a má, či nemá, před ní být čárka. Text se zpátky sestavuje již v opraveném stavu.

5.9 Synchronizace času

Jakmile uživatel přepis dokončí a odešle ho na server, obsahuje pouze jednu frázi, která zahrnuje veškerý text v odstavci. Proto, aby všechny zabudované nástroje mohly správně fungovat, je nutné text napřed rozdělit po slovech a přiřadit k nim časové značky začátku a konce. Toto je provedeno pomocí synchronizace času („forced alignment“ viz kap. 1.6).

Pro synchronizaci času jsem využil HTK (viz kap. 2.3), jeho nástroje ale vyžadují sestavení několika souborů, aby bylo zarovnání možné provést. Nejprve je nutné z videozáznamu vyříznout zvuk pro zvolený odstavec. To lze provést pomocí nástroje HCopy. Pro zvýšení výkonu je pro každý záznam audio předzpracováno do podoby příznaků, které HTK využívá. Ze zdrojového souboru je tak vykopírována část podle začátku a konce odstavce. Další z potřebných souborů je gramatika popisující strukturu věty. Vytvářím ji tak, že rozděluji přijatý odstavec po slovech a mezi jednotlivá slova vkládám nepovinnou pomlku. To znamená, že během zarovnání nemusejí slova navazovat za sebe, ale mohou mít mezi sebou pauzy (to je u spontánně mluvené řeči často pozorovatelné). HTK umožňuje použití značně složitějších gramatik, kde je možné ji větvit do stromu a vytvářet tak model například pro objednávkový systém nebo telefonní spojovatelku, kde se rozpoznává telefonní číslo pomocí sady cifer rozpoznaných klasifikátorem. Takto složité gramatiky ale nejsou pro synchronizaci potřeba. Gramatiku je následně nutné převést z textového formátu do formátu automatu, se kterým pracuje synchronizační program pomocí programu HParse. Poslední vstupní soubor, který je potřeba vytvořit, je výslovnost jednotlivých slov zapsaných pomocí fonémové abecedy, kterou HTK používá.

K převodu do fonémové abecedy využívám dvou kroků. Nejprve převedu výchozí text do fonémového přepisu ve formátu PAC[2] (Phonetic alphabet for Czech), který následně překonvertuji do formátu HTK. Formát PAC má výhodu v tom, že každý foném zabere právě jedno písmeno a nepotřebuje oddělovače mezi fonémy, zatímco formát využívající HTK tuto výhodu nemá a hůře se s ním pracuje. Konverze do psaného textu na fonémovou reprezentaci není jednoznačná úloha. Kromě pravidel upravujících znělost hlásek za určitých podmínek (spodoba znělosti, spodoba artikulační, a další) existují navíc výjimky (například slovo diplomat se nepřepíše jako ďiplomat, ale jako diplomat, fonémová abeceda využívá jen jednoho i). Z tohoto důvodu nelze realizovat pravidla jen formou automatu. Pro konkrétní realizaci jsem zvolil použití jednoduchých přepisovacích pravidel (di→ďi, ts→c, atd.), které jsou postupně aplikovány na zdrojový text, výsledkem pak je fonémový přepis bez aplikací výjimek. Vhledem k tomu, že výjimky často neupravují výslovnost slova zásadně, je možné provádět synchronizaci i bez nich a tím částečně (ale nikoli zásadně) snížit její přesnost. Konverzi do formátu HTK jsem implementoval prostým nahrazením fonému z abecedy PAC za odpovídající protějšek z abecedy HTK a oddělovač.

Dalším vstupem do algoritmu je zvukový model natrénovaný pro každý foném. V ideálním případě by se mělo jednat o stejný hlasový model, který byl použit při přepisování textu. Ten jsem ale neměl k dispozici ve formátu pro HTK, a tak jsem použil hlasový model vytvořený pro testovací účely během cvičení z předmětu Pokročilé metody rozpoznávání řeči. Tento model byl natrénován z nahrávek 6 osob (1 ženy a 5 mužů). Výsledná přesnost zarovnání je i s tímto modelem přijatelná a pro účely aplikace dostačující

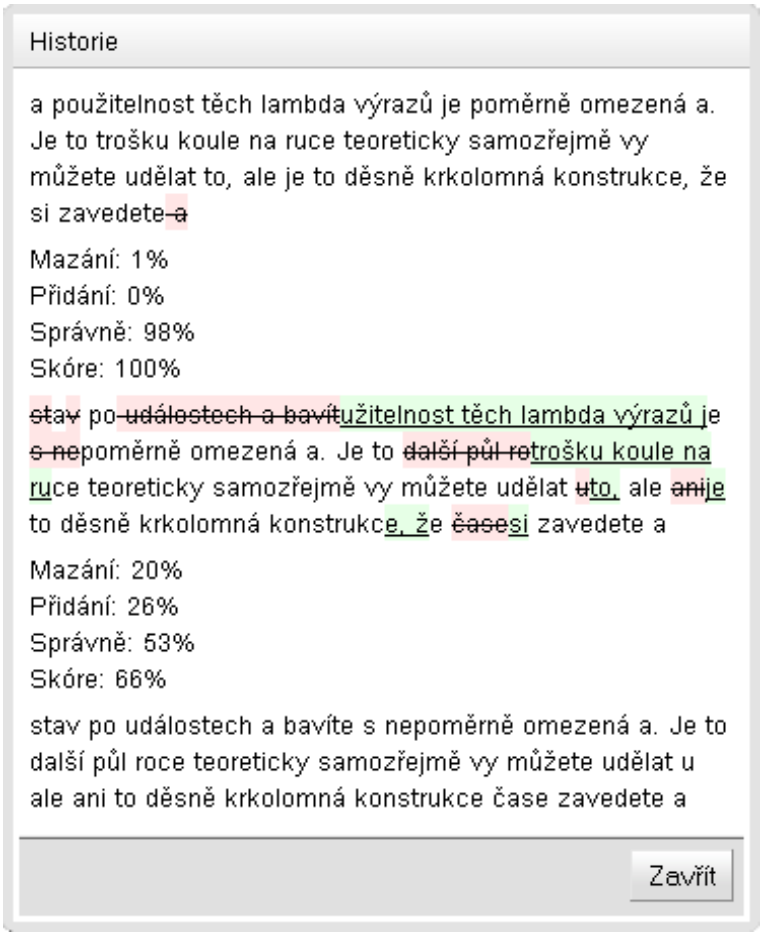
Jakmile je zarovnání dokončeno, je výsledek rozparsován do objektů POJO a ve formě frází s časovými značkami zapsán do databáze.

5.10 Ukládání a porovnávání historie

Jak jsem se již zmínil v kapitole 5.3, jsou odstavce a fráze ukládány formou revizí. Před uložením systém zkontroluje, zda je upravovaná revize poslední revizí v databázi. Tím je zamezeno tomu, aby si uživatelé navzájem přepisovali již opravené odstavce, pokud danou přednášku opravují současně. V tomto případě je uživatel upozorněn a musí odstavec obnovit, aby mohl provést změny.

Pokud má daný odstavec více revizí, je navíc možné je mezi sebou porovnávat a zjišťovat tak změny, které byly v průběhu úprav provedeny. Navíc společně s každou zobrazenou revizí je možné vypočítat procentuální zastoupení změn. Při porovnání jsou

zvýrazněny zvlášť smazané, přidané a nezměněné části textu. Porovnávací algoritmus je podrobněji rozebrán v kapitole 5.14.2. Pro každou z těchto části je následně vypočteno její zastoupení. Parametr skóre označuje, jak moc úsilí bylo nutné vynaložit pro provedení dané změny. Je vypočteno jako počet správných částí vůči délce nového odstavce. Pokud tedy uživatel opraví celý odstavec, ihned vidí, jak moc byl přepisovač úspěšný na každém konkrétním odstavci. Konkrétní vzhled dialogového okna zobrazení historie je na Obr. 6.



Obr. 6 Zobrazení historie

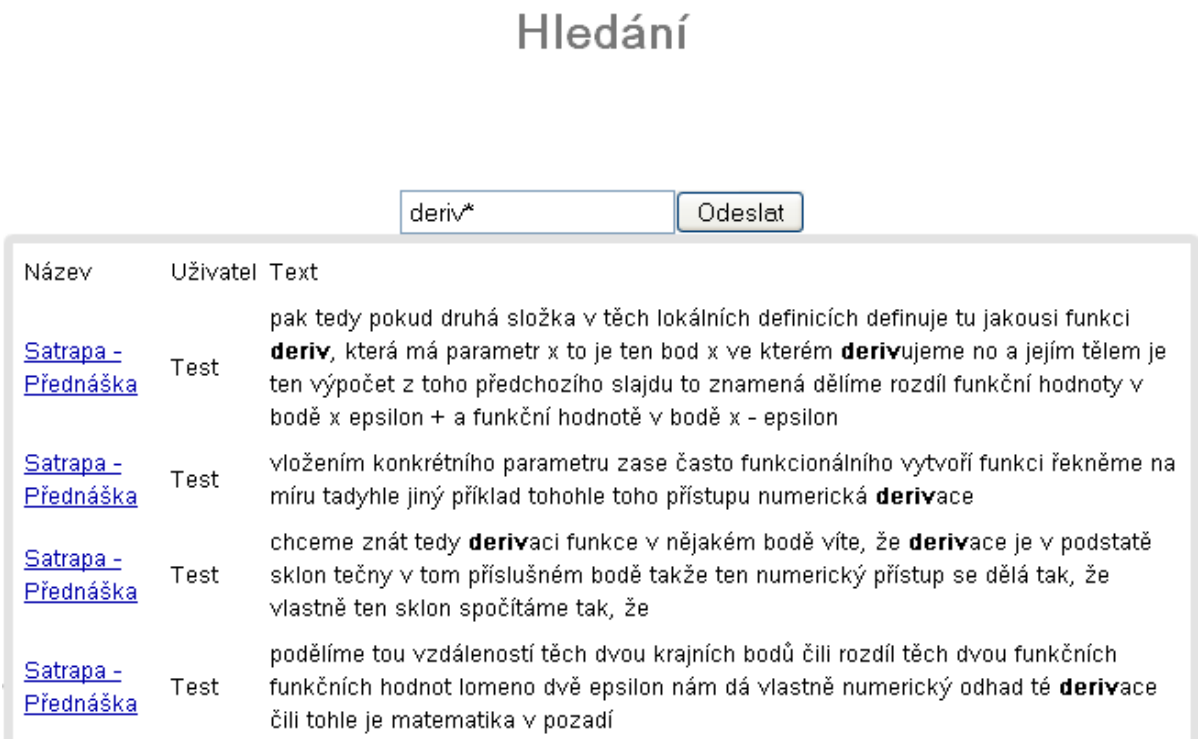
5.11 Vyhledávání

Další funkcí systému je možnost vyhledávat v přepsaných přednáškách. Uživatel jen zadá hledaný výraz a server mu vrátí všechny odstavce ve všech přednáškách, kde se hledaný výraz nachází. Po otevření odstavce začne přehrávání od místa v záznamu, kde byl výraz nalezen.

Pro urychlení vyhledávání (a možnosti využití operátorů v hledaném výrazu) je v databázi držena kopie nejnovější revize každého odstavce v textové podobě bez přítomnosti časových značek. K občerstvení kopie dochází po změně každého odstavce. K uložení této

kopie je využita pomocná tabulka **paragraphstext**, v databázi navíc využívá netransakčního enginu MyISAM namísto InnoDB. Výhoda MyISAM spočívá v možnosti využití fulltextového indexu, který je pro vyhledávání vhodný, oproti InnoDB je ale pomalejší na větší objemy dat. Způsobem indexace přepisů s vyšší efektivitou se ve své práci „Indexace a prohledávání multimédií“ zabývá Bc. Karel Blavka. MySQL přestane být u většího počtu záznamů použitelné, pro potřeby této práce je ale dostačující. Způsob implementace vyhledávání umožňuje snadno přejít z realizace pomocí fulltextového indexu MySQL na jiné řešení. Případně celé hledání může zajišťovat jiný server, který bude zásobován texty z přepisů.

Po nalezení příslušných odstavců jsou uvnitř nich zvýrazněny výskyty příslušného hledaného výrazu. Kompletní řešení vystihuje Obr. 7.



Obr. 7 Hledání

5.12 Integrace do jiných systémů

Pro rozšiřování softwaru do budoucna, případně jeho začleňování do internetových aplikací třetích stran, je potřeba mít na paměti tři hlavní body:

1. **Organizace databáze** – jak jsem již zmínil v kapitolách 5.2 a 5.3, je způsob uložení dat a jejich zpracování nezávislý na konkrétní databázové platformě. Ze strany kódu je navíc vrstva získávání dat z databáze oddělena od dalších

funkčností, takže i v případě, kdy je třeba využití zcela jiného systému uložení dat (XML, vlastní textový formát apod.), je nutné neimplementovat jen velmi malou část serverové stránky aplikace. Systém bude využívat objektů POJO k udržení dat, které je zcela nezávislé na jejich uložení.

2. **Organizace pohledů** – je-li třeba využít jen některých pohledů, případně využít pohledy nezávisle (na různých stránkách bez použití menu implementovaného přímo v aplikaci), lze toto vyřešit několika kompilacemi klientského kódu aplikace vždy s použitím jiného počtu pohledů. Na pohledy je pak možné se odkazovat pomocí parametrů stránky stejně jako v případě klasických (statických) HTML stránek.
3. **Vlastní přihlašování** – pokud je při integraci do cizího systému potřeba použít jiný systém přihlašování (LDAP[8], využití jiných prostředníků pro přihlašování), je nutné modifikovat serverovou část aplikace tak, aby správně vygenerovala „logon seed“ (viz kap. 5.1) při každém načtení stránky.

Po zvážení výše uvedených bodů je pak začlenění aplikace velmi jednoduché. Pro příklad uvedu tři hlavní oblasti integrace, ve kterých je vhodné aplikaci použít:

A. **Integrace pro jinou databázi uživatelů** – pokud bude mít zájem aplikaci nasadit organizace s již stávající databází uživatelů, vše, co pro to musí udělat, je částečně upravit způsob získávání uživatelů tak, aby se shodoval se strukturou uložení v již existující databázi. Všechny ostatní vazby zůstanou zachovány.
B. **Integrace celého prostředí** – tento způsob začlenění uvažuje použití jádra aplikace pro tvorbu celého nového webu. Při použití modelu pohledů může kdokoliv, kdo chce aplikaci použít, její funkce rozšířit přidáním dalších pohledů, ty je možné mezi sebou spojovat a vytvořit tak kompletní web.
C. **Integrace pouze přehrávače** – pokud je žádoucí, aby editace a přehrávání bylo odděleno, případně aby více webových aplikací sdílelo stejná data (ale jen některé z nich by data mohly upravovat), je možné libovolnou část aplikace rozdělit. Takto rozdělené pohledy se pak mohou stát součástí jiných stránek. Především by takto bylo možné přidat do stránky pouze přehrávač s danou přednáškou a jejím opraveným přepisem.

[8] LDAP (Lightweight Directory Access Protocol) je definovaný protokol pro ukládání a přístup k datům na adresářovém serveru. Součástí LDAP je autentizace klienta.

5.13 Streaming videa

V kapitole 2.6 jsem popisoval dva přístupy, které lze použít k přenosu videozáznamu a které zároveň aplikace používá. Lze mezi nimi volit změnou jednoho parametru a je možné je dokonce kombinovat. Metoda s použitím serveru Red5 je vhodnější pro přehrávání, kdy nedochází k častým posunům, ale lze očekávat podstatně větší zátěž na server z důvodu většího počtu současně připojených uživatelů. Využití pseudo-streamingu je přínosnější v editačním režimu, kdy se videem často pohybuje a možnost využití vyrovnávací paměti prohlížeče zvyšuje rychlost odezvy na posuny ve videu.

Z tohoto důvodu jsem v serverové části implementoval vlastní pseudo-streaming mechanismus. Pro snížení vytížení serveru a přenosových cest je přenosová rychlost omezena, aby byla o něco málo vyšší než datový tok videa. Během stahování záznamu pak video na klientovi běží a postupně se načte a nedochází k zasekávání videa nedostatečným datovým tokem (pokud je dostatečně rychlé i připojení uživatele). Pro snížení času, kdy přehrávač „bufferuje“ (ukládá data do vyrovnávací paměti potřebná k tomu, aby bylo možné vůbec začít s přehráváním), je navíc prvních několik sekund záznamu odesíláno maximální rychlostí.

Tento streamovací mechanismus je omezen na využití kontejneru videa FLV (Flash video), který je pro potřeby pseudo-streamingu jednodušší na implementaci. V dnešní době už většina videí na internetu začíná být konvertována do kontejneru MP4, který bude možno použít ve standardu HTML5 bez nutnosti instalace Flashového přehrávače a zjednoduší se tím i přenositelnost na mobilní platformy. Nevýhodou formátu MP4 je ale výrazně složitější implementace pseudo-streamingu.

5.14 Vyhodnocovací software

Společně s webovou aplikací jsem vytvořil i vyhodnocovací program sloužící k porovnání zdrojový přepisů a jejich opravených ekvivalentů a následnému výpočtu přesnosti přepisu. Vyhodnocovací software lze použít ve dvou režimech. První režim pracuje nad databází a sleduje první a poslední revizi každého odstavce, kde je postupně aplikována skupina vyhodnocovacích algoritmů. Ve druhém režimu je navíc programu předán TXZ soubor, který je nejprve zarovnán (s využitím třídy `TXZParser` viz kap. 5.7) k opraveným odstavcům z databáze a následně jsou na zarovnané dvojice odstavců aplikovány vyhodnocovací algoritmy. Pro realizaci programu jsem použil mechanismy pro přístup k databázi z původní serverové části webové aplikace.

Vyhodnocovací software má opět modulární strukturu a je tak možné zaregistrovat libovolné množství vyhodnocovacích algoritmů, jejichž výsledky jsou nakonec uloženy do souboru. Pro výpočet vyhodnocovacích dat jsem zvolil dva různé algoritmy, které popíšu v následujících dvou podkapitolách.

5.14.1 Vyhodnocení pomocí HTK

HTK obsahuje pro účely vyhodnocení nástroj HResults, podle zdrojových souborů může provádět vyhodnocení buď na základě fonému, nebo celých slov. Přepis a referenční zdrojová data je nutné opět převést do formátu, se kterým HResults pracuje. Podle formátu zdrojových dat se určuje, jaká základní jednotka bude pro porovnání použita, buď foném, nebo celé slovo. Pokud je porovnání založeno na fonémech, je nutné výchozí text převést do fonémového přepisu obdobným způsobem, jaký je popsán v kapitole 5.9.

HResults[3] se následně pokusí zarovnat základní jednotky tak, aby jejich odchylka byla minimální. Výstup zarovnání ukazuje následující příklad (pro lepší čitelnost uvedu jen příklad využívající jako základní jednotku slova):

```
Aligned transcription: 1__data/169.lab vs 1__data/169.rec
LAB: to je  liché číslo takže ten výsledek bude definovaný jako...
REC:    ale které číslo takže ta  výstava  bude definovat  jako...
```

Pro takto zarovnané jsou následně vypočteny počty:

- substitučních chyb (dále budu značit S), základní jednotka ze zdrojového textu je nahrazena za jinou v textu rozpoznaném automaticky
- delecí (vymazání, dále budu značit D), přepisovač vynechá některá ze slov
- insercí (dále budu značit I), dojde k přidání slova do automaticky přepsaného textu

Součet všech základních jednotek označím N. Následně jsou pro všechny odstavce vypočítány dvě charakteristiky, správnost (correctness) (5.1) a přesnost (accuracy) (5.2).

$$\text{Percent Correct} = \frac{N - D - S}{N} \cdot 100\% \tag{5.1}$$

$$\text{Percent Accuracy} = \frac{N - D - S - I}{N} \cdot 100\% \tag{5.2}$$

5.14.2 Vyhodnocení pomocí rozdílů

Jako druhou vyhodnocovací metodu jsem zvolil vyhodnocení editačních rozdílů (v praxi označováno jako diff). Metoda využívá Mayersova rozdílového algoritmu[4], implementovaného knihovnou google-diff-match-patch, která je využívaná pro operace vyžadující synchronizaci textu. Cílem metody je najít nejdelší společnou podsekvenci v textu (nebo obdobných zřetězených datech, původní článek popisuje možnost jejího nasazení i v genetice), nebo ekvivalentně najít minimální přepis, který pomocí vložení (dále budu značit I) a vymazání (dále budu značit D) transformuje původní text na nový. Součet zbylých částí textů, které byly nezměněny (unmodified), označím U.

Pro konkrétní výpočet jsem nepoužil počty vložených, vymazaných a nezměněných částí, ale součet všech znaků uvnitř každé z částí, součty označím $I_\Sigma, D_\Sigma, U_\Sigma$. Následně je možné vypočítat procentuální zastoupení vložených částí (5.3), vymazaných částí (5.4) a nezměněných částí (5.5).

$$\text{Percent Inserted} = \frac{I_\Sigma}{I_\Sigma + D_\Sigma + U_\Sigma} \cdot 100\% \tag{5.3}$$

$$\text{Percent Deleted} = \frac{D_\Sigma}{I_\Sigma + D_\Sigma + U_\Sigma} \cdot 100\% \tag{5.4}$$

$$\text{Percent Unmodified} = \frac{S_\Sigma}{I_\Sigma + D_\Sigma + U_\Sigma} \cdot 100\% \tag{5.5}$$

Můžeme-li převést výchozí text na text nový aplikováním výše uvedených pravidel, lze délku nového textu ve znacích vypočíst jako $I_\Sigma + U_\Sigma$. Pokud automaticky přepsaný text označíme jako výchozí a opravený text jako nový, můžeme vypočíst i procentuální přesnost přepisu.

$$\text{Percent Accuracy} = \frac{U_\Sigma}{I_\Sigma + U_\Sigma} \cdot 100\% \tag{5.6}$$

Takto vypočtené charakteristiky se ale velmi (odchylka cca 4%) blíží porovnání prostřednictvím HTK pomocí fonémů. Po provedení algoritmu dochází k nalezení nezměněných částí textu, které spolu nesouvisejí. Pro získání hodnot více odpovídajících realitě je potřeba změny (vložení a vymazání) vyhladit (provést cleanup). Vyhlazení je

prováděno pomocí slučování změn, které tvoří nesouvisející nezměněné části. Například převedením slova „Petra“ na slovo „Pavel“ vznikne P(+ave)(-etra)(+l), po vyhlazení lze získat přirozenější přepis P(+avel)(-etra). Výše uvedená knihovna podporuje tři typy vyhlazení (sémantické, účinnostní, slučovací). Pouze dvě metody vyhlazení (sémantické, účinnostní) ale vracejí vhodné výsledky. Slučovací vyhlazení není pro tento typ úlohy vhodný, protože neeliminuje velké množství změn. Pro ilustraci provedu jejich porovnání na krátkých příkladech (v příkladech jsou použity reálné texty pořízené z přepisu přednášky Alternativní metody programování):

Zdrojový text: „po přáli za třetí řekněme a pěstování procent#0 Dánsko vypsalo na místo pro CeBIT uznávanou po a pití za vlády s parametrem řekněme 10“

Cílový text: „třeba x na třetí řekněme a hned to teď jsem to děláme z toho ypsilonu, aby se to od sebe lépe poznávalo a hned ji zavoláte s parametrem řekněme 10“

Vložení jsou označena zelenou barvou a vymazání červenou barvou.

1. Bez vyhlazení – výchozí výstup algoritmu

Příklad: „po ptřálieba zx na třetí řekněme a pěshned tování proctenď jsem t#0o Ddělánskme z toho vypsailo nu, aby míse to prod sebe ClépeBIT upoznávanou plo a phned jití za voládyte s parametrem řekněme 10“

2. Sémantické vyhlazení – přepisuje rozdíly tak, aby byly lépe čitelné pro člověka, odstraňuje přitom společné části, které vypadají náhodně

Příklad: „po přáli za třetí řekněme a pěstování procent#0 Dánsktřeba x na třetí řekněme a hned to teď jsem to děláme z toho vypsalo na místo pro CeBIT uznávanou po a pití za vládyilonu, aby se to od sebe lépe poznávalo a hned ji zavoláte s parametrem řekněme 10“

3. Účinnostní vyhlazení – upravuje rozdíly způsobem, aby byly lépe zpracovatelné počítačem a to tak, že odstraňuje krátké bloky vypadající jako nezměněné. Tento způsob má navíc parametr (p) určující, jak moc budou jednotlivé krátké bloky penalizovány. Čím je p nižší, tím více se výsledek podobá formátu bez vyhlazení, a naopak, čím je vyšší, tím více se změny spojují, až nakonec zbudou jen dvě, jedno vymazání a jedno přidání.

Příklad ($p = 2$): „po přáli ztřeba x na třetí řekněme a pěshned tování procent#0 Dánskteď jsem to děláme z toho vypsailo nu, aby míse to prod sebe ClépeBIT upoznávanou plo a phned jití za voládyte s parametrem řekněme“

Příklad ($p = 4$): „po přáli ztřeba x na třetí řekněme a pěstování procent#0 Dánskhned to teď jsem to děláme z toho vypsailo nu, aby míse to pro CeBIT uod sebe lépe poznávanou plo a pitíhned ji za voládyte s parametrem řekněme 10“

Příklad ($p = 16$): „po přáli ztřeba x na třetí řekněme a pěstování procent#0 Dánsko vypsalo na místo pro CeBIT uznávanou po a pití za vládyhned to teď jsem to děláme z toho ypsilonu, aby se to od sebe lépe poznávalo a hned ji zavoláte s parametrem řekněme 10“

6 Vyhodnocení

Pro výpočet úspěšnosti automatického přepisu jsem použil program a algoritmy popsané v kapitole 5.14. Porovnával jsem výsledný opravený text s přepisy zvukového záznamu přednášky vytvořené programem Newton Dictate. Pro přepis jsem používal obecný slovník původně navržený na přepis novinářské řeči, který byl natrénován pomocí velkého množství novinových článků. Slovník tak neobsahoval odborné výrazy používané v přednáškách, případně neobsahoval určitá slovní spojení, nebo tato spojení byla značně podhodnocena (například „binární vyhledávací strom“, „průchod stromem“, „X na druhou“ a další). Další dopad na úspěšnost přepisu pak má navíc změna hlasitosti během projevu, kdy přednášející najednou začne mluvit hlasitěji a dochází ke zkreslení v záznamu, které má za následek chybné rozpoznání vlivem nepřesnosti akustického modelu v daném okamžiku. Protože se jedná o rozpoznávání spontánní řeči, řečník také mění tempo projevu, případně se pozastavuje uprostřed fráze nebo je opakuje. Všechny tyto aspekty vedou ke snížení úspěšnosti.

V následující tabulce (Tabulka 1) jsou uvedeny konkrétní výsledky za použití následujících porovnávacích algoritmů. Výsledky obou algoritmů mezi sebou ale nelze přímo porovnávat, protože každý z nich využívá jiných metod výpočtu:

- Porovnání pomocí HTK se základní jednotkou slovem nebo fonémem. Obě metody HTK používají pro určení přesnosti vzorce (5.2) a pro určení správnosti vzorec (5.1), další procentuální zastoupení jsou vypočtena z poměru vložení, vymazání a substitucí ke všem základním jednotkám. Metoda výpočtu, kterou HTK využívá, je standardní způsob vyhodnocování úspěšnosti přepisu.
- Porovnání pomoci Mayersova algoritmu s použitím žádného, sémantického (sem) nebo účinnostního (Ef, s parametrem 4) vyhlazení. Přesnost je vypočtena pomocí vzorce (5.6), správnost vzorcem (5.5), vložení vzorcem (5.3) a vymazání vzorcem (5.4). Substituce nejsou vypočteny, protože je tato metoda nevyhodnocuje. Tento způsob porovnání jsem do přehledu zařadil z důvodu, že řada substitucí v slovním porovnání HTK obsahuje slova velmi podobná často se lišící v jediném písmenu. Tento postup porovnání tak lépe postihuje úsilí, jaké musí uživatel vynaložit, aby daný text opravil.

Tabulka 1 Výsledky

Metoda	Přesnost [%]	Správnost [%]	Vložení [%]	Vymazání [%]	Substitucí [%]
HTK – Slovo	41,19	45,51	4,62	12,56	41,63
HTK – Foném	67,55	72,36	4,81	12,87	17,77
Mayers	71,84	60,50	23,71	15,79	
Mayers – Sem	54,22	39,76	33,57	26,67	
Mayers – Ef 4	62,77	49,11	29,13	21,76	

Ačkoliv výsledky získané pomocí algoritmů v HTK nevykazují výraznou úspěšnost při porovnání slov, je potřeba vzít v úvahu značnou rozdílnost slov v gramatickém modelu (slovníku) od obsahu přednášek. Pokud pak tyto výsledky srovnáme s výsledky pomocí Mayersova algoritmu s vyhlazením (sémantickým nebo účinnostním), můžeme pozorovat, že i při této úspěšnosti nemusí uživatel opravit tak značnou část textu, jaká by této úspěšnosti odpovídala. Činnost přepisovače a jeho úspěšnost při rozpoznání alespoň podobných slov napomůže uživateli při opravě.

7 Cíle do budoucna

Pro zlepšení efektivity přepisovače bude nutné natrénovat lepší gramatický model pro konkrétní obor a styl přednášení konkrétní osoby. Jako rozšíření aplikace by pak mohl vzniknout modul, který bude slučovat různé gramatické modely (obecná čeština, matematika, medicína, programování, …) a vytvářet je tak na míru konkrétnímu oboru.

Aby aplikace mohla fungovat skutečně nezávisle, bylo by vhodné umístit přepisovač přímo na server a nahradit jím stávající funkce vyžadující HTK. Bylo by pak nejen možné na straně serveru řídit (i paralelní) přepis přednášek (a dalšího zvukového materiálu), ale zároveň by bylo možné použít implementaci „forced alignmentu“ přepisovače, který je rychlejší než HTK. Po zrychlení zarovnání textu by bylo možné jej provádět častěji i v době, kdy uživatel aktuálně edituje odstavec. Informace o zarovnání by se tak přímo promítala během editace a informace o pozici v editovaném textu by byla ještě přesnější.

K zvýšení podpory nových formátů by bylo vhodné implementovat pseudo-streaming i pro enkapsulaci MP4.

Rozšířením klientské části o další funkčnosti jako jsou uživatelská oprávnění (server může sdílet s klientem kód pro jejich kontrolu), komentáře k přednáškám, návrhy pro opravy (pokud uživatel sám nemá oprávnění k jejich provedení), složky pro videa, kontrola pravopisu v reálném čase bez nutnosti ručně odesílat text ke kontrole by dále zvýšily pohodlí uživatelů při práci se systémem. Pokud by měl být systém nasazen v praxi, bude právě klientská část prvním, co každý uživatel uvidí a také to, podle čeho bude systém nejprve posuzovat.

Jakmile bude k dispozici více dat, je možné úspěšnost přepisu otestovat na více nahrávkách z různých oborů zaznamenaných více mluvčími.

8 Závěr

V práci byl představen přístup, jak využít webové technologie společně s moderními hlasovými technologiemi a dále možnost propojení s dalšími funkcemi nabízenými jinými subjekty. Příkladem může být ukázka v přehrávacím pohledu aplikace využívající Google Translate pro překlad přednášky do jiného jazyka a okamžité zobrazení titulků ve videu. Další výhodou je pak možnost indexace takto zaznamenaných a opravených dat a následné podpoře rozvoje E-learningových systémů. V aplikační části je použita řada metod demonstrujících možnosti webových aplikací včetně pomůcek pro opravy online tak, aby byly pro uživatele co nejjednodušší a nejpohodlnější. Zároveň jsou zde ukázány možnosti spolupráce více uživatelů na opravách a správě fungování takového systému s výhledem pro budoucí řešení.

Práce také prezentuje nutnost adaptace slovníků pro konkrétní obor. Ukazuje se, že v spontánní řeči, speciálně pak v případě přednášek, kdy učitel musí často přemýšlet dopředu, se vyskytují problémy, se kterými přepisovač, ani gramatický model pro něj vytvořený, nepočítá. Navíc je možné, že v jedné přednášce bude hovořit více mluvčích, což celý proces ještě ztíží. Pro další rozvoj takovéhoto systému bude tedy nutné se zaměřit na celou řadu aspektů, aby bylo možné ještě zvýšit jeho celkovou efektivitu, jak u přepisu záznamu, tak i klientské části aplikace.

Také jsem se pokusil prezentovat způsob vyhodnocení úspěšnosti přepisů metodou, směřující k lepšímu vystižení počtu zásahů do textu, které musí uživatel provést, aby daný text opravil. I když je slovo vyhodnoceno jako chybné, uživatel stále může použít jeho část a nemusí ho opravovat celé.

Obsah přiloženého CD

- Text diplomové práce ve formátu .PDF

Citace

[1] *Cross-site scripting*, URL: http://en.wikipedia.org/wiki/Cross-site_scripting

[2] NOUZA, J., PSUTKA, J., UHLÍŘ, J.: *Phonetic Alphabet for Speech Recognition of Czech*. In: Radio Engineering, vol. 6, no. 4, Prosinec 1997, p. 16-20

[3] YOUNG, S., EVERMANN, G., GALES, M., HAIN, T., KERSHAW, D., LIU, X., A., MOORE, G., ODELL, J., OLLASON, D., POVEY,D., VALTCHEV, V., WOODLAND, P.: *The HTK book (Revised for HTK Version 3.4)*, Březen 2009

[4] MYERS, E. W.: *An O(ND) Difference Algorithm and Its Variations*. In: Algorithmica, vol. 1, no. 2, 1986, p. 251-266

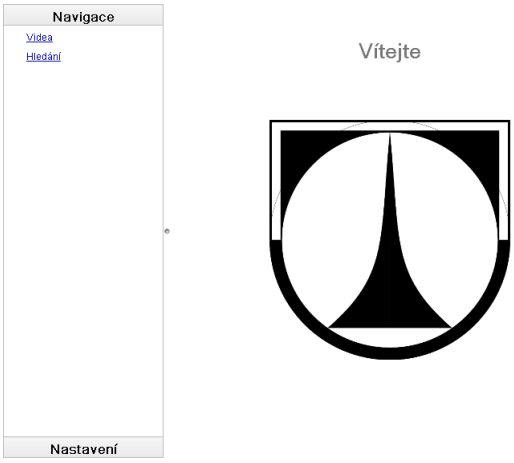
[5] NOUZA, J.: *Využití hlasových technologií v praxi*. V: Uživatelsky přívětivá rozhraní, Horava & Associates, 2009, s. 150-163, ISBN: 978-80-254-5295-0

Přílohy

V příloze je obsažen návod na použití webové aplikace pro opravu přepisů přednášek a také struktura (schéma) souborů v prací použitých.

A Návod k použití

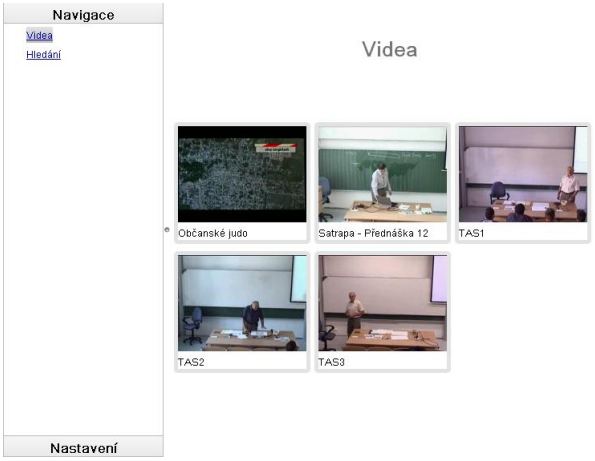
Po přihlášení do systému se zobrazí uvítací obrazovka (Obr. 8). Nalevo je navigační pruh umožňující volit mezi pohledy (stránkami) a napravo je obsah zvoleného pohledu.



Obr. 8 Uvítací obrazovka

A.1 Výběr videa

Po zvolení „Videa“ z navigačního pruhu se zobrazí seznam videí (Obr. 9) v systému zaregistrovaných. Po najetí na nahrávky se zobrazí dodatečné informace (délka, přenášející, apod.). Kliknutím na video je možné přejít do editačního pohledu (viz kap. A.2) nebo ho spustit s titulky v přehrávacím režimu (viz kap. A.3).



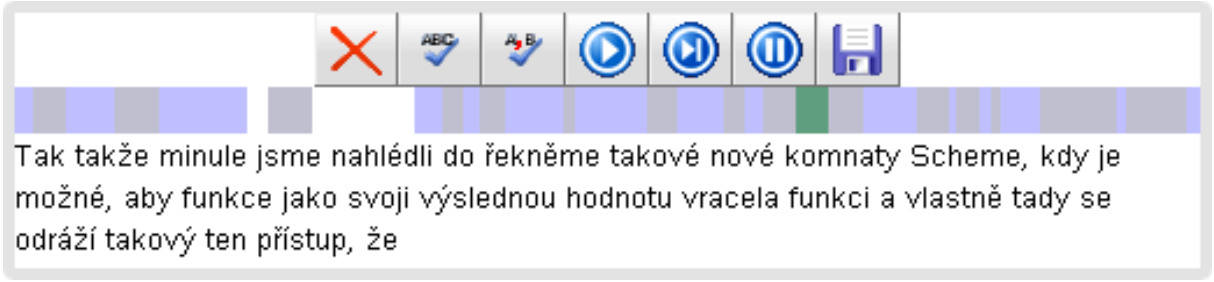
Obr. 9 Výběr videa

A.2 Editace přepisu

Editační pohled (Obr. 10) slouží k opravě přepisů. Když je video přidáno do systému, přepis k němu je možné nahrát z menu „Titulky/Nahrát titulky“. Jakmile jsou titulky nahrány, je možné začít s opravou. Po kliknutí na video se zobrazí vysouvací menu:

- Editovat – přepne zvolený odstavec do režimu editace, pokud byl předtím jiný odstavec editován, jeho editace bude zrušena.
- Rozdělit – pomocí této volby lze titulky rozdělit (rozstřihnout), kliknutím lze vybrat fráze, před kterou bude odstavec rozdělen. Fráze bude zvýrazněna a volbu je nutné ještě potvrdit. Tato operace uživatelsky je **nevratná** (nicméně díky verzování je možné odstavec obnovit do původního stavu přímo v databázi).
- Historie – zobrazí historii všech revizí zvoleného odstavce.
- Skóre – vyznačí v titulku přesnost synchronizace textu s nahrávkou. Je to pouze informativní funkce.

Obr. 10 Editační pohled



Obr. 11 Editace odstavce

Panel editace odstavce je rozdělen na tři části:

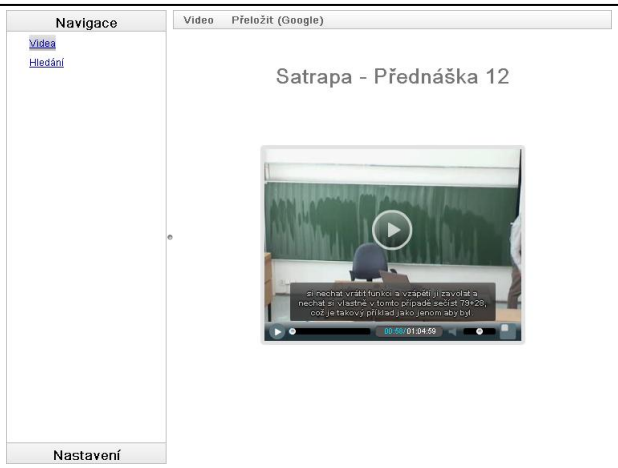
1. Panel nástrojů obsahuje tlačítka k provedení akcí (popsána zleva doprava):
    - Zrušit editaci – zruší editaci odstavce a zahodí veškeré změny
    - Kontrola pravopisu – zkontroluje pravopis a zvýrazní chyby (nebo neznámá slova)
    - Doplnění čárek – doplní do textu čárky před častými spojkami a odstraní chybně zapsané čárky před spojkami, kde by čárky být nemněli. Rozdíly jsou zvýrazněny, aby bylo snazší je rozeznat a překontrolovat.
    - Přehrát odstavec – přehraje video od posledně zvolené pozice nebo od začátku odstavce, pokud nebyla pozice zvolena
    - Zvol pozici – zvolí pozici k přehrávání, od té doby bude po kliknutí na předchozí tlačítko titulek přehrán od této pozice, aby nebylo nutné poslouchat znovu již opravené části
    - Pauza – pozastaví přehrávání
    - Uložit – zašle změny na server, kde jsou uloženy jako nová revize a současně je text sesynchronizován se záznamem
2. Ukazatel pozice – zobrazuje hranice slov v originálním odstavci (před editací), aktuálně zvolené slovo, na kterém stojí kurzor (zeleně) a pozici videa v rámci titulku (zeleným pruhem), pomocí tohoto panelu je možné se snadno orientovat v přepisovaném titulku.
3. Textový editor

Pro ovládání přehrávače je možné použít klávesové zkratky:

- Ctrl + P – Pozastavit/spustit
- Ctrl + [– Skok zpět
- Ctrl +] – Skok vpřed

A.3 Přehrávání přednášek

Po zobrazení videa se otevře „přehrávací pohled“ (Obr. 12). V tomto pohledu je možné sledovat přednášku se zobrazenými titulky přímo ve videu. Hlavní okno navíc obsahuje pruh menu, který umožňuje video zvětšit nebo zmenšit a také titulky přeložit do jiného jazyka pomocí služby Google Translate.



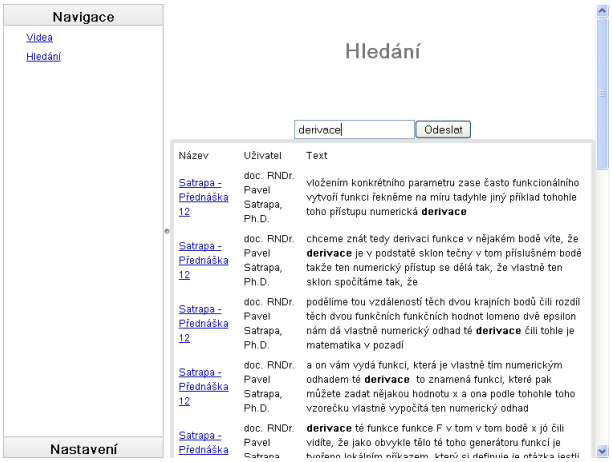
Obr. 12 Přehrávací pohled

A.4 Vyhledávání

Ve vyhledávacím pohledu je možné prohledávat přepsané přednášky. Po nalezení (Obr. 13) příslušného výrazu je možné kliknutím na nadpis videa toto video spustit od odstavce, ve kterém byla fráze nalezena. Text nalezeného je zobrazen vpravo od nadpisu s tučně vyznačenými hledanými výrazy.

Ve vyhledávacím výrazu je možné použít operátory, které je možné dále spojovat pomocí závorek:

- - – ve vyhledaném výsledku nesmí být následující slovo (funkce NOT)
- + – spojuje dvojici slov tak, že ve vyhledaném výsledku musí být obě dvě (funkce AND)
- „bez operátoru" – spojuje dvojici slov tak, že ve vyhledaném výsledku musí být alespoň jedno z nich (funkce OR)
- „uvozovky" – vyhledá konkrétní frázi tak, jak je v uvozovkách zapsána



Obr. 13 Vyhledávací pohled

B Výchozí XML schéma pro strukturu databáze

```
<xs:schema xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

	<xs:element name="Transcription">
		<xs:complexType>
			<xs:sequence>
				<xs:element name="Chapters" type="TypeChapters" />
				<xs:element name="SpeakersDatabase"
type="TypeSpeakersDatabase" />
			</xs:sequence>
		</xs:complexType>
	</xs:element>

	<xs:complexType name="TypeChapters">
		<xs:sequence>
			<xs:element name="Chapter" type="TypeChapter"
maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>

	<xs:complexType name="TypeChapter">
		<xs:sequence>
			<xs:element name="Sections" type="TypeSections" />
		</xs:sequence>
		<xs:attribute name="name" type="xs:string" />
		<xs:attribute name="kanal" type="xs:string" minOccurs="0" />
		<xs:attribute name="porad" type="xs:string" minOccurs="0" />
		<xs:attribute name="href" type="xs:string" minOccurs="0" />
		<xs:attribute name="begin" type="xs:integer" />
		<xs:attribute name="end" type="xs:integer" />
	</xs:complexType>

	<xs:complexType name="TypeSections">
		<xs:sequence>
			<xs:element name="Section" type="TypeSection"
maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>

	<xs:complexType name="TypeSection">
		<xs:sequence>
			<xs:element name="Paragraphs" type="TypeParagraphs" />
		</xs:sequence>
		<xs:attribute name="name" type="xs:string" />
		<xs:attribute name="begin" type="xs:integer" />
		<xs:attribute name="end" type="xs:integer" />
	</xs:complexType>

	<xs:complexType name="TypeParagraphs">
		<xs:sequence>
			<xs:element name="Paragraph" type="TypeParagraph"
maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>

	<xs:complexType name="TypeParagraph">
		<xs:sequence>
			<xs:element name="Phrases" type="TypePhrases" />
			<xs:element name="speakerID" type="xs:integer" />
```

```
            </xs:sequence>
            <xs:attribute name="begin" type="xs:integer" />
            <xs:attribute name="end" type="xs:integer" />
      </xs:complexType>

      <xs:complexType name="TypePhrases">
            <xs:sequence>
                  <xs:element name="Phrases" type="TypePhrase"
maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>

      <xs:complexType name="TypePhrase">
            <xs:sequence>
                  <xs:element name="Text" type="xs:string" />
            </xs:sequence>
            <xs:attribute name="begin" type="xs:integer" />
            <xs:attribute name="end" type="xs:integer" />
      </xs:complexType>

      <xs:complexType name="TypeSpeakersDatabase">
            <xs:sequence>
                  <xs:element name="Speakers" type="TypeSpeakers" />
                  <xs:element name="speakersIndexCounter" type="xs:integer"
/>
            </xs:sequence>
      </xs:complexType>

      <xs:complexType name="TypeSpeakers">
            <xs:sequence>
                  <xs:element name="Speaker" type="TypeSpeaker"
maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>

      <xs:complexType name="TypeSpeaker">
            <xs:sequence>
                  <xs:element name="ID" type="xs:integer" />
                  <xs:element name="FirstName" type="xs:string" />
                  <xs:element name="Surname" type="xs:string" />
                  <xs:element name="Sex" type="xs:string" />
                  <xs:element name="Comment" type="xs:string" />
            </xs:sequence>
      </xs:complexType>

</xs:schema>
```

C Struktura TXZ souboru

```
[0] [100]1. Odstavec, 1. Fráze[200] [250]2. Fráze[400]...[1000]↵
[1100][1200]2. Odstavec, 1. Fráze[1400]...[xxxx]↵
...
```