Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

DIPLOMOVÁ PRÁCE

PLZEŇ, 2024 **Bc. Jan TUPÝ**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2023/2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Jan TUPÝ**
Osobní číslo:	**A22N0114P**
Studijní program:	**N0714A150011 Kybernetika a řídicí technika**
Specializace:	**Umělá inteligence a automatizace**
Téma práce:	**Automatická kontrola výslovnosti**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování

1. Nastudujte problematiku dialogových systémů a automatického rozpoznávání řeči.
2. Navrhněte jednoduchý dialogový systém pro kontrolu výslovnosti promluvy pomocí srovnání grafémového a slovního přepisu s referencí.
3. Dialog realizujte jako jednoduchou webovou aplikaci. Aplikaci otestujte a dosažené výsledky vyhodnoťte.

Rozsah diplomové práce: **40-50 stránek A4**
Rozsah grafických prací:
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí práce.

Vedoucí diplomové práce: **Ing. Luboš Šmídl, Ph.D.**
Katedra kybernetiky

Datum zadání diplomové práce: **2. října 2023**
Termín odevzdání diplomové práce: **20. května 2024**

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Dr. Ing. Vlasta Radová
vedoucí katedry

V Plzni dne 2. října 2023

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 20. května 2024 .....

podpis

Poděkování

Chtěl bych tímto rád poděkovat panu Ing. Luboši Šmídlovi, Ph.D. za odborné vedení, trpělivost a věcné připomínky k této diplomové práci.

Abstrakt

Tato práce se zabývá vývojem aplikace, která automaticky kontroluje výslovnost uživatele. Webová aplikace ukazuje uživateli jednoduché obrázky a jeho úkolem je obrázek správně pojmenovat. Současně může být uživateli zobrazena nápověda či přehráno slovo syntetickým hlasem jako vzor. Vstup od uživatele je automaticky rozpoznán grafémovým rozpoznávačem a pomocí modifikované Levenstheinovy metody srovnán s referenčním textem. Uživateli je poskytnuta okamžitá zpětná vazba o správnosti a preciznosti jeho výslovnosti.

Klíčová slova

automatická kontrola výslovnosti, hlasový dialogový systém, grafické uživatelské rozhraní, SpeechCloud, testování ASR, Levenshteinova vzdálenost

Abstract

This thesis focuses on the development of an application designed to automatically evaluate the user's pronunciation. The web-based application presents the user with simple images, and the user's task is to correctly identify and pronounce the depicted item. Additionally, the application may provide hints or play the word using a synthetic voice as a reference pronunciation. The user's input is automatically recognized by a grapheme-based recognizer and compared with a reference text using a modified Levenshtein method. Immediate feedback on the accuracy and precision of the user's pronunciation is provided to the user.

Key words

automatic pronunciation evaluation, voice dialogue system, graphical user interface, SpeechCloud, ASR testing, Levenshtein distance

Obsah

1 Úvod

Komunikace pomocí řeči představuje jeden z nejzákladnějších, nejstarších a nejpřirozenějších způsobů, jak si lidé mezi sebou vzájemně vyměňují informace. Tento způsob komunikace, který byl dlouho výsadou pouze mezi lidmi, začal být v druhé polovině minulého století intenzivně zkoumán i v rámci interakce mezi člověkem a strojem [1]. V dnešní době, ve světě neustále se rozvíjejících technologií, se hranice mezi člověkem a strojem stále více stírá. To, co bylo dříve považováno za nemožné, je dnes běžnou součástí našeho každodenního života. Jednou z takových věcí je právě komunikace mezi člověkem a strojem. Tuto komunikace zajišťují systémy, které jsou označovány jako hlasové dialogové systémy (HDS), jejichž základní charakteristikou je schopnost přijímat a zpracovávat lidskou řeč.

Aplikace HDS mohou v mnoha ohledech zjednodušit a automatizovat úlohy, které by jinak museli lidé vykonávat ručně, což často není praktické ani ekonomicky výhodné. Tyto systémy jsou často nasazovány v situacích, kde jiné formy interakce s uživatelem nejsou možné nebo vhodné. HDS se stávají nezbytnými nástroji v mnoha oblastech, včetně zákaznické podpory, asistivních technologií, vzdělávacích aplikací a lékařské rehabilitace. Tyto systémy jsou zvláště cenné ve chvílích, kdy má člověk plné ruce práce nebo je jeho zrak zaneprázdněn jinými činnostmi. Příkladem může být řízení dopravního prostředku, kdy se řidič musí věnovat řízení, a nemá tak možnost použít ruce ani oči k interakci se systémem. Další příkladem využití HDS je pomoc lidem s omezenými zrakovými schopnostmi. Pro tyto uživatele může být používání vizuálních rozhraní nemožné, a proto jsou hlasové dialogové systémy vhodnou alternativou pro přístup k informacím.

Tématem této práce je využití HDS na automatickou kontrolu výslovnosti. Hlavní motivací této práce je pomoci lidem, kteří ztratili schopnost mluvit, například v důsledku úrazu nebo mozkové příhody, a kteří se během rekonvalescence snaží tuto schopnost znovu obnovit. Pro člověka, který se znovu učí mluvit, je důležitá zpětná vazba o tom, zda je jeho řeč srozumitelná. Místo toho, aby se dotazoval ostatních, zda je jeho výslovnost již na takové úrovni, která je nutná pro dorozumívání, může daná osoba využít aplikaci na kontrolu výslovnosti. Je zde uplatňována myšlenka:

Porozumí-li mi stroj, bude mi rozumět i člověk.

Tento přístup umožňuje individuální trénink bez nutnosti přímé interakce s terapeutem, což může snížit pocit studu a zvýšit pohodlí uživatele během rehabilitace.

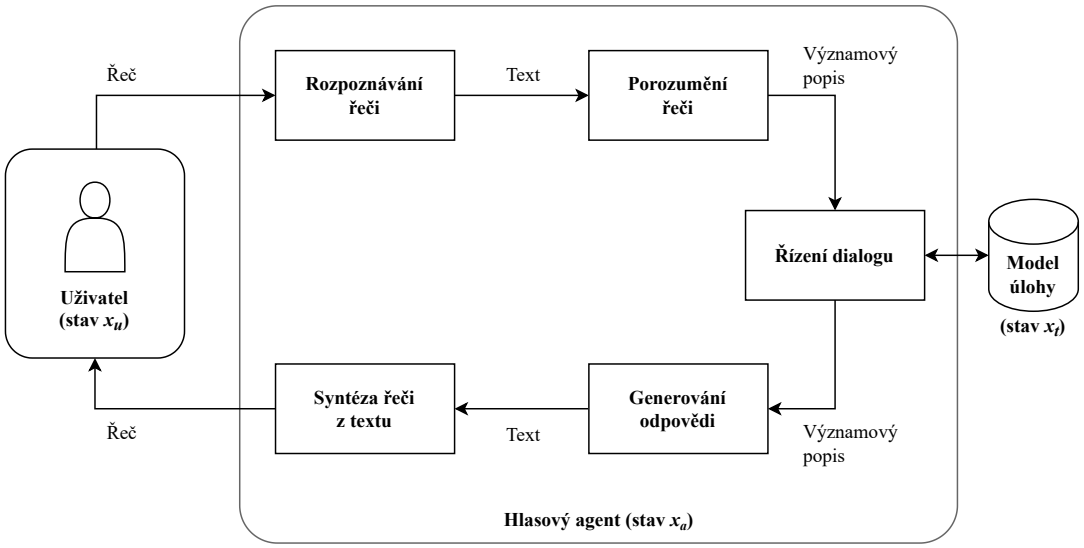
Cílem této práce je vytvořit webovou aplikaci pro automatickou kontrolu výslovnosti řeči. Systém uživateli předkládá postupně sadu obrázků, na kterých je znázorněn nějaký předmět. Úkolem je, aby uživatel vyslovil název předmětu co nejsrozumitelněji. Pokud uživatel neví nebo si není jistý, co obrázek zobrazuje, může využít textovou nápovědu, a pokud má problémy s výslovností, může si nechat slovo přečíst pomocí syntézy řeči. Aplikace využívá grafémovou analýzu pro porovnání vyslovených slov s jejich referenčním textem, čímž poskytuje uživatelům okamžitou zpětnou vazbu na jejich výslovnost. Hlavním cílem je, aby výslovnost uživatelů byla srozumitelná, což se měří pomocí Levenstheinovy vzdálenosti, která ukazuje procentuální shodu.

V teoretické části této práce je nejprve řešena problematika hlasových dialogových systému, kde je větší část věnována automatickému rozpoznávání řeči. Následuje přehled metod porovnávání textových řetězců, kde je pozornost soustředěna hlavně na Levenstheinovu metodu. Poté je poskytnut teoretický základ použitých technologií a je představena platforma SpeechCloud. V praktické části se práce nejprve zaměřuje na testování různých rozpoznávačů řeči, které jsou jádrem pro správné fungování webové aplikace sloužící k automatické kontrole výslovnosti. Nakonec se práce zabývá samotným návrhem aplikace, která je následně realizována a testována.

2 Hlasové dialogové systémy

Hlasové dialogové systémy (HDS) přinášejí nové rozhraní, které lidem umožňuje přirozenou a efektivní výměnu informací mezi uživatelem a strojem pomocí řeči [2]. V hlasových dialogových systémech probíhá hlasový dialog, který má vždy nějaký cíl. V HDS je několik modulů, které jsou potřeba k realizaci hlasového dialogu [3].

Automatické rozpoznávání řeči (ASR) převádí mluvené slovo do textové formy. Tento text je dále za pomoci **porozumění řeči** (SLU) transformován do strojově čitelné reprezentace. **Dialogový manažer** (DM) pak s využitím této reprezentace řídí dialog a zpracovává případné úkoly a generuje významový popis (akci). Následně **generátor odpovědí** (NLG) tuto akci přetváří na text, který je poté předmětem **syntézy řeči** (TTS) [4]. Schéma HDS je zobrazeno na Obrázku 2.1.



Obrázek 2.1: Schéma hlasového dialogového systému (převzato a upraveno) [4]

Stavy hlasového dialogového systému

Celkový stav hlasového dialogového systému (x_{hds}) vznikne složením vnitřních stavů uživatele, hlasového agenta a řešené úlohy. Lze vyjádřit pomocí zápisu:

$$\text{stav HDS} = [\text{stav uživatele, stav agenta, stav úlohy}] \ , \tag{1}$$

nebo symbolicky:

$$x_{hds} = [x_u,\ x_a,\ x_t] \ . \tag{2}$$

Stav HDS obsahuje všechny informace, včetně historie dialogu, které jsou nezbytné pro úspěšné pokračování hlasového dialogu.

Stav uživatele vychází především z cílů uživatele a jeho důvodů pro zahájení interakce. K dalším významným faktorům patří mentální stav uživatele a jeho znalosti a zkušenosti, které umožňují přizpůsobení dialogu individuálním potřebám.

Stav agenta zahrnuje kompletní historii komunikace od začátku hlasového dialogu. Aktualizace tohoto stavu probíhá v modulu řízení dialogu pokaždé, pokud je rozpoznána nová promluva. Historie rozpoznaných promluv umožňuje u některých HDS parametrizovat jednotlivé moduly (ASR, SLU, NLG, TTS).

Stav úlohy představuje aktuálně řešenou úlohu. Vzhledem k její složitosti není úloha přímo začleněna do řízení dialogu, ale existuje jako samostatný model. Tímto způsobem lze úlohu sledovat pouze nepřímo, skrze její řízení a výstupy.

2.1 Automatické rozpoznání řeči

Automatické rozpoznávání řeči převádí řečový signál na text [5]. Tento proces je náročný kvůli velké variabilitě v řečovém signálu. Každý řečník může stejnou promluvu vyslovit odlišně, a dokonce i když ten samý řečník opakuje stejnou promluvu, bude ji vyslovovat pokaždé jinak. Navíc řečový signál ovlivňují jakékoliv změny v prostředí (rušivé zvuky, akustika místnosti) stejně jako změny v přenosovém kanálu (telefonní hovor, změna mikrofonu) [6].

Na počátku vývoje technologií pro automatické rozpoznávání řeči bylo rozpoznávání omezeno pouze na **izolovaná slova**. Pro jednoduché rozpoznávání izolovaných slov se většinou používaly metody známé jako *srovnání se vzorem* [7]. Pro každé slovo byl v databázi uložen referenční vzor a vstupní signál byl porovnáván se všemi referenčními vzory, přičemž největší podobnost určovala výsledný výběr slova. S nárůstem potřeby rozpoznávat **souvislou řeč**, se začaly uplatňovat **statistické modely** [8]. Tyto modely umožnily zpracovávat větší a složitější datové sady, což přispělo k vývoji pokročilých technik rozpoznávání.

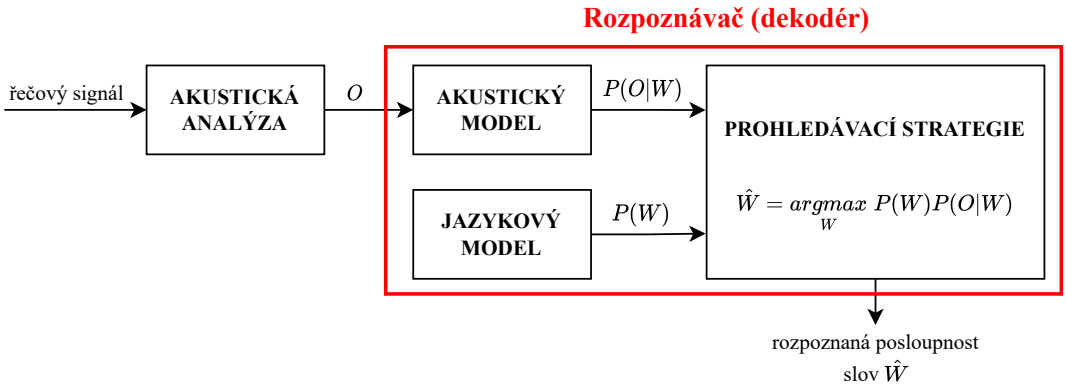
Postupem času byly statistické modely nahrazeny **neuronovými sítěmi**, které dále zlepšily přesnost rozpoznávání řeči. V současnosti se nejvíce využívají modely typu *transformer* [9]. Tyto modely významně přispěly k pokroku v oblasti díky své schopnosti zpracovávat sekvenční data s vysokou úrovní porozumění kontextu. Nejnovější a z pohledu současného výzkumu nejperspektivnější model založený na transformer architektuře je **wav2vec 2.0** [10]. Statistické modely a různé typy neuronových sítí, včetně transformeru, jsou dále popisovány v následující části textu.

2.1.1 Parametrizace

Parametrizace řeči slouží k extrakci podstatných informací z řečového signálu, které jsou zásadní pro jeho rozpoznání, neboli snaha zbavit všech nadbytečné informací z řečového signálu. Díky principu stacionarity je možné signál zpracovávat po malých úsecích (≈10-30 ms). Pro stejné hlásky, neboli fonémy, pak budou odpovídat shodné parametry. Podle Shannonova teorému vzorkování, musí být rychlost vzorkování minimálně dvojnásobná oproti nejrychleji měnící se frekvenci v signálu. Výsledek tohoto postupu tvoří posloupnost příznakových vektorů O. Mezi nejčastěji používané metody parametrizace patří LPC, PLP a MFCC. [11]

2.1.2 Statistický přístup

Schéma rozpoznávání řeči pomocí tohoto přístupu je znázorněné na Obrázku 2.2. Řečníkova promluva je reprezentována vektorem $W = [w_1, w_2, ..., w_N]$, který představuje původní posloupnost slov. Akustická analýza a metody parametrizace transformují tuto řeč na posloupnost příznakových vektorů $O = [o_1, o_2, ..., o_M]$. Rozpoznávač (dekodér), pracující na statistickém základě, má za úkol identifikovat nejpřesnější odhad $\hat{W}$ původní posloupnosti slov W. Výsledek spočívá ve výběru posloupnosti $\hat{W}$, která pro daný signál představuje nejpravděpodobnější řetězec slov. [6]



Obrázek 2.2: Rozpoznávání řeči - dekodér (převzato a upraveno) [6]

Tento proces lze matematicky popsat následujícím způsobem:

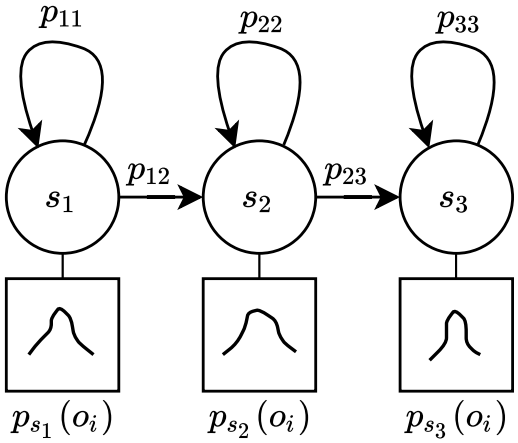
$$\hat{W} = \arg\max_{W}\ P(W|O) = \arg\max_{W}\ \frac{P(O|W)\,P(W)}{P(O)} = \arg\max_{W}\ P(O|W)\,P(W). \quad (3)$$

Pro zjednodušení lze zanedbat pravděpodobnost $P(O)$, která sice ovlivňuje celkovou pravděpodobnost, avšak nemá vliv na výběr maxima. Pro určení ostatních pravděpodobností je nezbytný akustický model - P(O|W) a jazykový model - P(W).

(1) Akustický model

Akustický model obsahuje kromě hlasového traktu, který umožňuje transformaci myšlenek do akustických vln, také technické parametry mikrofonu a akustické vlastnosti okolního prostředí. Z tohoto důvodu může i malá změna v jakémkoli z těchto parametrů akustického kanálu vyžadovat výměnu celého akustického modelu. [6]

Od počátku 80. let minulého století se v modelování začaly využívat *skryté Markovovy modely* (HMM) [12], jež se zaměřují na modelování hlásek (Obrázek 2.3).



Obrázek 2.3: Model hlásky (převzato a upraveno) [6]

Pokud hlasové ústrojí vyslovuje hlásku, může být v jednom ze tří možných stavů (s_1, s_2, s_3), a během toho produkuje zvuky, které jsou transformovány do posloupnosti příznakových vektorů $O = [o_1, o_2, ..., o_i, ..., o_M]$. Pravděpodobnost produkce jednotlivých zvuků v každém stavu $p_{sj}(o_i)$ se liší, protože zvuky nejsou předem jednoznačně definovány. Hlasové ústrojí může zůstat ve svém současném stavu s pravděpodobností p_{jj} nebo přejít do jiného stavu p_{jk}. Slova se modelují spojením modelů hlásek a lze tak vytvořit jakékoli slovo z předem definovaného slovníku, který zahrnuje i fonetickou transkripci slov. Řetězením těchto slovních modelů pak vznikají věty. [6]

Parametry modelu se automaticky upravují podle trénovacích dat, která zahrnují záznamy promluv a jejich co nejpřesnější popisy. V pozdějších fázích vývoje začaly být skryté Markovovy modely (HMM) nahrazovány *hlubokými neuronovými sítěmi* (DNN) [13, 14] nebo jejich kombinacemi HMM-DNN. V dnešní době jsou tyto přístupy již považovány za překonané.

(2) Jazykový model

Jazykový model stanovuje pravděpodobnost $P(W)$, což je pravděpodobnost, s jakou řečník vysloví konkrétní posloupnost slov W. Nastavení parametrů modelu probíhá automaticky s využitím textových databází (korpusů).

Nejvíce přesný jazykový model je založen na **všech předchozích slovech** promluvy. Díky tomu se pravděpodobnost jakéhokoli slova v rámci libovolně dlouhé promluvy vyčísluje na základě všech předchozích slov. Pro posloupnost slov W s počtem slov K lze tuto pravděpodobnost vyjádřit pomocí tzv. *řetězového pravidla* [6]. Matematický zápis tohoto pravidla je následující:

$$\begin{aligned} P(W) &= P(w_1^K) = P(w_1, w_2, \ldots, w_K) \ , \\ &= P(w_1)P(w_2|w_1^1)P(w_3|w_1^2) \ \ldots \ P(w_K|w_1^{K-1}) \ , \\ &= \prod_{i=1}^{K} P(w_i|w_1^{i-1}) \ . \end{aligned} \tag{4}$$

Přestože by bylo ideální zahrnout v jazykovém modelu všechna předchozí slova, takový model by obsahoval extrémně mnoho parametrů, které je velmi složité automaticky určit z trénovacích dat. Proto se v praxi obvykle používá zjednodušený model, tzv. **N-gramový model**, který předpokládá, že pravděpodobnost slova závisí pouze na předchozích $n-1$ slovech. Tento model aproximuje podmíněnou pravděpodobnost slova w_i z původní rovnice (4) pouze na závislost $n-1$ předchozích slov. Matematický zápis tohoto modelu je ve tvaru:

$$P(w_i|w_1^{i-1}) \approx P(w_i|w_{i-n+1}^{i-1}) \ . \tag{5}$$

Pravděpodobnost celé posloupnosti slov W je možné aproximovat do tvaru:

$$P(W) = P(w_1^K) \approx \prod_{i=1}^{K} P(w_i|w_{i-n+1}^{i-1}) \ . \tag{6}$$

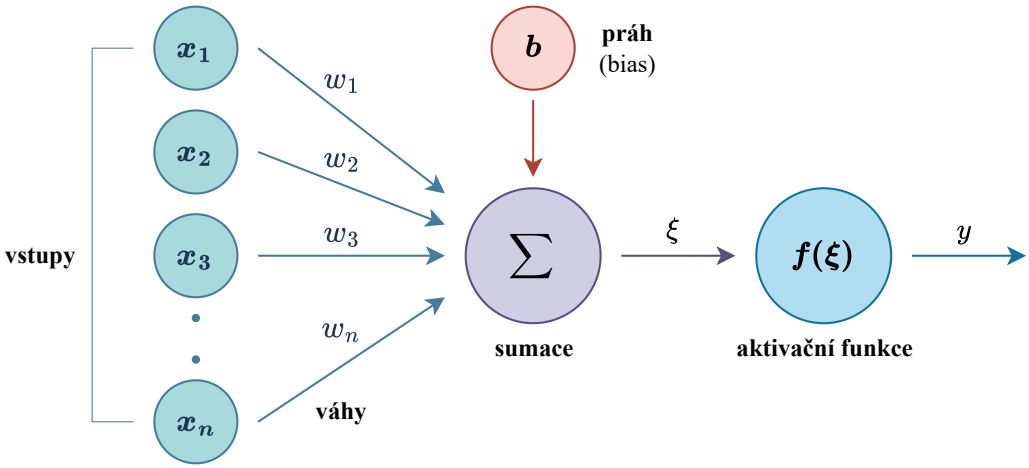
Nejčastěji se pro N-gramový jazykový model používá:

- **unigram** $n = 1$ $\rightarrow$ nezávisí na předchozích slovech $P(w_i)$,
- **bigram** $n = 2$ $\rightarrow$ závisí na jednom předchozím slově $P(w_i|w_{i-1})$,
- **trigram** $n = 3$ $\rightarrow$ závisí na dvou předchozích slovech $P(w_i|w_{i-2}, w_{i-1})$.

Kromě N-gramového modelu existují i další způsoby trénování. Pokud nejsou k dispozici žádná trénovací data nebo pro řešení úlohy stačí malý slovník, je možné využít přístup založený tzv. *formálních gramatikách*. Pro situace, kdy je dostupná velká množina trénovacích dat, jsou používány výhradně *neuronové sítě*. [6]

2.1.3 Neuronové sítě

V oblasti automatického rozpoznávání řeči (ASR) došlo v posledních dekádách k významnému vývoji, zejména díky aplikaci různých typů neuronových sítí. Neuronové sítě fungují na principu modelování interakcí mezi umělými neurony ve struktuře inspirované biologickým mozkem. Základní stavební jednotkou neuronových sítí je perceptron, což je jednoduchý lineární klasifikátor, který přijímá vstupy, váží je pomocí přidružených vah, aplikuje na ně součet (lineární kombinaci) a nakonec tuto sumu transformuje pomocí aktivační funkce do výstupu [15]. Model perceptronu je znázorněn na následujícím Obrázku 2.4.



Obrázek 2.4: Model neuronu - perceptron (převzato a upraveno) [15]

Model neuronu je reprezentován matematickým vzorcem [15]:

$$y = f(\xi) = f\left(\sum_{i=1}^{n} x_i w_i + b\right), \tag{7}$$

kde x_i jsou vstupní proměnné, w_i jsou váhy přiřazené každému vstupu, které se využívají k posílení nebo oslabení vlivu odpovídajících vstupních proměnných, b je práh (bias), což je konstantní hodnota přidaná k výslednému součtu vážených vstupů a ξ je aktivační hodnota, která vstupuje do aktivační funkce f, která transformuje výsledek lineární kombinace do výstupu perceptronu y. [15]

Aktivační funkce [16] představují důležitý prvek v architektuře neuronových sítí, neboť umožňují modelům začlenění nelinearit do učícího procesu, což je zásadní pro řešení složitějších úkolů. Nejčastěji používané aktivační funkce zahrnují sigmoidální funkci (sigmoid), hyperbolický tangens (tanh), ReLU (Rectified Linear Unit) a softmax. [15]

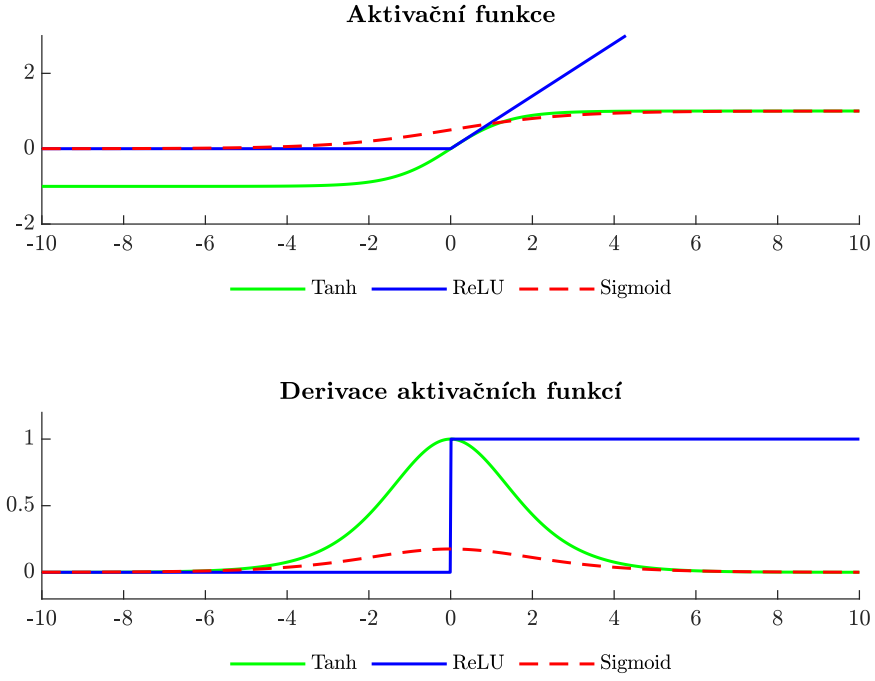
Sigmoida a hyperbolický tangens byly tradičně využívány pro jejich schopnost mapovat vstupy na hladce se měnící výstupy, které se pohybují mezi 0 a 1, respektive -1 a 1. Avšak, obě tyto funkce mohou vést k problému zvanému „problém mizejícího gradientu“ (*vanishing gradient*), kde gradienty potřebné pro aktualizaci vah během učení se postupně snižují na velmi malé hodnoty, což zpomaluje učení. Naproti tomu, ReLU, která je charakteristická svou schopností udržovat gradienty v relativně stabilním rozsahu během tréninku, se stala populární díky své efektivitě a jednoduchosti. [15]

Softmax je funkce, která se nejčastěji používá na výstupních vrstvách neuronových sítí pro úkoly klasifikace, kde zajišťuje výpočet pseudo-pravděpodobností pro jednotlivé třídy [15]. Vzorce zmíněných aktivačních funkcí jsou prezentovány s parametrem λ, který určuje strmost těchto funkcí:

$$\textbf{Sigmoid: } f(\xi) = \frac{1}{1+e^{-\lambda\xi}} \qquad \textbf{Tanh: } f(\xi) = \frac{e^{\lambda\xi}-e^{-\lambda\xi}}{e^{\lambda\xi}+e^{-\lambda\xi}}$$

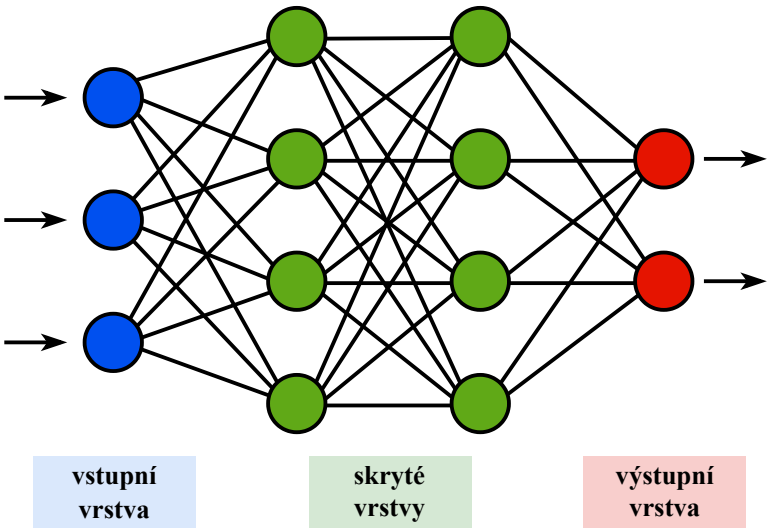
$$\textbf{ReLU: } f(\xi) = \max(0, \lambda\xi) \qquad \textbf{Softmax: } f(\xi) = \frac{e^{\xi_i}}{\sum_j e^{\xi_j}}$$

Na následujícím Obrázku 2.5 jsou zobrazené vybrané aktivační funkce a jejich derivace s parametrem strmosti $\lambda = 0.7$.



Obrázek 2.5: Aktivační funkce a jejich derivace (λ= 0.7)

Perceptron slouží jako základní lineární klasifikátor, který dokáže třídit data s lineárně oddělitelnými vztahy. Nicméně, pro řešení složitějších úloh, kde se v datech vyskytují nelineární vztahy, je nutné použít více perceptronů v uspořádání, které tvoří vrstvy. Takové uspořádání, kde modely neuronů jsou skládány do vrstev, tvoří základ pro neuronové sítě. Nejjednodušším typem takové sítě je dopředná neuronová síť, kde signály postupují pouze jedním směrem ze vstupu sítě na její výstup. Dopředné neuronové sítě jsou typicky strukturovány do tří základních typů vrstev: vstupní vrstva, která přijímá data, jedna nebo více skrytých vrstev, které zpracovávají data, a výstupní vrstva, která produkuje konečný výstup sítě. Tyto vrstvy jsou vzájemně propojeny, což umožňuje komplexní zpracování signálů. Na Obrázku 2.6 je znázorněné schéma dopředné neuronové sítě. [15]

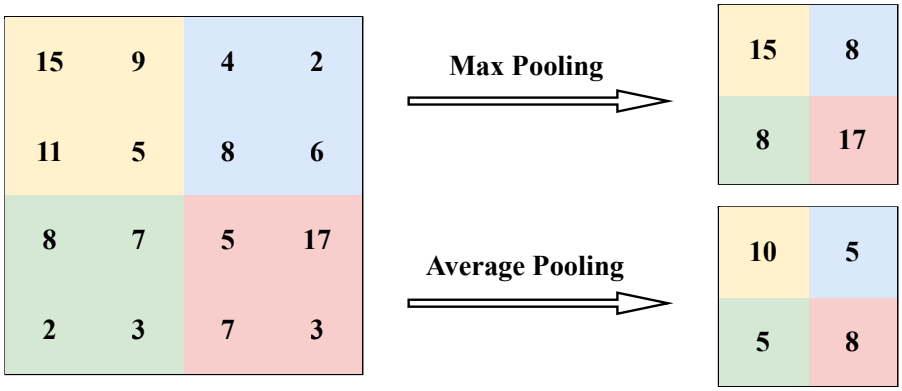


Obrázek 2.6: Schéma dopředné neuronové sítě (převzato a upraveno) [17]

V oblasti ASR se v počátcích používaly jednoduché dopředné neuronové sítě (FNN), které se zaměřovaly na mapování vstupních akustických rysů na výstupní fonetické symboly. FNN modely, které se skládají z jedné nebo více skrytých vrstev, jsou schopné se učit reprezentace dat pomocí váh spojujících jednotlivé neurony, avšak jejich hlavní omezení spočívá v ignorování časového kontextu zvukových signálů. [18]

Dalším významným krokem v evoluci neuronových sítí pro ASR bylo zavedení architektury Time Delay Neural Network (TDNN) [19]. TDNN představuje klíčový průlom v modelování časových závislostí, jelikož je navržené tak, aby explicitně zachytávalo kontextuální informace z různých časových úseků vstupu. TDNN toho dosahuje prostřednictvím sekvencí zpožděných neuronových vrstev, které umožňují efektivnější zpracování dynamických změn v řečovém signálu.

S cílem zlepšit zachycení časových závislostí v akustických sekvencích byly implementovány konvoluční neuronové sítě (CNN) [20], které se vyznačují použitím konvolučních a pooling vrstev. Konvoluční vrstvy efektivně zpracovávají vstupní data s využitím lokálních vážených operací, což vede k zachování prostorových vztahů mezi rysy. Pooling vrstvy následně redukují dimenzionalitu dat, čímž dochází k zefektivnění výpočtů a minimalizaci přetrénování. Dvě běžně používané pooling vrstvy jsou max pooling a average pooling a jsou zobrazené na Obrázku 2.7. [18, 21]



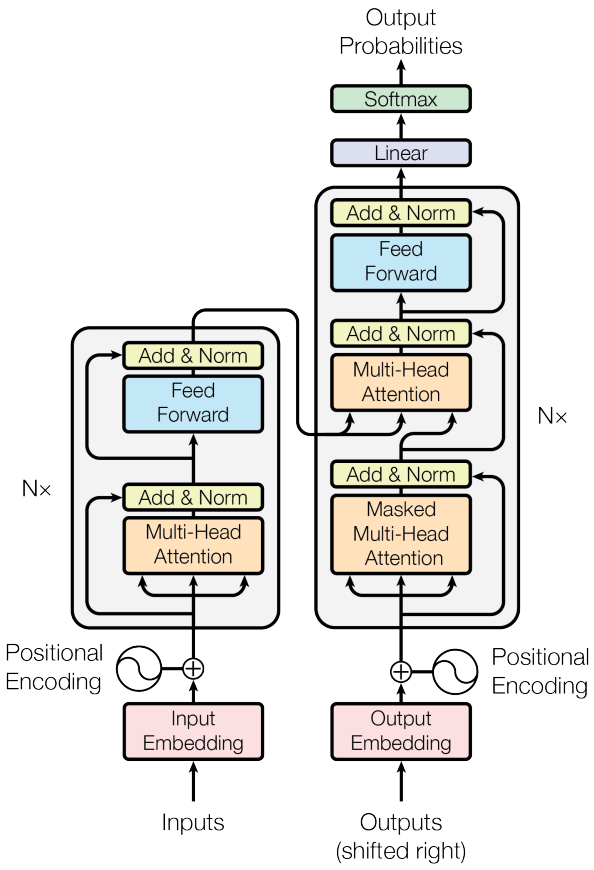
Obrázek 2.7: Max pooling a average pooling (převzato a upraveno) [22]

Max pooling funguje tak, že prochází vstupními daty pomocí okna (filtru) o specifikované velikosti (např. 2x2) a kroku (stride). V každém okně je vybrána maximální hodnota, která je dále předávána do další vrstvy. Max pooling je efektivní v zachytávání výrazných rysů ze vstupních dat, protože preferuje nejsilnější signály a ignoruje méně významné. Na rozdíl od max pooling, average pooling vrstva vypočítává průměr hodnot v každém okně filtru. Tímto způsobem předává průměrnou hodnotu signálu do dalších vrstev. Average pooling je méně citlivý na extrémy ve vstupních datech a poskytuje hladší a rovnoměrnější sumarizaci rysů. [18, 21]

Dalším krokem ve vývoji ASR systémů bylo zavedení rekurentních neuronových sítí (RNN) [23], které umožňují modelování sekvencí dat s proměnlivou délkou tím, že zachovávají stav z předchozích časových kroků. Nicméně, standardní RNN často trpí problémy s tzv. *vanishing gradient* a *exploding gradient*, kde se během trénování gradient buď zmenšuje na tak malou hodnotu, že se síť přestane učit, nebo naopak naroste do extrémních hodnot, což způsobuje nestabilitu ve výpočtech. Pro řešení těchto problémů byly speciálně navrženy modely Long Short-Term Memory (LSTM) [24] a Gated Recurrent Unit (GRU) [25]. Tyto modely zavádějí různé typy regulačních bran (gates), které pomáhají udržet gradienty v kontrolních mezích během tréninku a zároveň umožňují zachování dlouhodobých závislostí v datech. [18]

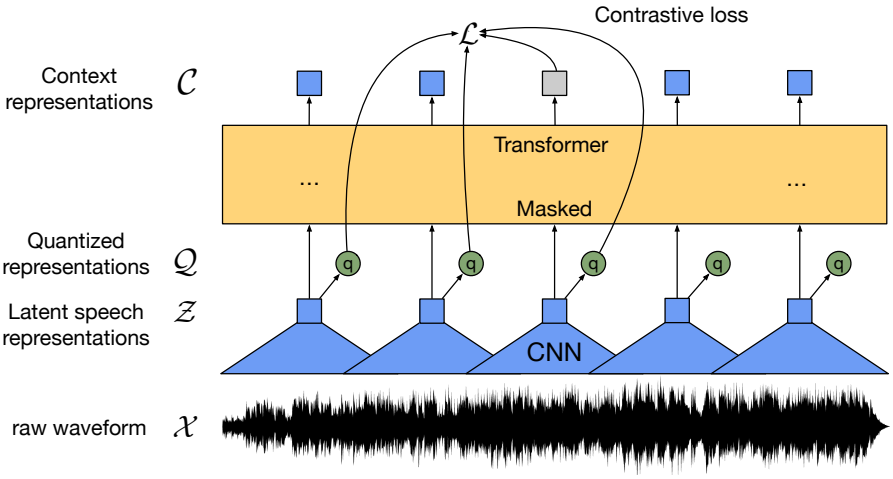
Nejnovějším průlomem v oblasti ASR je využití *self-attention* mechanismů a *transformer* modelů, které byly poprvé představeny ve studii „*Attention Is All You Need*“ [9]. Oproti rekurentním sítím umožňují tyto modely paralelizaci díky *self-attention* vrstvě. V této vrstvě se pro každý časový index nezávisle a paralelně počítá, jaký význam (*attention*) pro něj mají okolní časové indexy. Výstup vrstvy je vážený průměr, kde váhy jsou určeny na základě míry pozornosti (*attention*). Transformer modely používají *multi-head attention* mechanismy, což jsou paralelně pracující *self-attention* vrstvy, které modelují vzájemné vztahy mezi všemi pozicemi ve vstupní sekvenci bez potřeby sekvenčního zpracování. To vede k významnému zlepšení schopnosti modelu zpracovávat dlouhé závislosti. [18]

Transformer architektura je založena na *encode-decoder* struktuře [26]. Encoder převádí vstupní tokeny na posloupnost vektorů. Decoder poté auto-regresně generuje výstupní tokeny, přičemž zároveň zohledňuje encodovanou vstupní posloupnost pomocí tzv. *cross-attention*. Základním stavebním kamenem je *transformer block*, který je opakován N-krát. Na Obrázku 2.8 je zobrazena architektura transformeru. [18]



Obrázek 2.8: Transformer - architektura modelu [9]

Specifickým příkladem aplikace neuronových sítí v ASR je model **wav2vec 2.0** [10]. Tento model je navržen pro převod zvukových vln na kontextové reprezentace, nikoli přímo na text. Převod na text probíhá skrze proces *fine-tuning*. Během této fáze je výstup z modelu spojen s vrstvou, která používá Connectionist Temporal Classification (CTC) [27] pro generování textu. Na Obrázku 2.9 je schéma wav2vec 2.0.



Obrázek 2.9: Schéma wav2vec 2.0 [10]

Wav2vec 2.0 využívá metodu **předtrénování** (*pretraining*), kde model zpracovává velké množství neoznačeného audio materiálu. Během tohoto procesu model naslouchá a analyzuje zvukové vlny, aby identifikoval a zakódoval podstatné charakteristiky řeči. Tento krok umožňuje modelu vytvořit informačně bohaté vektory, efektivně zachycující různé aspekty zvukového signálu. Model implementuje architekturu podobnou transformeru, která využívá sekvence zakódovaných audio vektorů pro generování kontextualizovaných reprezentací. Tento krok umožňuje modelu rozpoznávat vzory a závislosti v řeči napříč různými časovými úrovněmi. [18]

Po předtrénování je model **doladěn** (*fine-tunning*) na menším množství označených dat. Využívá se metoda CTC, která umožňuje modelu mapovat sekvence vstupních audio dat na sekvence výstupních grafémů (písmenek) bez potřeby přesného zarovnání. CTC je vhodná pro modelování situací, kde je délka vstupů a výstupů různá a není předem známo, které zvuky přesně odpovídají jednotlivým grafémům. [18]

K dosažení ještě lepších výsledků je možné do systému integrovat **jazykový model** (LM - Language Model). Tento model pracuje s pravděpodobnostmi následujících slov nebo frází a pomáhá dále zlepšit přesnost a srozumitelnost generovaného textu tím, že řeší nejednoznačnosti a zvyšuje kontextové porozumění. [18]

2.2 Porozumění řeči

Porozumění řeči je modul v hlasových dialogových systémech, který stojí mezi rozpoznáním řeči a dialogovým manažerem. Jeho úkolem je interpretovat, co uživatel řekl, a převést jeho promluvu na významový popis. Tento popis je nezbytný pro realizaci požadované akce. Pro zlepšení přesnosti interpretace je možné využít více alternativních hypotéz rozpoznávání řeči z tzv. *n-best mřížky*, což umožňuje vytvoření více alternativních významových variant. K převodu lexikální podoby na strojovou reprezentaci se využívají tyto přístupy dle [28]:

- znalostní přístup,
- statistický přístup.

2.2.1 Znalostní přístup

Znalostní přístup se opírá o bezkontextovou gramatiku [29] a syntaktickou analýzu (*parsing*). Je to proces, který určuje, zda slovo spadá do jazyka generované gramatikou. Tato gramatika používá syntaktické kategorie (podstatná jména, přídavná jména, slovesa, apod.) a určuje pravidla pro jejich propojení. Tvorba těchto pravidel pro extrakci sémantiky je náročná a vyžaduje zapojení expertů z daného oboru. [28]

2.2.2 Statistický přístup

Statistický přístup k porozumění řeči využívá techniky strojového učení k generování gramatických pravidel, které jsou uchovány ve formě parametrů modelu. Tento proces probíhá na anotovaném korpusu, kde se pro velké množství přepsaných řečových promluv odhadují parametry modelu. Výhody statistického přístupu spočívají ve flexibilitě a škálovatelnosti, jelikož metody odhadování parametrů jsou nezávislé na konkrétním jazyku nebo úloze. [28]

Tento přístup se v posledních letech dále vyvinul s příchodem pokročilých modelů zpracování přirozeného jazyka, jako je BERT (Bidirectional Encoder Representations from Transformers) [30], který využívá dvoufázový tréninkový proces. Během první fáze, známé jako *pre-training*, je model trénován na velkém množství textových dat ve formě bez učitele (*unsupervised*). V této fázi se model učí rozumět jazyku tím, že vyplňuje chybějící slova ve větách a identifikuje, zda spolu dvě věty navzájem souvisejí. Druhá fáze (*fine-tuning*) pak model dotrénuje na specifické úlohy. Tato fáze je trénována s učitelem (*supervised*) a vyžaduje výrazně méně trénovacích dat, protože model již má obecné pochopení jazyka získané během první fáze. [28]

2.3 Řízení dialogu

Za řízení hlasového dialogu je odpovědný **dialogový manažer**. Na základě aktuálního stavu dialogu, jak je popsáno v kapitole 2, vybírá příslušné strategie a akce, které zajistí požadované chování systému. Úlohou dialogového manažera je koordinovat vztahy mezi jednotlivými moduly a zajišťovat komunikaci systému HDS s uživatelem a externími aplikacemi (databáze, řízení robota, apod.). Řízení dialogu zahrnuje různé strategie, způsoby vedení a struktury dialogu. [4, 31]

2.3.1 Vedení dialogu

Nejpřirozenější způsob vedení dialogu se soustředí na řešení nejasností, které mohou vyplývat z chyb uživatele, nedostatků systému nebo problémů s komunikačním kanálem a okolním prostředím. Dialogový manažer tyto nejasnosti může odstranit položením dobře formulovaných dotazů, kterými žádá uživatele o doplňující informace nebo potvrzení toho, co bylo řečeno. Pro potvrzování informací se využívají dva hlavní přístupy dle [31]:

- **Explicitní ověřování**: Uživatel potvrzuje obsah promluvy odpovědí ano/ne.
- **Implicitní ověřování**: Potvrzení obsahu je integrováno přímo do další otázky systému, což dává uživateli příležitost k opakovaným úpravám.

2.3.2 Strategie řízení dialogu

Strategie určuje akci pro následující stav. Podstatné je v každém stavu úlohy vyřešit dílčí cíle dialogu. Tyto dílčí cíle typicky zahrnují některou z následujících úloh [29]:

- **potvrzení** → zjištění správnosti informace,
- **zotavení z chyby** → oprava chyb po nedorozumění,
- **opětovná pobídka** → reakce na neobdrženou informaci,
- **dokončení** → dotazování na chybějící informace,
- **omezení** → omezení rozsahu dotazu,
- **uvolnění** → rozšíření rozsahu dotazu,
- **zjednoznačnění** → vyjasnění nejasných nebo protichůdných vstupů,
- **pozdrav/zakončení** → zjištění začátku a konce konverzace.

Podle iniciativy lze rozdělit strategii řízení na 3 základní typy dle [31]:

- iniciativa systému,
- iniciativa uživatele,
- smíšená iniciativa.

Dialog s **iniciativou systému** znamená, že systém řídí celý dialog, pokládá otázky a nabízí řešení úlohy. Při **iniciativě uživatele** je dialog řízen příkazy uživatele, který přímo ovládá hlasového asistenta a řešené úlohy. **Smíšená iniciativa** představuje kombinaci obou přístupů, kde může v průběhu dialogu iniciativu převzít buď systém nebo uživatel. [31]

Dalším způsobem dělení řízení dialogu je rozlišení podle jeho struktury. První možností je tzv. **turn based dialog**, kde uživatel a hlasový agent střídají promluvy bez možnosti vzájemného přerušení. Hlasový agent reaguje na úplné vstupy uživatele generováním odpovědi. Tento postup, kdy se střídají dotazy a odpovědi, se označuje jako *dialogová obrátka (turn)*. Změny stavu hlasového agenta nastávají v diskrétních okamžicích. Druhou možností je tzv. **inkrementální dialog**, kde hlasový agent zpracovává vstupy uživatele průběžně a při nejasnostech může uživatele okamžitě přerušit (tzv. *barge-in*), což umožňuje dynamickou změnu stavu agenta. [31]

2.4 Generování odpovědi

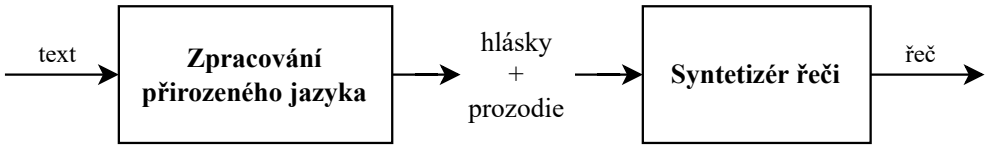
Modul generování odpovědi transformuje významovou reprezentaci z modulu řízení dialogu do textové podoby, která se následně používá pro syntézu řeči. Nejjednodušší metoda generování textu spočívá ve vyplňování předdefinovaných šablon. Tento přístup má však svá omezení, zejména v modifikaci a lokalizaci do jiného jazyka. Pro složitější úkoly generování odpovědí se využívají statistické modely, které jsou trénovány na rozsáhlých textových korpusech pomocí metod strojového učení. [4]

V současné době dominují v oblasti generování odpovědí architektury založené na transformerech. Nejpoužívanějšími modely jsou GPT (Generative Pre-trained Transformer) [32] a T5 (Text-to-Text Transfer Transformer) [33]. Tyto modely představují pokročilé techniky zpracování přirozeného jazyka a umožňují generovat text, který je přirozený a kontextově relevantní. Tyto modely jsou trénovány na obrovských korpusech textových dat a jsou schopné generovat texty, které jsou v mnoha případech těžko odlišitelné od textů psaných člověkem. [18]

Modely jako GPT a T5 nejenže zvládají generovat koherentní a relevantní odpovědi na dotazy, ale také umožňují aplikace v širším rozsahu jazykových úloh, jako je překlad, shrnutí textů a mnoho dalších. Tyto modely představují vrchol současné technologie generování odpovědí a otevírají nové možnosti pro interakci člověka s počítačem.

2.5 Syntéza řeči

Proces syntézy řeči umožňuje vytváření umělé řeči z textu. Systém syntézy řeči (TTS) umožňuje převést jakýkoli textový výstup hlasového agenta na mluvenou řeč. Tento systém se skládá ze dvou hlavních částí: první se věnuje z**pracování přirozeného jazyka** (NLP) a druhou tvoří samotný **syntetizér řeči** [34]. Schéma TTS je na Obrázku 2.10.

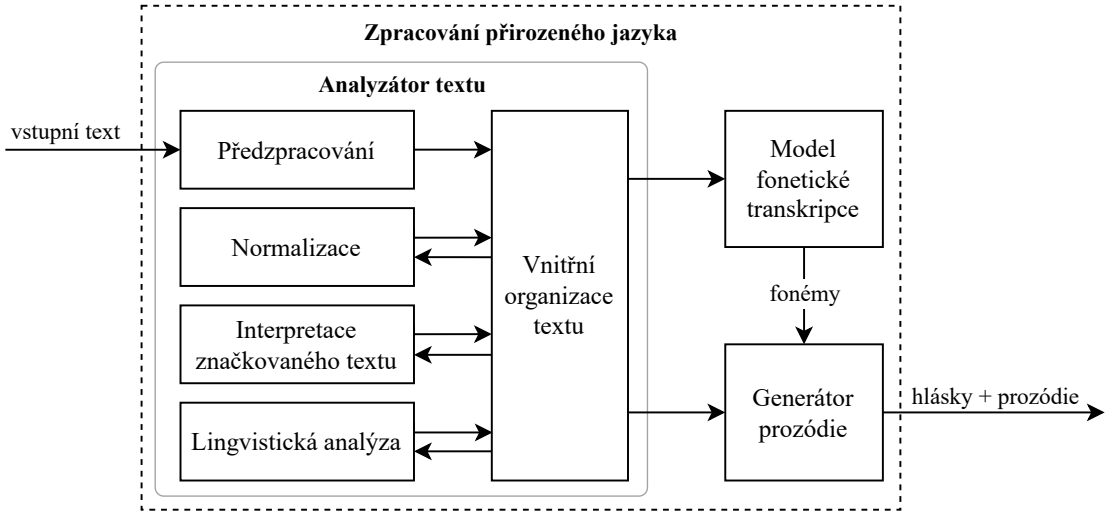


Obrázek 2.10: Schéma TTS systému (převzato a upraveno) [34]

Kritéria pro hodnocení syntézy řeči jsou přirozenost a srozumitelnost. Ideálně by syntetická řeč měla být nerozlišitelná od lidské řeči, což ale nemusí být vždy žádoucí.

2.5.1 Zpracování přirozeného jazyka

Proces zpracování přirozeného jazyka, který převádí text na jeho výslovnostní podobu, je znázorněn na Obrázku 2.11.



Obrázek 2.11: Schéma zpracování přirozeného jazyka (převzato a upraveno) [34]

Proces zpracování textu začíná analýzou, během které dochází k detekci typu textu, odstranění nadbytečných znaků (*formátovací, bílé znaky*) a k analýze struktury textu, známé jako tzv. *tokenizace*. Následně se provádí normalizace, která zahrnuje

přepis číslovek, letopočtů, zkratek a symbolů do plné slovní formy, což se označuje jako tzv. *verbalizace*. Verbalizace vyžaduje zohlednění správného skloňování, což je v morfologicky bohatých jazycích náročnější. Interpretace značkovaného textu pak zahrnuje zvýraznění určitých vlastností, jako jsou emotivní styly (zloba, smutek, radost) nebo expresivní prvky (zakašlání, pauza, nádech). Lingvistická analýza rozkládá text na slovní úseky, zkoumá slovní skladbu a kontext. [34, 35, 36]

Fonetická transkripce převádí předzpracovaný text do fonetické podoby pomocí definovaných fonetických pravidel a slovníků. Nesprávný převod může nastat, pokud pro dané slovo neexistuje příslušné pravidlo a navíc není zahrnuto ani ve slovníku, což se často týká cizích slov, názvů měst nebo jmen osob. Poslední fází zpracování je tvorba prozódie, která zahrnuje určení intonace, rychlosti, hlasitosti, přízvuku, rytmu a členění řeči. Výsledkem je text převedený na výslovnostní formu, zahrnující fonémy a prozódii. [34, 35, 36]

2.5.2 Syntetizér řeči

Syntetizér řeči, který funguje jako software, je základním nástrojem pro umělé generování řeči. Jako vstupní data používá fonetickou informaci o významu řeči a prozodickou informaci o jejím provedení. Tento syntetizér představuje základní komponentu každého TTS systému.

2.5.3 Základní přístupy k syntéze řeči

Základní přístup k syntéze řeči je signálový přístup (*konkatenační syntéza*) a modelový přístup (*statistická parametrická syntéza a syntéza pomocí neuronových sítí*).

Konkatenační syntéza využívá přímo segmenty přirozeného jazyka, známé jako řečové jednotky, které jsou uloženy v inventáři. Tyto jednotky jsou následně řetězeny, aby se vytvořila syntetická řeč imitující charakteristiku řečníka zaznamenaného v inventáři. Nejběžnější technikou konkatenační syntézy je *syntéza výběrem jednotek (unit selection)*. Kvalita této metody závisí na množství a přesnosti zdrojových nahrávek a jejich pečlivé anotaci. Chyby v anotaci a jejich dopady jsou podrobněji rozebrány v článku [37]. Nahrávky pořízené v ideálních akustických podmínkách, jako jsou akustická studia, mohou výrazně zlepšit přirozenost syntetické řeči pro konkrétní hlas a styl mluvy. Výběr odpovídajících jednotek musí pečlivě zohlednit kontext. Nedostatek metody je problém se změnou stylu řeči nebo hlasu. [34, 38]

Statistická parametrická syntéza využívá statistické modely natrénované na řečových signálech ke generování řečových parametrů, které jsou následně přetvořeny na řeč pomocí *vokodéru*. Tato metoda umožňuje dosáhnout uspokojivé kvality syntetické řeči i s menším množstvím nahrávek. Nejprve byly k trénování modelů využívány skryté Markovovy modely (HMM), později je nahradily hluboké neuronové sítě (DNN). I když generovaná řeč může mít nižší akustickou kvalitu, například bzučení nebo přehlazení, tato technika nabízí větší flexibilitu pro změnu hlasu nebo stylu řeči prostřednictvím úprav parametrů modelu. [39, 34]

2.5.4 Neurální syntéza řeči

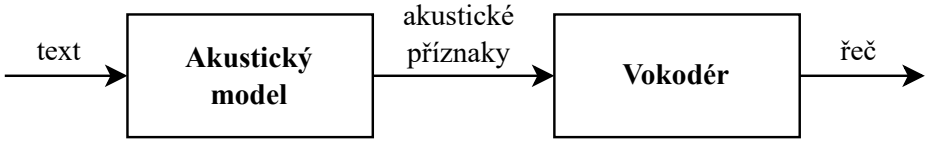
Neurální syntéza řeči využívá modely hlubokých neuronových sítí (DNN). Tento přístup je čistě datový a je založený na strojovém učení. Všechny znalosti potřebné k vytváření řeči jsou uloženy přímo v síti. Aby bylo dosaženo kvalitního výstupu, je nezbytné síť natrénovat na reálných datech, přičemž pro dosažení dostatečné kvality je potřeba minimálně 20 hodin nahrávek. Tento proces je výpočetně náročný. Hlavní výhodou je možnost trénovat síť na datech více řečníků, což poskytuje větší flexibilitu při generované řeči, umožňující například změnu stylu řeči nebo hlasu. [40]

Neurální syntézu řeči lze dělit na dva typy dle [40]:

(a) Dvoufázové neurální modely
(b) End-to-End modely (E2E)

(a) Dvoufázové neurální modely

Dvoufázové neurální modely obsahují akustický model a vokodér. Oba dva modely se trénují samostatně, proto se užívá označení „dvoufázové“. [40]



Obrázek 2.12: Schéma dvoufázového neurálního modelu

Na Obrázku 2.12 je znázorněno schéma dvoufázového trénování, kde do prvního modelu vstupuje text (fonémy, lingvistické příznaky) a výstupem jsou akustické příznaky (mel-spektrogramy), do modelu vokodéru vstupují právě tyto akustické příznaky a výstupem je řečový signál [40].

Systém TTS si lze sestavit z různých akustických modelů a vokodérů, podle potřeby. Přehled vybraných typů modelů včetně příkladu:

Akustické modely

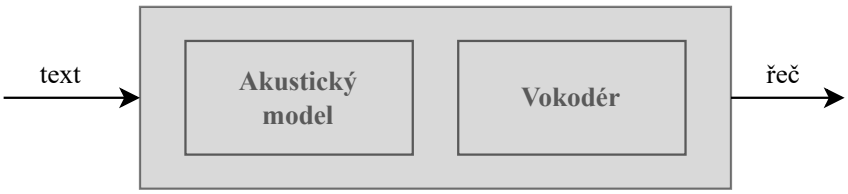
- RNN (př. Tacotron [41])
- CNN (př. DeepVoice [42])
- Transformer (př. FastSpeech [43])

Vokodéry

- Autoregresní (př. WaveNet [44])
- GAN (př. MelGAN [45])
- VAE (př. Wave-VAE [46])

(b) End-to-End modely

End-to-End modely přímo generují řeč z textu (fonémů). Hlavní výhodou tohoto přístupu je přímá optimalizace vzhledem ke konečnému cíli. Tento způsob také redukuje chyby, které mohou vzniknout při dvoufázovém trénování a zároveň nezvýhodňuje mel-spektrogramové reprezentace. Na Obrázku 2.13 je znázorněné schéma E2E systému, který se trénuje v rámci jedné fáze přímo z textu. [40]



Obrázek 2.13: Schéma End-to-End modelu

Příklady End-to-End modelů zahrnují model ClariNet [47], VITS [48] a EATS [49].

2.6 Vyhodnocení

Pro **vyhodnocení celého hlasového HDS** se používá specificky navržený experiment s jasně definovaným cílem dialogu. Jako objektivní metriky hodnocení se používá míra dosažení cíle a počet obrátek (*turn*), které byly k tomuto účelu potřeba. Subjektivní hodnocení zahrnuje kritéria jako přirozenost a plynulost průběhu dialogu. Navíc se provádí vyhodnocení jednotlivých modulů systému dle [4]:

- **ASR** → přesnost rozpoznávání,
- **SLU** → přesnost porozumění,
- **NLG** → ověření správnosti generovaných promluv,
- **TTS** → srozumitelnost a přirozenost generování řeči.

3 Porovnávání textových řetězců

V úvodu kapitoly o porovnávání textových řetězců je vhodné nejprve definovat, co textový řetězec představuje. Textový řetězec může být chápán jako sekvence znaků, která je běžně využívána v různých formách datové komunikace a zpracování informací. V informatice a programování je textový řetězec obvykle reprezentován jako data obsahující slova a věty, které lze ukládat, manipulovat nebo přenášet. [50]

Porovnávání textových řetězců je klíčový proces, který nachází uplatnění v mnoha aplikacích. Hlavním cílem porovnání je určení míry podobnosti mezi dvěma řetězci, což umožňuje identifikaci, zda jsou shodné, nebo jak velké jsou mezi nimi rozdíly. Tato analýza je zvláště užitečná ve vědeckých disciplínách, kde je nezbytné objektivně hodnotit shodu nebo rozdíly mezi textovými údaji. [51]

Pro porovnávání textových řetězců existuje řada metod, každá s unikátními vlastnostmi a specifickým využitím v různých aplikacích. Mezi nejběžnější patří [52]:

1. **Hammingova vzdálenost** - Tato metoda je vhodná pro porovnání řetězců stejné délky a spočívá ve stanovení počtu pozic, na kterých se odpovídající znaky liší.
2. **Levenshteinova vzdálenost** - Měří rozdíly mezi řetězci pomocí minimálního počtu jednoznakových editací (vložení, odstranění, substituce), které jsou potřebné pro převedení jednoho řetězce na druhý.
3. **Damerau-Levenshteinova vzdálenost** - Tato varianta metody Levenshteinovy vzdálenosti navíc zahrnuje transpozici sousedních znaků jako další možnou editaci, což je užitečné například při opravě překlepů v textu.
4. **LCS (nejdelší společná podposloupnost)** - Metoda hledá nejdelší sekvenci znaků, která se vyskytuje v obou řetězcích v tom samém pořadí, ale ne nutně souvisle. Tato metoda je vhodná pro situace, kde jsou změny ve formě vložení nebo odstranění znaků.
5. **Jaro vzdálenost** - Zaměřuje se na porovnání krátkých řetězců s potenciálními transpozicemi znaků. Metoda hodnotí podobnost a umístění shodujících se znaků.
6. **Jaro-Winklerova vzdálenost** - Rozšiřuje Jaro vzdálenost tím, že přidává zvýšenou váhu shodám na začátku řetězců, což je užitečné například v aplikacích pro vyhledávání osob, kde jsou u jména počáteční znaky často stěžejní.

Každá z těchto metod nabízí různé přístupy k měření textové podobnosti a může být aplikována v závislosti na konkrétních požadavcích a kontextu použití. Pro aplikaci, která kontroluje automatickou výslovnost řeči, kde jeden řetězec slouží jako referenční text a druhý řetězec je textový přepis řeči, který může být jinak dlouhý, není potřeba zvýhodňovat počáteční znaky ani provádět transpozici sousedních znaků, je jako nejvhodnější metoda pro vyhodnocení **Levenshteinova vzdálenost**.

3.1 Levenshteinova vzdálenost

Levenshteinova vzdálenost, známá také jako editační vzdálenost, je metoda výpočtu míry rozdílu mezi dvěma textovými řetězci. Byla pojmenována po matematikovi Vladimiru Levenshteinovi, který ji poprvé představil v roce 1965. Tato metoda měří minimální počet jednoznakových operací potřebných k transformaci jednoho řetězce na druhý. Tyto operace zahrnují vložení, odstranění a substituci (nahrazení jednoho znaku jiným). Hodnota vzdálenosti tedy indikuje, kolik úprav je potřeba provést, aby se jeden řetězec transformoval na druhý. [52, 53]

Jednotlivé operace potřebné k převedení jednoho slova na druhé se mohou lišit v závislosti na tom, které slovo je považováno za referenční a které jako testované. Nicméně, celková Levenshteinova vzdálenost, tedy minimální počet jednoznakových operací potřebných k dosažení shody, zůstává stejná bez ohledu na to, které slovo je bráno jako výchozí a které jako testované.

Příklad: Lze uvést příklad, kde budou pozorovány slova „*stole*“ a „*strom*“ ve dvou situacích, vždy s jiným referenčním slovem. Pro obě situace budou uvedeny potřebné operace pro transformaci.

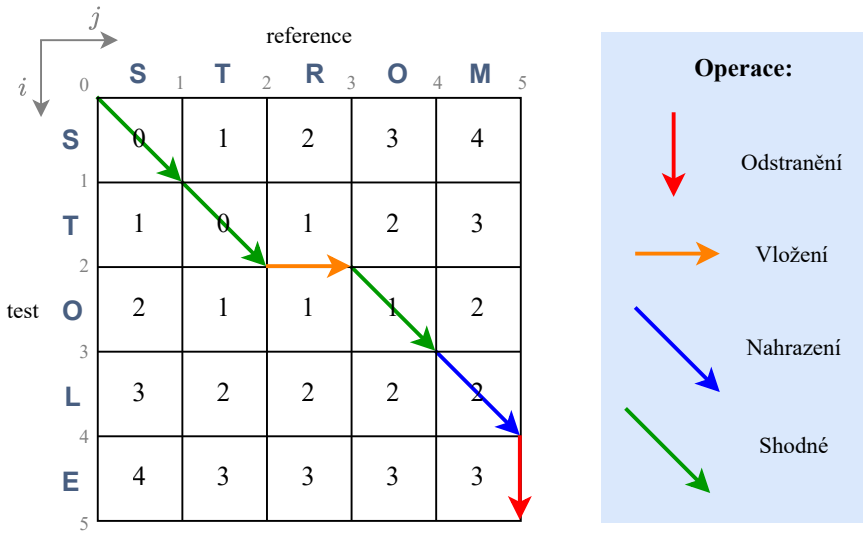
(a) Jednotlivé kroky pro transformaci slova „*stole*“ na referenční slovo „*strom*:“
  1. stole → st**r**ole (vložení „*r*“ před „*o*“)
  2. str**o**le → str**o****m**e (náhrada „*l*“ za „*m*“)
  3. strom**e** → strom_ (odstranění „*e*“ na konci slova)

(b) Jednotlivé kroky pro transformaci slova „*strom*“ na referenční slovo „*stole*:“
  1. st**r**om → st**o**om (náhrada „*r*“ za „*o*“)
  2. sto**o**m → sto**l**m (náhrada „*o*“ za „*l*“)
  3. stol**m** → stol**e** (náhrada „*m*“ za „*e*“)

Příklad jasně demonstruje, že ačkoliv se jednotlivé operace liší, celková Levenshteinova vzdálenost zůstává stejná a je rovná 3.

V praxi je Levensteinova vzdálenost implementována pomocí dynamického programování. Algoritmus vytváří matici, kde jedna osa reprezentuje znaky prvního řetězce a druhá osa znaky druhého řetězce. Hodnoty v matici na pozicích (i, j) odpovídají minimální editační vzdálenosti mezi prvními i znaky prvního řetězce a prvními j znaky druhého řetězce. Hodnoty v matici jsou postupně vyplňovány na základě minimálních nákladů operací potřebných k převedení podřetězců na sebe. Tento výpočetní přístup zajišťuje efektivitu i přesnost ve výsledku. [52]

Na následujícím Obrázku 3.1 je ukázka Levenshteinovy matice vzdáleností z příkladu **(a)**, tedy transformace slova „*stole*“ na referenční slovo „*strom*“. Tato matice je dvourozměrná tabulka, kde každý prvek $d[i][j]$ představuje minimální počet editačních operací potřebných k transformaci prvních i znaků slova „*stole*“ na prvních j znaků druhého slova „*strom*“.



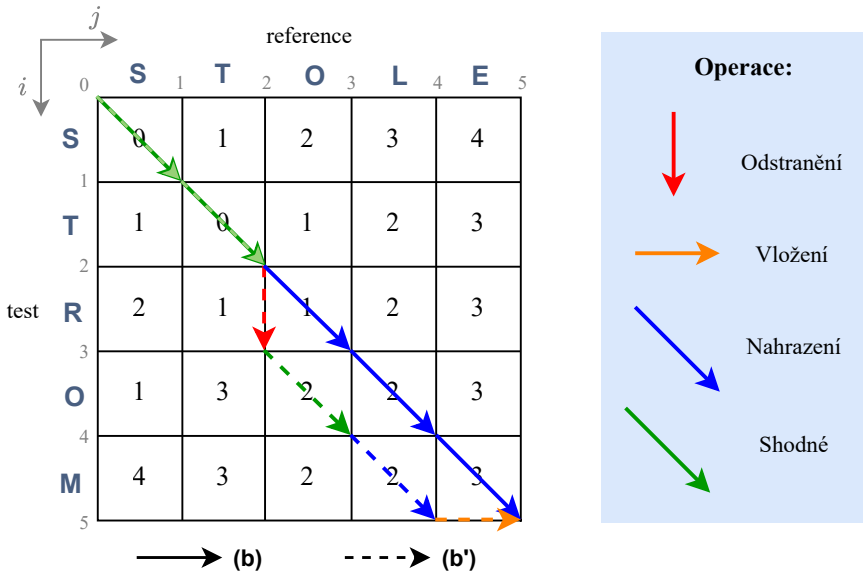
Obrázek 3.1: Ukázka Levenshteinovy matice vzdáleností - varianta **(a)**

Algoritmus Levenshteinovy vzdálenosti vybírá operace s nejmenší vzdáleností v souladu se základním principem minimalizace nákladů. Pokud jsou náklady (tj. celková editační vzdálenost) stejné pro více operací, algoritmus vybere tu, která přijde jako první v kódu algoritmu, když porovnává možné náklady. To znamená, že výsledná posloupnost operací může být ovlivněna konkrétním způsobem implementace, zejména pořadím, v jakém jsou operace v kódu kontrolovány a zapisovány do matice.

V příkladu **(b)** byly uvedeny 3 operace nahrazení pro transformaci slova „*strom*“ na referenční slovo „*stole*“. V tomto příkladu je více posloupností operací, které vedou na nejmenší vzdálenost. Další možnou variantou **(b')** je například inverzní varianta transformace **(a)**, jejíž operace jsou následující:

1. st**r**om → stom (odstranění „*r*“)
2. sto**m** → sto**l** (náhrada „*m*“ za „*l*“)
3. stol → stol**e** (vložení „*e*“ na konec slova)

Na následujícím Obrázku 3.2 je ukázána Levenshteinova matice vzdáleností pro varianty **(b)** a **(b')**. Obě varianty vykazují shodnou minimální cenu, která je rovna 3, ale liší se v posloupnosti operací.



Obrázek 3.2: Ukázka Levenshteinovy matice vzdáleností - varianta **(b)** a **(b')**

V tomto případě jsou všechny operace (vložení, odstranění, substituce) u Levenshteinovy metody standardně ohodnoceny stejnou hodnotou 1. Je však možné jednotlivé operace ohodnotit různými hodnotami nebo dokonce specifikovat různé hodnoty pro konkrétní operace na základě znaků, které jsou těmito operacemi zasaženy.

3.2 Upravená Levenshteinova metoda

Upravená Levenshteinova metoda umožňuje hledat referenční řetězec v libovolném podřetězci testovacího řetězce. Jinými slovy referenční řetězec lze hledat v testovacím řetězci, který může mít libovolný počátek a konec. Hlavní výhodou tohoto přístupu je možnost najít například slovní spojení ve větě a to s minimální cenou. Jako příklad lze uvést referenční slovo „*stříkačka*“, hledané ve větě s překlepem:

„*Pomocí injekční stříkečky se odebírá krev.*“

Nejvíce odpovídající řetězec je „*stříkečky*“, zvýrazněné modře. Jelikož se v nalezeném řetězci nachází jeden překlep („*a*“ → „*e*“), celková vzdálenost je rovna 1.

Pro některé účely by mohlo být vhodnější testovat jednotlivá slova ve větě a vybrat to slovo, které má nejmenší vzdálenost. Avšak v případě, kdy je textový řetězec výsledkem automatického rozpoznávání řeči a může docházet k chybějícím nebo přebývajícím mezerám, není vhodné kontrolovat pouze jednotlivá slova. Jako příklad lze uvést referenční slovo „*stříkačka*“, které je rozpoznáno jako „*stří kočka*“, kde je navíc vložena mezera, která slovo rozděluje na dvě slova a také došlo ke změně písmena „*a*“ na „*o*“. Ukázka možné rozpoznané věty:

„*Pomocí injekční stří kočky se odebírá krev.*“

Celková vzdálenost v nalezeném řetězci je rovna 2. Pokud by se testovala pouze samostatná slova, byla by nejlépe hodnocená možnost slovo „*kočky*“, s jedním překlepem (substituce) a čtyřmi chybějícími písmeny (vložení), což by dalo celkovou vzdálenost rovnu 5. Na tomto demonstrativním příkladě je vidět, že upravená Levenshteinova metoda je efektivnější.

3.3 Levenshteinův poměr

Levenshteinův poměr (Levenshtein Ratio - LR) je metrika využívaná k měření míry podobnosti mezi dvěma textovými řetězci. Na rozdíl od Levenshteinovy vzdálenosti, která počítá pouze minimální počet jednoznakových editačních operací potřebných k transformaci jednoho řetězce na druhý, Levenshteinův poměr poskytuje normalizovanou hodnotu podobnosti mezi řetězci. Existují dva základní způsoby výpočtu Levenshteinova poměru. První je tradiční způsob, jehož výpočet je přímý a snadno pochopitelný, což usnadňuje implementaci a interpretaci výsledků a druhý je součtově-normalizovaný způsob. [54]

Tradiční Levenshteinův poměr

Tato hodnota je vyjádřena jako poměr mezi počtem operací, které je třeba provést, a maximální délkou z porovnávaných řetězců. Vzorec Levenshteinova poměru je:

$$LR_1 = 1 - \frac{levenshtein\_distance(s1, s2)}{max(len(s1), len(s2))} , \tag{8}$$

kde $levenshtein\_distance(s1, s2)$ je Levenshteinova vzdálenost mezi řetězci $s1$ a $s2$ a $max(len(s1), len(s2))$ je maximální délka z těchto dvou řetězců.

Součtově-normalizovaný Levenshteinův poměr

Tato hodnota je vyjádřena jako poměr mezi rozdílem součtu délek obou porovnávaných řetězců a Levenshteinovy vzdálenosti mezi nimi, dělený součtem délek těchto řetězců. Vzorec Levenshteinova poměru je:

$$LR_2 = \frac{sum(len(s1), len(s2)) - levenshtein\_distance(s1, s2)}{levenshtein\_distance(s1, s2)} , \tag{9}$$

kde $levenshtein\_distance(s1, s2)$ je Levenshteinova vzdálenost mezi řetězci $s1$ a $s2$ a $sum(len(s1), len(s2))$ je součet délek obou řetězců.

Tento způsob výpočtu se využívá v některých implementacích Python knihoven, kde je navíc operace substituce ohodnocena cenou 2. Toto ohodnocení je založeno na předpokladu, že substituce je ekvivalentní dvěma po sobě jdoucím operacím: odstranění a následnému vložení, přičemž každá z těchto operací má cenu 1.

Příklad: Lze uvést příklad výpočtu pro slova „*strom*“ a „*stole*“, kde Levenshteinova vzdálenost je 3:

$$LR_1 = 1 - \frac{3}{5} = 0.4 , \tag{10}$$

$$LR_2 = \frac{10 - 3}{10} = 0.7 . \tag{11}$$

Tento výsledek naznačuje, že podle tradičního Levenshteinova poměru (10) je podobnost mezi „*strom*“ a „*stole*“ 40 %, zatímco podle součtově-normalizovaného Levenshteinova poměru (11), který bere v úvahu celkovou délku obou řetězců, je podobnost mezi „*strom*“ a „*stole*“ 70 %.

Pokud by byla v součtově-normalizovaném Levenshteinově poměru operace substituce ohodnocena cenou 2, jak je to běžné v některých Python knihovnách, výsledný poměr by se od původních 70 % lišil.

4 Teoretický základ použitých technologií

V následující části jsou popsány vybrané technologie využívané pro tvorbu uživatelského rozhraní. Bylo vybráno pět technologií: HTML, CSS, DALL·E 3, DOM a WebSocket.

4.1 HTML

HTML (*Hypertext Markup Language*) [55] je základním stavebním kamenem internetu. Jedná se o **hypertextový značkovací jazyk**, který je primárně určen pro tvorbu struktur webových stránek. Jeho původní účel zahrnoval nejen tvorbu a formátování obsahu, ale také jeho stylování. V průběhu času však byla většina stylovacích funkcí převedena na kaskádové styly (CSS), které umožňují širší možnosti v oblastech designu a animací.

HTML dokumenty jsou tvořeny značkami, známé jako **tagy**, které definují a ohraničují jednotlivé prvky na stránce. Tyto prvky mohou zahrnovat odstavce, nadpisy, odkazy, obrázky a mnoho dalších. Každý tag může obsahovat atributy, které poskytují další informace o elementu, například umístění obrázku na webu nebo cílovou URL adresu odkazu.

Struktura HTML dokumentu:

Správná struktura HTML dokumentu zahrnuje několik základních elementů:

- **!DOCTYPE html**: Deklarace typu dokumentu, která prohlížeči definuje, že se jedná o HTML5 dokument.
- **html**: Kořenový element, který obaluje celý obsah webové stránky.
- **head**: Sekce obsahující metadata, jako jsou CSS styly, externí skripty, meta informace a titulek stránky.
- **title**: Element určující titulek webové stránky, který se zobrazuje v záložce prohlížeče.
- **body**: Hlavní obsahová oblast, kde se nachází veškerý viditelný obsah stránky, od textů po obrázky.

4.2 CSS

Kaskádové styly, známé také pod zkratkou CSS *(Cascading Style Sheets)* jsou moderní jazyk navržený pro stylování webových stránek napsaných pomocí značkovacích jazyků jako HTML, XHTML, nebo XML. CSS umožňuje kompletní oddělení designu stránky od jejího obsahu. Princip kaskádování umožňuje, aby se v případě konfliktů více pravidel na stejný prvek uplatnilo pravidlo s vyšší prioritou.

4.3 DALL·E 3

Technologie DALL·E 3 [56] je pokročilý model umělé inteligence pro generování obrázků, vyvinutý společností OpenAI[1]. Tento model je schopen generovat vysoce kvalitní a relevantní vizuální obsah na základě textových popisů. Pro integraci a aplikaci tohoto modelu byla použita služba ChatGPT[2], konkrétně GPT model **image generator**, který umožňuje snadné používání DALL·E 3 pro generování obrázků přímo z webového rozhraní.

4.4 DOM

DOM (*Document Object Model*) je aplikační rozhraní (API) nezávislé na platformě i programovacím jazyku, které umožňuje manipulovat s objekty HTML, XHTML a XML dokumentů. Po načtení stránky prohlížeč vytvoří v paměti její DOM, který má stromovou strukturu a zahrnuje různé typy uzlů, jako jsou elementy, atributy, texty a komentáře. Tuto strukturu lze efektivně měnit například pomocí JavaScriptu, což umožňuje dynamickou interakci s webovou stránkou.

4.5 WebSocket

WebSocket je počítačový protokol, který poskytuje plně duplexní komunikační kanály přes jedno TCP spojení, což umožňuje klientské aplikaci, například webový prohlížeč, a serveru vzájemně komunikovat v reálném čase. Po zahájení spojení může server automaticky posílat data klientovi bez jeho předchozího vyžádání, což umožňuje obousměrný průběh konverzace. Protokol také udržuje trvalé připojení pro okamžité sdílení informací.

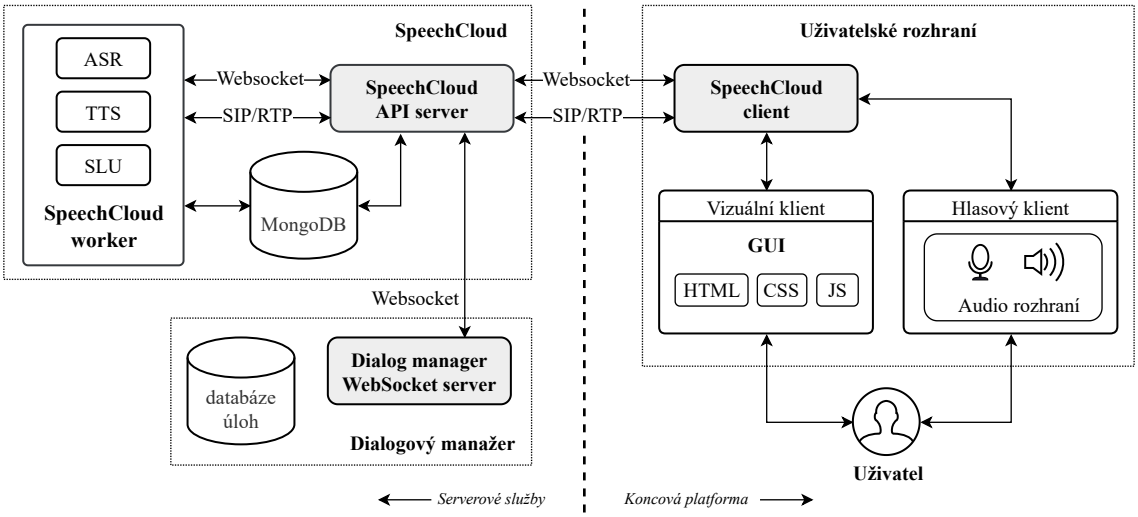
[1] `https://openai.com`
[2] `https://chatgpt.com/gpts`

5 Platforma SpeechCloud

SpeechCloud je platforma, která nabízí snadno přístupné řešení pro využívání řečových technologií napříč různými koncovými zařízeními. Tento systém je výhradně navržen jako interní nástroj, poskytující flexibilní rozhraní pro různé řečové moduly, včetně automatického rozpoznávání řeči (ASR), porozumění mluvené řeči (SLU) a syntézy řeči (TTS). [57]

5.1 Architektura

Architektura SpeechCloudu je navržena tak, aby podporovala multimodální dialogy. Pro každého klienta je pomocí URL adresy vytvořena nová relace (session). Tato architektura je rozdělena do tří základních částí: dialogový manažer, SpeechCloud a uživatelské rozhraní, jejichž struktura je graficky zobrazena na Obrázku 5.1.



Obrázek 5.1: Architektura SpeechCloudu (převzato a upraveno) [57]

5.1.1 Komunikace

Komponenty architektury SpeechCloud jsou integrovány skrze standardizované protokoly. Pro výměnu řídících zpráv je používán Websocket v kombinaci s formátem JSON. Audio signál je přenášen pomocí protokolů SIP/RTP (Session Description Protocol/Real-time Transport Protocol), což jsou standardy pro realizaci VoIP (Voice over Internet Protocol) služeb. [57]

5.2 SpeechCloud API server

Spojení mezi webovými prohlížeči a řečovými moduly v rámci SpeechCloudu je realizováno pomocí protokolů SIP a RTP, které jsou určeny pro přenos zvukových paketů [57].

Pro správu interakcí mezi klienty, řečovými moduly a relacemi, je využívána databáze MongoDB. Tato databáze eviduje všechny události a metody volané během relace, a také ukládá zpracované zvukové záznamy vzniklé v průběhu relace. [57]

5.2.1 Worker

Při zahájení každé relace přidělí SpeechCloud API server konkrétního workera k dané relaci, jehož úkolem je aktivace hlasových modulů (ASR, TTS, SLU) [57].

Výsledky rozpoznávání (ASR) jsou vytvářeny jako události, které lze sledovat pomocí posluchačů událostí (tzv. *lisener*) v dialogovém manažeru (DM) a v JavaScriptu (JS). V případě využití modulu TTS je zvuk přenášen přímo k uživateli. Pro modul SLU, který se spoléhá na algoritmus SED (Semantic Entity Detection), jsou pro definování entit využívány gramatiky formátované dle specifikace SRGS (Speech Recognition Grammar Specification)[3]. [57]

5.3 Dialogový manažer

V rámci platformy SpeechCloud není dialogový manažer začleněn přímo, ale je implementován jako samostatný Websocket server. Tento manažer zajišťuje řízení dialogů, přičemž nastavení dialogu lze upravovat podle požadavků konkrétní koncové platformy. Dialogový manažer využívá asynchronní metody poskytované platformou SpeechCloud a reaguje na příkazy, které jsou aktivovány událostmi generovanými touto platformou. [58]

5.4 Uživatelské rozhraní

Uživatelské rozhraní platformy je implementováno jako webová aplikace s vizuálním i hlasovým klientem, umožňující ovládání pomocí klávesnice, myši nebo hlasu. Tyto klienty propojuje SpeechCloud client, který slouží jako komunikační most mezi webovou aplikací a zbytkem architektury. [57]

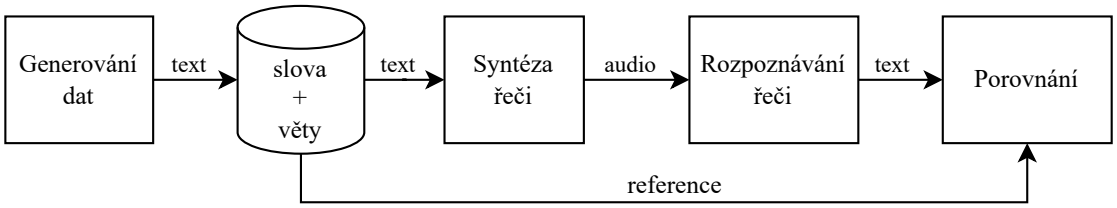
[3] `https://www.w3.org/TR/speech-grammar/`

6 Testování rozpoznávačů řeči a syntézy

Jedním z klíčových prvků pro úspěch webové aplikace zaměřené na automatickou kontrolu výslovnosti je správný výběr rozpoznávače řeči. Tato komponenta je zásadní pro efektivní fungování aplikace, jelikož její hlavní úlohou je s vysokou přesností identifikovat nesprávně vyslovená slova. Cílem testování různých modelů automatického rozpoznávání řeči (ASR) je tedy najít ten nejlepší rozpoznávač, který splní tento hlavní požadavek.

Testování se soustředí na tři rozpoznávače řeči, z nichž každý využívá různé technologie a přístupy k rozpoznávání řeči. Jedním z nich je tradiční rozpoznávač založený na architektuře **TDNN** *(Time Delay Neural Network)* s akustickým modelem, kde dekódování probíhá nad fonémovým výstupem s využitím slovníku a jazykového modelu. Dva další modely jsou založeny na modernější architektuře **wav2vec**, z nichž jeden využívá jazykový model a druhý grafémový přístup bez přidaného jazykového modelu. Právě grafémový přístup se jeví jako nejvhodnější pro zamýšlené využití v této aplikaci, neboť dekóduje řeč přímo na základě grafémů, aniž by na výstupu využíval přidaný jazykový model. Tato charakteristika je důležitá, protože jazykové modely mohou mít tendenci adaptovat i nesprávně vyslovená slova tak, aby se ve výsledku blížila slovům známým z jazykového modelu, což pro aplikaci kontroly výslovnosti může vést k nežádoucím kompromisům v přesnosti rozpoznávání.

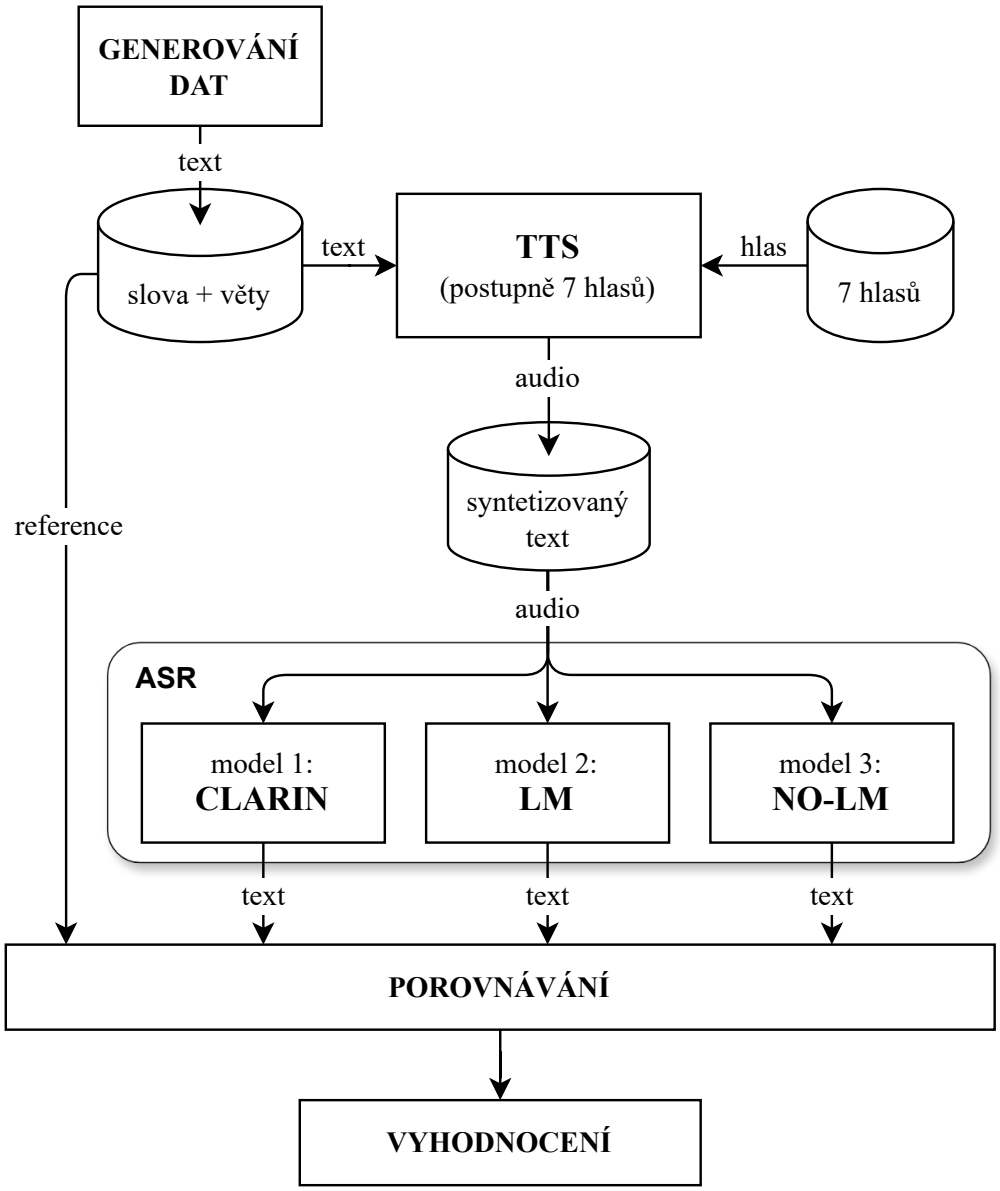
Na následujícím Obrázku 6.1 je názorně zobrazen průběh procesu testování rozpoznávání řeči (ASR) a syntézy řeči (TTS).



Obrázek 6.1: Jednoduché schéma průběhu testování ASR a TTS

Testovací proces je rozdělen do tří základních kroků, které zajistí komplexní a objektivní hodnocení každého rozpoznávače. Prvním krokem je generování dat, kde jsou vytvářeny referenční texty pro následné testování. Tyto texty jsou rozděleny do tří skupin podle účelu testování, což umožňuje detailní analýzu rozpoznávačů v růz-

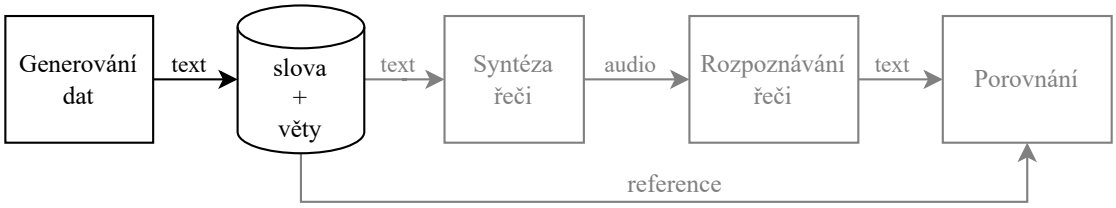
ných kontextech. Druhým krokem je syntéza řeči, která převádí tyto texty na audio nahrávky. Tento proces se provádí postupně pro každou referenci v sedmi různých hlasech, aby bylo možné zajistit rozmanitost testovacích dat a vyhodnotit flexibilitu rozpoznávačů při rozpoznávání různých typů hlasů. Nakonec ve třetím kroku dochází k samotnému rozpoznávání řeči a porovnání výsledků s referencí, což představuje závěrečnou fázi testování, ve které jsou audio nahrávky převedeny zpět na text a porovnány s původními referencemi. Porovnávání referenčního textu a výsledného přepisu z ASR modulů probíhá na základě Levenshteinovy vzdálenosti. Schéma procesu testování je zobrazeno na následujícím Obrázku 6.2.



Obrázek 6.2: Schéma průběhu testování ASR a TTS

6.1 Příprava dat

Prvním krokem, který je zvýrazněn na následujícím Obrázku 6.3, je generování vhodných dat pro testování.



Obrázek 6.3: Testování ASR a TTS - příprava dat

Příprava dat pro testování je organizována do tří specifických skupin, které reflektují rozdílné účely testování.

Jednotlivé skupiny:

a) Porovnání samostatně vysloveného slova a stejného slova použitého ve větě, aby se posoudil vliv kontextu na rozpoznávání.

b) Vygenerované věty pomocí ChatGPT[4], které testují schopnost rozpoznávačů řeči správně rozpoznávat řeč, i když struktura nebo smysl věty je neobvyklý. Jinými slovy, věty mohou obsahovat nezvyklé slovosledy, gramatické struktury nebo slovní spojení, které nejsou běžně používány v každodenní komunikaci.

c) Testování homonym, tedy slov s podobnou výslovností, ale odlišným významem (např. 'plast' a 'plášť'), což pomáhá hodnotit, jak rozpoznávače zvládají nuance výslovnosti.

Hlavním cílem je pozorování, jak různé technologie rozpoznávačů řeči, jako jsou TDNN a wav2vec, reagují na tyto výzvy, a zvláště jak se projeví absence jazykového modelu u grafémového přístupu. U rozpoznávače založeného na TDNN, kde má jazykový model velkou váhu, lze očekávat, že bude existovat silnější tendence nahrazovat nejasná nebo nesmyslná slova reálnými slovy ze slovníku. U wav2vec s jazykovým modelem lze předpokládat, že bude docházet k nahrazování chybně rozpoznaných slov správnými s menší intenzitou, zatímco u grafémového přístupu lze očekávat nejpřesnější reprezentaci výslovnosti na základě jednotlivých grafémů.

[4] `https://chat.openai.com`

a) Samostatné slovo vs. slovo ve větě

Tato skupina testů se zaměřuje na porovnání rozpoznávání slov, kdy jsou vyslovována samostatně, a rozpoznávání stejných slov, kdy jsou začleněna do kontextu věty. Cílem je zjistit, jak kontext věty ovlivňuje přesnost rozpoznávání slov rozpoznávačem řeči. Tento test umožní pozorovat, zda a jak se změní schopnost rozpoznávače správně identifikovat slovo, když je součástí většího textového celku. V následující části je uveden ilustrativní příklad, který demonstruje, jak by mohly být jednotlivé rozpoznávače řeči ovlivněny, pokud jsou vybaveny jazykovým modelem nebo slovníkem.

Lze si představit situaci, kdy je cílem vyslovit slovo „*včela*“, ale dojde k jeho zkomolení tak, že výslovnost se více podobá „*včema*“ ($l \rightarrow m$). V běžném životě, stejně jako u strojů provádějící automatické rozpoznávání řeči, není snadné rozlišit, zda takto vyslovené slovo je opravdu zkomolené, anebo zda je slovo vyslovené správně, ale posluchač ho nezná, analogicky u strojů by nemuselo být v jazykovém modelu či slovníku. V tomto případě, kdy je slovo vysloveno samostatně bez jakéhokoliv kontextu, může rozpoznávač vrátit přímo slovo „*včema*“, nebo jako výsledek rozpoznávání vrátí některé z nejbližších existujících slov ze slovníku, například „*včera*“ nebo „*včela*“ (pro jednoduchost jsou uvedeny pouze dvě možnosti).

Tento přístup je typický pro rozpoznávače, které využívají jazykový model a slovník. Avšak když je zkomolené slovo „*včela*“ použito ve větě poskytující kontext, například „*Z úlu vyletěla **včema** a letěla na květ.*“, lze s velkou pravděpodobností odvodit, že „*včema*“ je ve skutečnosti „*včela*“. Ve větě „*Koupil si **včema** ten dárek, nebo ho mám koupit dnes já?*“ naznačuje kontext, že „*včema*“ je pravděpodobně „*včera*“. Když mají rozpoznávače nastavenou silnou váhu jazykového modelu, mají tendenci rozpoznávat slova, která existují ve slovníku. V některých případech je toto chování žádoucí. Avšak v situacích, kde je cílem zachytit drobné chyby ve slově, automatické nahrazování zkomoleného slova nejbližším existujícím slovem ze slovníku není vhodné.

V této skupině je testováno celkem 12 výchozích slov různých délek a ke každému slovu jsou přidány překlepy/změny. Celkem je 80 samostatných slov, a tím pádem stejný počet vět, takže test **a)** celkem obsahuje 160 textových řetězců. Výpis jednotlivých slov a vět lze nalézt na sdíleném disku[5]. Seznam jednotlivých slov včetně

[5] `https://drive.google.com/drive/folders/1cCyiWMzffOWDOZoxuq8F-gc3TOsh_1Ik`

změn je zobrazen v Tabulce 6.1. Jednotlivé varianty výchozích slov jsou doplněny do předem připravených míst ve větách, které jsou uvedeny v Tabulce 6.2.

Tabulka 6.1: Seznam variant samostatných slov pro test ASR - **a)**

Výchozí slovo	**Jednotlivé varianty slova** (originál + překlepy/změny)
les:	les, los, lez, las, lis, kes
ves:	ves, vos, vez, vaz, vis
pes:	pes, pos, pez, paz, pis
nás:	nás, náz, néz, más, máz, ráz, vás
auto:	auto, auot, atuo, autto, aoto, uato, auta
hora:	hora, bota, bora, hoar, horx, hroa, hota, horv
stůl:	stůl, stúl, sůl, stěl, stál, sťůl, ztůl
bicykl:	bicykl, bicykil, bicyl, bicikl, bycikl, biclyk
knihovna:	knihovna, knihoovna, knhiovna, knihovnna, knihovnaa
veverka:	veverka, vevorka, veverko, voverka, vaverka, veverda, meverka, neverka, keverka
stříkačka:	stříkačka, stříkočka, stříkečka, míchačka, střékačka, ztříkačka, střídečka, střídačka
architektura:	architektura, architektra, architekturaa, arhitektura, archtiktura, architektrua, architekutura

Tabulka 6.2: Seznam vět pro doplnění jednotlivých variant pro test ASR - **a)**

Výchozí slovo	**Věta** (s místem pro doplnění jednotlivých variant)
les	V dálce vidím jehličnatý _____.
ves	Nedaleko od města byla malá _____.
pes	Za plotem na mě štěkal _____.
nás	Bylo _____ pět.
auto	Po silnici jezdí _____.
hora	Sněžka je nejvyšší _____ v ČR.
stůl	Koupil jsem si nový kancelářský _____.
bicykl	V muzeu je vystavený historický _____.
knihovna	Studentům je k dispozici univerzitní _____.
veverka	Po stromě běhala _____.
stříkačka	K odběru krve se používá injekční _____.
architektura	Ve městě byla vidět pouze barokní _____.

b) Vygenerované věty ChatGPT

Testování s větami vygenerovanými ChatGPT se zaměřuje na schopnost rozpoznávačů řeči rozpoznat a správně interpretovat věty, které nemusí být strukturálně nebo významově optimálně sestavené. Tyto věty mohou obsahovat neobvyklé konstrukce nebo mohou být formulovány způsobem, který není typický pro běžnou řeč. Tento test umožní posoudit, jak jazykové modely v rozpoznávačích řeči reagují na atypické nebo neočekávané formulace a struktury vět a jak přítomnost nebo absence jazykového modelu ovlivňuje celkovou přesnost rozpoznávání.

Tento test obsahuje celkem 93 vět. Seznam vět je k dispozici na sdíleném disku[5]. V následující části jsou ukázány příklady 4 vygenerovaných vět:

1. *Rychlý hnědý lišák přeskakuje přes líného psa.*
2. *Dnes je krásný den na piknik v parku.*
3. *Slunce zapadá na západě, malujíc oblohu odstíny oranžové a růžové.*
4. *V srdci města se každé ráno ožívá rušný trh.*

c) Homonyma

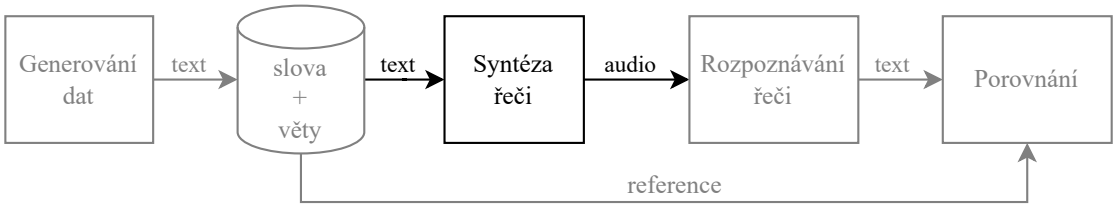
V této druhé skupině se testují slova, která se vyslovují podobně, ale mají odlišný význam – *tzv. homonyma* (například „plast“ a „plášť“). Tento test je zásadní pro hodnocení schopnosti rozpoznávačů řeči rozlišit mezi slovy, která jsou výslovnostně blízká, ale liší se významem. Porovnání, jak rozpoznávače zpracovávají tyto významové nuance, poskytne cenné informace pro volbu vhodného rozpoznávače pro aplikaci na kontrolu výslovnosti. V následující Tabulce 6.3 je celkem 7 skupin homonym, které dohromady obsahují 16 slov.

Tabulka 6.3: Seznam homonym pro test ASR - **c)**

Skupina	**Slova**
homonyma 1	oběd, objet
homonyma 2	svát, sval
homonyma 3	sálat, salát
homonyma 4	rada, ráda, řada
homonyma 5	plast, plást, plášť
homonyma 6	bedna, jedna
homonyma 7	včela, včera

6.2 Syntéza řeči

Dalším krokem je syntéza řeči, která probíhá pomocí modelu TTS poskytovaného platformou SpeechCloud, jak je podrobněji popsáno v předchozí kapitole 5. Hlavním úkolem v této části je převedení textu na audio nahrávku, tato část je zvýrazněná na následujícím Obrázku 6.4.



Obrázek 6.4: Testování ASR a TTS - syntéza řeči

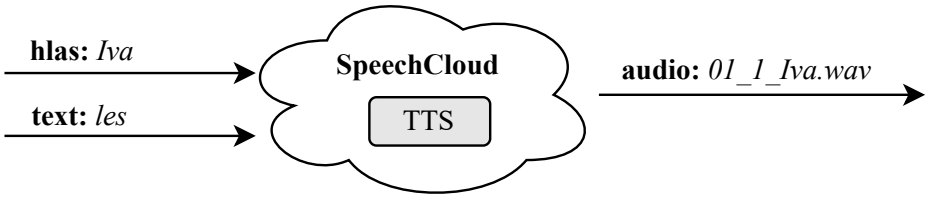
Syntéza využívá metodu výběru jednotek, známou jako *Unit selection*, o které se lze více dočíst v předchozí kapitole 2.5.3. Aby byl test co nejkomplexnější, pro každý textový řetězec je syntéza provedena pomocí sedmi různých hlasů – čtyř ženských a tří mužských, přičemž každý hlas je pojmenován křestním jménem. Seznam jednotlivých jmen je uveden v následující Tabulce 6.4.

Tabulka 6.4: Seznam jmen jednotlivých hlasů syntézy

číslo	hlas
1.	Iva
2.	Jan
3.	Jiří
4.	Kateřina
5.	Radka
6.	Stanislav
7.	Alena

Metoda *Unit selection* [59] byla vybrána zejména proto, že umožňuje syntézu hlasu generovat pomocí řetězení přímo z řečových jednotek, které jsou skutečně nahrány od člověka, nikoliv uměle generovány z parametrů modelu. Vzhledem k tomu, že syntéza je využívána pro testování jednotlivých rozpoznávačů a tento test má za cíl co nejpřesněji simulovat výslovnost lidských slov, je využití syntézy obsahující skutečné hlasové nahrávky považováno za výhodnější, i když nemusí být tak příjemná na poslech jako některé novější metody syntézy.

Pro skupinu **a)** bylo připraveno 160 textových řetězců, pro skupinu **b)** 93 textových řetězců a pro skupinu **c)** 16 textových řetězců. Každý z těchto řetězců byl syntetizován do řeči celkem 7krát, pokaždé s jiným hlasovým modelem. Z toho plyne, že celkový počet audio nahrávek je 7násobkem počtu textových řetězců. S celkem 269 připravenými textovými řetězci ve všech třech skupinách dosahuje celkový počet audio nahrávek čísla **1883**. Syntéza probíhala přes server SpeechCloud a trvala v řádu jednotek, popřípadě v nižších řádech desítek minut pro všechny nahrávky. Navíc probíhala jednorázově, což eliminovalo potřebu řešit paralelizace odesílání požadavků na syntézu.



Obrázek 6.5: Ilustrativní příklad syntézy řeči

Na Obrázku 6.5 je znázorněn postup syntézy řeči včetně příkladu jednotlivých vstupních dat a výstupního audio souboru WAV. Všechny výsledné nahrávky jsou k nalezení na sdíleném disku[6], kde jsou rozřazeny do jednotlivých složek podle struktury:

a) Samostatné slovo vs. slovo ve větě (160 textových řetězců, 1120 audio nahrávek)
- Samostatná slova (80 textových řetězců, 560 audio nahrávek)
- Slova ve větě (80 textových řetězců, 560 audio nahrávek)

b) Vygenerované věty ChatGPT (93 textových řetězců, 651 audio nahrávek)

c) Homonyma (16 textový řetězců, 112 audio nahrávek)

V každé složce jsou jednotlivé nahrávky pojmenovány následovně:

`<ID>_<number_string>_<voice>.wav`

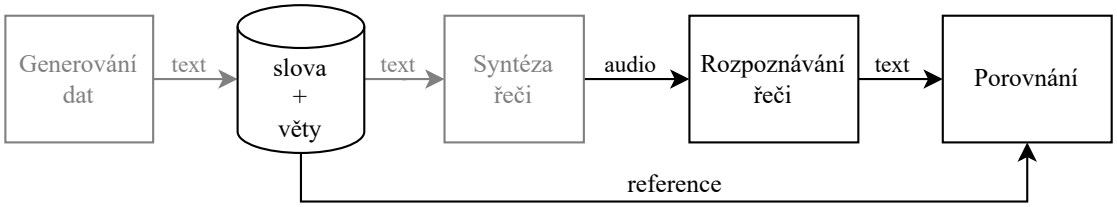
kde `ID` je identifikační číslo nahrávky v dané složce, `number_string` je číslo daného textového řetězce a `voice` je použitý hlas. Ukázka:

`01_1_Iva.wav` (01. nahrávka, 1. věta, hlas Iva)
`02_1_Jan.wav` (02. nahrávka, 1. věta, hlas Jan)
⋮
`64_9_Iva.wav` (64. nahrávka, 9. věta, hlas Iva)

[6] `https://drive.google.com/drive/folders/1fT7ahot1ZqbZADG3HYY-nxyZ5bABxbRu`

6.3 Rozpoznávaní řeči a porovnání

Posledním krokem při testování je převedení audio nahrávky zpět na text pomocí automatického rozpoznávání řeči, které zajišťují tři různé modely ASR. Následně je převedený text porovnán s referenčním textem pomocí Levenshteinovy vzdálenosti (LD). Na následujícím Obrázku 6.6 jsou tyto akce zvýrazněné.



Obrázek 6.6: Testování ASR a TTS - rozpoznávání řeči a porovnání

Testování probíhalo pomocí tří rozpoznávačů řeči. První z nich využívá architekturu TDNN a jazykový model, zatímco zbývající dva modely jsou založeny na architektuře wav2vec. Z nich jeden využívá jazykový model a druhý je založen převážně na grafémovém přístupu. Označení jednotlivých rozpoznávačů:

1) model: CLARIN (architektura TDNN, jazykový model)
2) model: LM (wav2vec, jazykový model)
3) model: NO-LM (wav2vec, grafémový přístup)

1) model - CLARIN:

- **Akustický model:** Používá architekturu TDNN (Time Delay Neural Network) se vstupním kontextem 0,2 sekundy na 16 kHz signálu, který je parametrizován pomocí PLP (Perceptual Linear Prediction) metody. Model byl natrénován na rozsáhlém datasetu obsahujícím 1900 hodin nahrávek od 6600 řečníků, nejprve pomocí kritéria křížové entropie (cross-entropy) a poté sekvenčním diskriminačním tréninkem.
- **Dekódování:** Proces dekódování využívá Viterbiho algoritmu nad fonémovým výstupem akustického modelu, s použitím slovníku obsahujícího 1,2 milionu slov a 3-gramového jazykového modelu s 55 miliony n-gramů.
- **Shrnutí:** S jeho pokročilým akustickým modelem a rozsáhlým slovníkem může být účinný v rozpoznávání řeči, ale jeho 3-gramový jazykový model by mohl mít tendenci korektovat výslovnostní chyby na základě kontextu a frekvence slov, což není ideální pro účely kontroly výslovnosti.

2) a 3) model wav2vec - LM a NO-LM:

- **Popis:** Použitý ASR model má architekturu wav2vec 2.0 [10] v základní velikost (tzv. *base*), tj. jedná se o encoder s 12 transformerovými bloky s celkem 95 miliony trénovatelných parametrů. Pro předtrénování modelu bylo použito 80 tisíc hodin audio nahrávek s českou řečí a pro fine-tuning na úlohu ASR bylo použito téměř 6 tisíc hodin přespané české řeči z různých domén. Při dekodování přepisů je použit ještě 4-gramový jazykový model, který byl natrénován z velkého množství webových stránek z projektu Common Crawl [60]. Použitý jazykový model obsahuje téměř 5 milionů slov a modeluje pravděpodobnosti pro celkem 34 miliony různých n-gramů v českém jazyce. Podrobný popis modelu a jeho natrénování lze nalézt v [61].
- **Shrnutí LM:** Tento wav2vec model s jazykovým modelem (LM) může poskytovat ještě vyšší přesnost rozpoznávání, ale stejně jako v předchozím případě, jeho použití jazykového modelu by mohlo vést k opravování nesprávně vyslovených slov, což není žádoucí pro odhalování chyb ve výslovnosti.
- **Shrnutí NO-LM:** Tento wav2vec model založený na grafémové úrovni se jeví jako nejvhodnější pro účely kontroly výslovnosti. Jeho zaměření na grafémy (základní písmena nebo znaky psaného jazyka), bez korekce přidaným jazykovým modelem na výstupu, umožňuje detekci a zachycení nepřesností ve výslovnosti na základě jednotlivých grafémů, což je klíčové pro identifikaci výslovnostních chyb.

6.3.1 Rozpoznávání řeči

Samotné rozpoznávání řeči probíhalo tak, že se postupně všechny jednotlivé audio nahrávky posílaly do všech 3 modelů ASR, které vracely rozpoznaný text. Audio nahrávky se odesílaly na server SpeechCloudu, kde probíhalo samotné rozpoznávání.

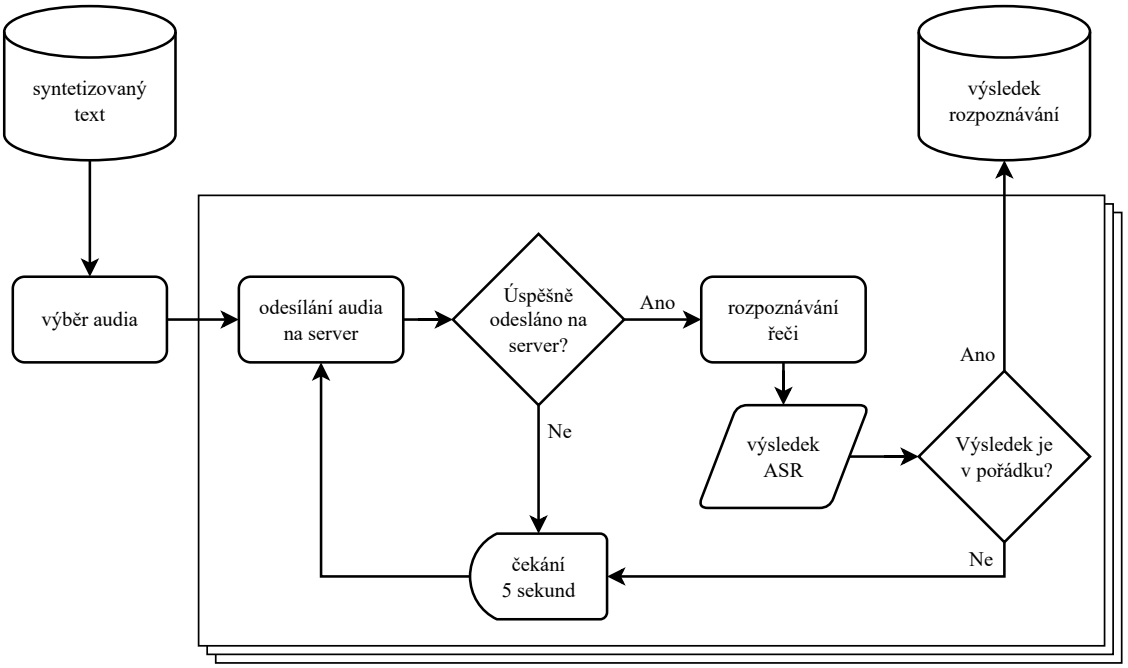
Obrázek 6.7: Ilustrativní příklad rozpoznávání řeči - paralelismus

Pro wav2vec modely byla stanovena minimální fixní doba 20 sekund na rozpoznávání, nezávisle na délce textového řetězce. U delších řetězců se doba rozpoznávání mírně prodlužovala. Průměrně byla doba rozpoznávání pro jeden wav2vec model 22 sekund, zatímco pro model CLARIN to bylo pouze 6 sekund. Celkově tak všechny tři rozpoznávače vyžadovaly pro rozpoznání jednoho textového řetězce průměrně 50 sekund (22 + 22 + 6 sekund). Vzhledem k tomu, že bylo potřeba rozpoznat 1883 textových řetězců, celková doba rozpoznávání by bez dalších úprav trvala přibližně 1570 minut, což odpovídá více než 26 hodinám. Z tohoto důvodu byla využita paralelizace jednotlivých požadavků, která je jednoduše schématicky znázorněna na Obrázku 6.7.

Pro tento test bylo připraveno celkem 269 textových řetězců, ale vzhledem k tomu, že každý řetězec byl syntetizován sedmi různými hlasy a následně každá nahrávka byla rozpoznána třemi různými rozpoznávači, došlo k 21násobnému zvýšení počtu výsledků rozpoznávání oproti počtu původních testovacích textů. Rozmanitost hlasů a použití více rozpoznávačů způsobuje, že i při relativně nízkém počtu textových řetězců, konkrétně v nižších řádech stovek, trvá rozpoznávání řeči více než 26 hodin. Tato kombinace zvyšuje celkový počet výsledků rozpoznávání na **5649**.

Pro testování rozpoznávačů řeči byl využit paralelismus, aby se urychlil proces rozpoznávání. Experimentálně bylo zjištěno, že server, na kterém probíhalo rozpoznávání, může obsloužit současně maximálně 15 požadavků, proto byl paralelní běh nastaven na 15 vláken. Aby byly ošetřeny případy, kdy server nepřijme požadavek kvůli

ochraně nebo přetížení, je implementována smyčka, ve které vlákno po neúspěšném pokusu počká 5 sekund a následně požadavek opakuje. Toto opatření se vztahuje i na případy, kdy výsledek ASR není uspokojivý (například pokud nebylo nic rozpoznáno). Tento proces je znázorněn vývojovým diagramem na následujícím Obrázku 6.8. Díky paralelizaci se doba testování zkrátila z více než 26 hodin na méně než 2 hodiny.

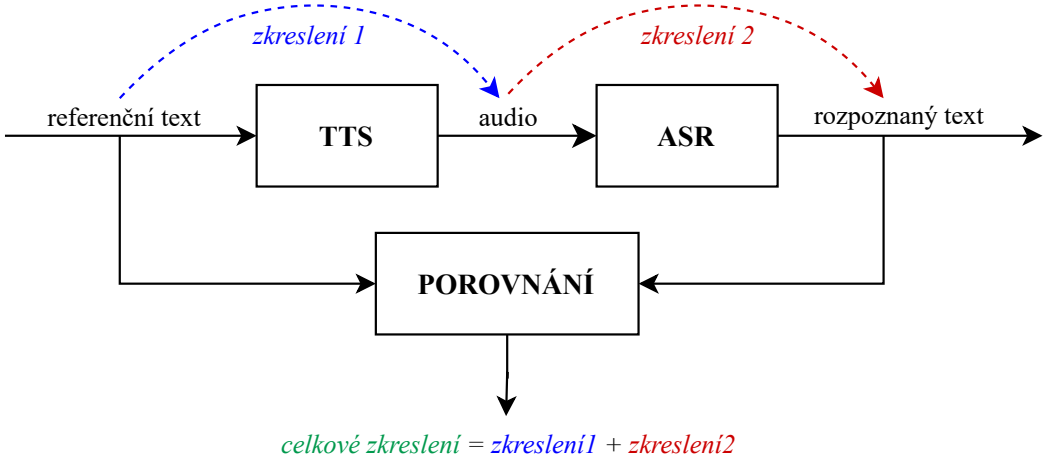


Obrázek 6.8: Vývojový diagram testování ASR - paralelismus

6.3.2 Porovnání

Aby bylo možné testování rozpoznávačů řeči kvantifikovat, je nezbytné zvolit objektivní metriku. Jako nejvhodnější se jeví použití Levenstheinovy vzdálenosti, známé také jako editační vzdálenost. Tato metoda umožňuje porovnávat dva textové řetězce a vyjadřuje počet operací (vložení, smazání, nahrazení), které jsou potřeba pro jejich vzájemnou shodnost. V další části bude tato vzdálenost označována zkratkou LD (Levensthein distance).

Je důležité si uvědomit, že existuje rozdíl mezi psaným textem a jeho výslovností, což podrobně popisuje fonetická transkripce uvedená v kapitole 2.5.1. Fonetickou transkripci zajišťují jednotlivé TTS moduly, a proto se kvalita výslovnosti pro jednotlivé hlasy může lišit. Je nutné brát v úvahu i nesprávnou výslovnost v syntetizovaných nahrávkách, což může ovlivnit výsledky testování. Na Obrázku 6.9 je znázorněno možné dvojí zkreslení, které může nastat při testování.



Obrázek 6.9: Schéma celkového zkreslení při testování ASR

Může nastat případ, kdy je referenční text převeden na audio nahrávky s chybou ve výslovnosti (například záměna písmena) - toto představuje první zkreslení. Poté audio nahrávka projde rozpoznávačem řeči, který může vlivem jazykového modelu rozpoznat tuto audio nahrávku s chybnou výslovností jako původní text - toto je druhé zkreslení. Výsledek rozpoznávání pak může být identický s referenčním textem, což je však způsobeno dvojím zkreslením, kde druhé zkreslení vynuluje první. Výsledek může být zkreslený, ačkoliv se zdá být s nulovou editační chybou. Příklad:

reference: kostel $\xrightarrow[\text{k}\to\text{h}]{\text{TTS}}$ **audio:** hostel $\xrightarrow[\text{h}\to\text{k}]{\text{ASR}}$ **text:** kostel

S tímto scénářem je nutné počítat, protože neexistuje jiný objektivnější způsob testování. Uživatel může sice poslechem nahrávky zkontrolovat, co slyší, ale to již představuje spíše subjektivní než objektivní testování. Tento nepříjemný fakt může zvýhodňovat modely s jazykovým modelem, což je důležité brát v úvahu při hodnocení jejich výkonnosti.

Porovnávání výsledného rozpoznaného textu s referencí se provádí po následujících úpravách:

- Převedení textových řetězců na malá písmena.
- Odstranění interpunkčních znaků.

Poté se pomocí Levensteinovy metody vypočítá jejich vzdálenost.

Při porovnávání ve skupině **a)** se pro nalezení daného slova ve větě používá Levensteinova vzdálenost s minimální cenou. To znamená, že se hledá řetezec, který může mít libovolný začátek a konec a má nejmenší Levensteinovu vzdálenost. Hlavním důvodem použití této upravené metody pro hledání jednotlivých slov ve větě je skutečnost, že rozpoznávač může vrátit slovo s mezerami, například „*stří kačka*“ místo „*stříkačka*“, nebo špatně rozpoznané jako „*stří kočka*“. Tato upravená Levensteinova metoda umožňuje automaticky nalézt dané slovo ve větě a porovnat ho s rozpoznaným slovem bez kontextu. Nevýhodou je, že nalezení slova nemusí být přesné, zejména pokud hledané vyslovené slovo obsahuje mnoho chyb. Nicméně i přes tyto omezení je tento přístup ve většině případů velmi účinný.

Jak již bylo zmíněno při diskuzi o fonetické transkripci pro TTS, existuje rozdíl mezi výslovnostní a psanou formou. Toto platí nejen pro modul TTS, který si fonetickou transkripci zajišťuje sám, ale i pro zpětné dekódování, tedy převod z výslovnostní formy zpět na textovou podobu, což zajišťují jednotlivé ASR moduly. V tomto testování není při vyhodnocování brána v úvahu žádná jiná alternativa pro výslovnost než referenční text. Hlavním důvodem je testování schopností jednotlivých rozpoznávačů zvládat situace, kde existuje více přípustných variant výslovnosti, zejména u i/y a párových souhlásek (například „*lež*“ vyslovované jako *[leš]*). Tato metodika poskytuje cenné informace, které budou následně zohledněny a integrovány do vyhodnocování ve webové aplikace pro kontrolu výslovnosti.

6.4 Dílčí výsledky

Testování je rozděleno do tří skupin, přičemž každá skupina se zaměřuje na rozdílné účely. V následujících sekcích jsou prezentovány dílčí výsledky ze skupin:

a) Samostatné slovo vs. slovo ve větě
b) Vygenerované věty ChatGPT
c) Homonyma

Cílem testování je identifikovat nejlepší rozpoznávač, který je nejvhodnější pro automatickou kontrolu výslovnosti. Detailní výsledky pomohou odhalit klíčové vlastnosti a výkonnost jednotlivých modelů ASR v rámci specifických testovacích scénářů.

Vzhledem k tomu, že testování je rozsáhlé a generuje velké množství výsledků, protože jsou použity tři rozpoznávače ASR a syntéza je prováděna sedmi různými hlasovými modely, existuje pro jeden testovací textový řetězec 21 různých výsledků. Kompletní výsledky jsou přiloženy v samostatném PDF souboru, který je dostupný na sdíleném disku[7]. Tento dokument obsahuje 180 stran. V dalších částech této práce budou prezentovány pouze vybrané zajímavé aspekty.

6.4.1 a) Samostatné slovo vs. slovo ve větě

V této skupině je hlavním cílem testování zkoumání rozpoznávání samostatně vysloveného slova a jeho následného začlenění do věty, což přidává kontext. Test se zaměřuje na rozdíly v rozpoznávání a na to, jaký vliv na výsledky bude mít přítomnost jazykového modelu v rozpoznávačích. Testovaná slova zahrnují nejen reálná slova, ale i uměle vytvořená slova nebo slova s překlepy a chybami. Rozpoznávače s jazykovým modelem by v kontextově bohatých situacích mohly mít tendenci nahrazovat nesprávně vyslovená slova správnými slovy ze slovníku, což je pro aplikaci určenou ke kontrole výslovnosti nežádoucí. Kompletní výsledky jsou k nalezení v samostatném PDF souboru[7] v Tabulkách 6, 7, 8. V následující Tabulce 6.5 je ukázána struktura hlavičky tabulek **a)**.

Tabulka 6.5: Ukázka struktury hlavičky tabulek pro testování **a)**

ASR	**Hlas**	**Slovo**	**LD_1**	**Slovo ve větě**	**LD_2**
<number_string>:<výchozí slovo>	hlas	<varianta slova>	LD_1	<varianta věty>	LD_2

[7] https://drive.google.com/drive/folders/1U71YtkWSlpfyC0R2E8IUpjTmRmmYADv-

Význam jednotlivých elementů v Tabulce 6.5 je následující:

- **hlas** = jméno použitého hlasu pro syntetizovanou nahrávku;
- **LD** = Levensteinova vzdálenost mezi referenční variantou slova nebo věty a výsledkem rozpoznávače;
- `<number_string>` = číslo textového řetězce;
- **<výchozí slovo>** = slovo, ze kterého jsou odvozeny jednotlivé varianty referenčního slova;
- **<varianta slova>** = referenční slovo, odvozené od výchozího slova;
- **<varianta věty>** = referenční věta, zahrnující referenční slovo.

S využitím informace `number_string` a **hlasu** je možné si přehrát danou nahrávku ve formátu WAV[6], která odpovídá výsledkům uvedeným v konkrétní tabulce.

V následující Tabulce 6.6 je uveden výsledek testování slova „*las*“, které bylo odvozeno od výchozího slova „*les*“. V hlavičce tabulky je specifikováno `number_string` s číslem 3. Tělo tabulky v tomto ukázkovém příkladu obsahuje výsledek testování pouze pro jeden hlas, konkrétně pro hlas Kateřina (v kompletních výsledcích jsou zobrazeny výsledky pro všech 7 hlasů). Název nahrávky pro tento konkrétní případ, kde `number_string` je číslo 3 a hlas patří Kateřině, je `025_3_Kateřina.wav`. Pro přehrání syntetizovaného slova nebo věty je nutné najít tento název v odpovídající složce podle typu testování. V kolonce LD_2 jsou uvedeny dvě hodnoty oddělené lomítkem. První číslo reprezentuje Levenshteinovu vzdálenost pro celou větu. Druhé číslo, zvýrazněné modře, ukazuje Levenshteinovu vzdálenost pro specifické slovo ve větě, které je zvýrazněno modře. Toto slovo je identifikováno automaticky pomocí upravené Levenshteinovy metody, která minimalizuje vzdálenost.

Tabulka 6.6: Výsledek testování slova „*las*“ - **a)**

ASR	**Hlas**	**Slovo**	**LD_1**	**Slovo ve větě**	**LD_2**
3:les	hlas	las	LD_1	V dálce vidím jehličnatý las.	LD_2
LM	Kateřina	hlas	1	v dálce vidím jehličnatý les	1/1
NO-LM	Kateřina	hlas	1	v dálce vidím jehličnatý las	0/0
CLARIN	Kateřina	las	0	v dálce vidím jehličnatý les	1/1

Výsledky testování uvedené v Tabulce 6.6 pro **samostatné slovo** „*las*“ ukazují, že oba modely wav2vec (LM a NO-LM) rozpoznaly slovo jako „*hlas*“, což znamená

jednu chybu v Levenshteinově vzdálenosti (LD), zatímco model CLARIN poskytl správný výsledek, tedy slovo „*las*“. Zde se nabízí otázka, zda je přítomnost písmene „*h*“ na začátku slova chybou rozpoznávače, nebo zda jde o problém se syntézou. Samotný proces syntézy může být nedokonalý a může vést k přidání nechtěných zvuků nebo deformací ve výslovnosti. Například, pokud TTS systém nepřesně modeluje začátek slova, může dojít k vytvoření nechtěného zvuku, který se může zdát jako „*h*“. Jelikož pro všech šest zbylých hlasů poskytly oba modely wav2vec (LM a NO-LM) správné výsledky „*las*“, zdá se, že problém se syntézou u tohoto modelu je pravděpodobnější. Naopak, výsledky modelu CLARIN s ostatními hlasy byly velmi různorodé a zahrnovaly varianty jako *„láz“*, *„les“*, *„ráz“*, *„kalas“*, *„nás“*.

Výsledky testování uvedené v Tabulce 6.6 pro **slovo ve větě** ukazují, že rozpoznávače s jazykovým modelem (LM a CLARIN) rozpoznají slovo „*las*“ jako slovo „*les*“, což je důsledkem vlivu jazykového modelu v kontextu věty. Jediný správný výsledek poskytl model NO-LM, který správně rozpoznal slovo „*las*“. U zbylých šesti hlasů správně rozpoznaly slovo „*las*“ oba modely wav2vec (LM a NO-LM), zatímco model CLARIN ve všech případech nesprávně identifikoval slovo jako „*les*“. Tento efekt je očekáván, jelikož model CLARIN s výrazným jazykovým modelem nahradí neznámé slovo „las“ slovem „les“, které je v jeho slovníku nejbližší známou alternativou. Tento výběr je odvozen z minimální Levenshteinovy vzdálenosti a je ovlivněn také kontextem věty. Naopak wav2vec model LM, který sice obsahuje jazykový model, avšak s menší váhou, se ukázal být přesnější. V ostatních případech správně rozpoznal slovo „*las*“. Podle výsledků tohoto testu se jako nejvhodnější volba rozpoznávače jeví wav2vec s grafémovým přístupem (NO-LM), který dokázal ve všech sedmi případech správně rozpoznat slovo „*las*“.

Druhým příkladem uvedeným v Tabulce 6.7 je výsledek testování slova „*lis*“, které je odvozeno od výchozího slova „*les*“.

Tabulka 6.7: Výsledek testování slova „*lis*“ - **a)**

ASR	**Hlas**	**Slovo**	**LD$_1$**	**Slovo ve větě**	**LD$_2$**
4:les	hlas	lis	LD$_1$	V dálce vidím jehličnatý lis.	LD$_2$
LM	Jiří	elis	1	v dálce vidím jehličnatý lis	0/0
NO-LM	Jiří	elis	1	v dálce vidím jihličnatý lis	1/0
CLARIN	Jiří	rys	2	v dálce vidím jehličnatý les	1/1

U obou wav2vec modelů bylo rozpoznáno slovo „*elis*“, přičemž na začátku tohoto slova se objevuje písmeno „*e*“. Tento jev je pravděpodobně způsoben tzv. *švou* (redukovanou samohláskou /ə/), která se objevila na začátku slova v syntetizovaném hlase. V kontextu věty, kde slovo není umístěno na první pozici, tento jev zmizí, a oba wav2vec modely správně rozpoznají slovo „*lis*“. U modelu CLARIN bylo jako samostatné slovo nesprávně identifikováno „*rys*“ a ve větě bylo vlivem kontextu nesprávně nahrazeno slovem „*les*“. Zajímavým aspektem je také slovo „*jehličnatý*“, které bylo modelem NO-LM rozpoznáno jako „*jihličnatý*“ (změna *e* na *i*). Při přehrání nahrávky se zdálo, že písmeno „*i*“ bylo výraznější než „*e*“ ve slově „*jehličnatý*“, což opět ukazuje na problém syntézy. I když bylo slovo „*jehličnatý*“ oběma rozpoznávači s jazykovým modelem rozpoznáno jako správné, výsledky jsou zkreslené nesprávnou syntézou. Tato informace je klíčová při vyhodnocování jednotlivých výsledků, protože číselné vyjádření výsledků v tomto případě je zkreslené. Oba rozpoznávače s jazykovým modelem by měly mít o jednu chybu (LD) navíc a naopak rozpoznávač bez přidaného jazykového modelu (NO-LM) o jednu chybu méně.

Dalším příkladem uvedeným v Tabulce 6.8 je výsledek testování slova „*vos*“, které je odvozeno od výchozího slova „*ves*“.

Tabulka 6.8: Výsledek testování slova „*vos*“ - **a)**

ASR	**Hlas**	**Slovo**	**LD$_1$**	**Slovo ve větě**	**LD$_2$**
7:ves	hlas	vos	LD$_1$	Nedaleko od města byla malá vos.	LD$_2$
LM	Iva	vos	0	nedaleko od města byla malá voz	1/1
NO-LM	Iva	voz	1	nedaleko od města byla malá voz	1/1
CLARIN	Iva	vůz	2	nedaleko od města byla malá vous	1/1
LM	Jan	mos	1	nedaleko od města byla malá vos	0/0
NO-LM	Jan	vos	0	nedaleko od města byla malávos	1/0
CLARIN	Jan	nos	1	nedaleko od města byla marná vlas	4/2

Jednoznačně nejhorším rozpoznávačem je v tomto příkladě opět model CLARIN, který má tendenci nahrazovat neznámá slova existujícími slovy ze slovníku. Například, slovo „*vos*“ bylo modelem CLARIN rozpoznáno jako „*vůz*“, „*vous*“, „*nos*“, „*vlas*“, dokonce v jednom případě bylo ve větě nahrazeno slovo „*malá*“ slovem „*marná*“. Rozpoznávače wav2vec ve většině případů správně rozpoznaly slovo „*vos*“, nebo ho zaměnily za „*voz*“, což je přijatelná varianta. Tato záměna je vysvětlena tím, že se jedná o párové souhlásky (s-z), které se ve výslovnosti mohou někdy za-

měňovat. V tomto testu není záměna párových souhlásek zvýhodněna nižší chybou (LD), což vede k nesprávnému započítávání chyb. Tento fakt je důležité zohlednit při vyhodnocování výsledků a je rovněž nezbytné tyto případy řádně ošetřit při vyhodnocení v rámci webové aplikace. U modelu LM bylo v jednom případě slovo „*vos*“ nesprávně rozpoznáno s jednou chybou (LD). Dalším zajímavým pozorováním je výsledek u modelu NO-LM pro hlas Jan, kde dvě slova byla rozpoznána spojeně, což bylo způsobeno nedostatečnou pauzou mezi slovy v syntéze a absencí jazykového modelu, který by mohl identifikovat alespoň jedno existující slovo „*malá*“ a tím oddělit jednotlivá slova.

Předposledním příkladem uvedeným v Tabulce 6.9 je výsledek testování slova „*bycikl*“, které je odvozeno od výchozího slova „*bicykl*“. Tento test se zaměřuje na způsoby, jakými jednotlivé rozpoznávače řeči zvládají záměnu měkkého a tvrdého i/y. Výslovnost samotných samohlásek i/y je identická, což znamená, že rozpoznávače řeči by měly být schopné je rozlišit pouze na základě kontextu v textu nebo s pomocí jazykového modelu, který bere v úvahu pravopisné pravidla českého jazyka. Rozpoznávač bez přidaného jazykového modelu nemá tyto kontextové informace, a proto může docházet k chybám v rozpoznání správné formy i/y.

Tabulka 6.9: Výsledek testování slova „*vos*“ - **a)**

ASR	**Hlas**	**Slovo**	**LD_1**	**Slovo ve větě**	**LD_2**
49:bicykl	hlas	bycikl	LD_1	V muzeu je vystavený historický bycikl.	LD_2
LM	Iva	bicykl	2	v muzeu je vystavený historický bicykl	2/2
NO-LM	Iva	bicikl	1	v muzeu je vystavený historický bicykl	2/2
CLARIN	Iva	bicykl	2	múzou je vystavený historický bicykl	6/2

Rozpoznávače s jazykovým modelem korektně rozlišily mezi měkkým a tvrdým i/y v souladu se správným pravopisem ve slovníku, a to jak pro samostatná slova, tak pro slova ve větě. Naopak model bez přidaného jazykového modelu (NO-LM), rozpoznal samostatné slovo jako „*bicikl*“ a ve větě jako „*bicykl*“. Vzhledem k tomu, že výslovnost samostatného měkkého a tvrdého i/y je shodná, protože měkké „*i*“ pouze v některých případech změkčuje předchozí souhlásku, je vhodné v rámci webové aplikace nezapočítávat záměnu tvrdého a měkkého i/y jako chybu. V tomto případě by si oba rozpoznávače s jazykovým modelem polepšily o 4 chyby (LD) při závěrečném vyhodnocení, zatímco model NO-LM by si polepšil o 3 chyby. Dále u modelu CLARIN bylo spojení „*v muzeu*“ nesprávně rozpoznáno jako „*múzou*“.

Posledním příkladem uvedeným v Tabulce 6.10 je výsledek testování slova „*stříkočka*“, které je odvozeno od výchozího slova „*stříkačka*“.

Tabulka 6.10: Výsledek testování slova „*stříkočka*“ - **a)**

ASR	**Hlas**	**Slovo**	**LD$_1$**	**Slovo ve větě**	**LD$_2$**
66:stříkačka	hlas	stříkočka	LD$_1$	K odběru krve se používá injekční stříkočka.	LD$_2$
LM	Kateřina	stří kočka	1	k odběru krve se používá injekční stří kočka	1/1
NO-LM	Kateřina	stříkočka	0	k odběru krve se používá injekční stříkočka	0/0
CLARIN	Kateřina	z tří kočka	3	k odběru krve se používá injekční stříkačka	1/1

U modelu NO-LM je výsledek rozpoznání v obou případech správný, bez jediné chyby. U wav2vec modelu s jazykovým modelem (LM) došlo při rozpoznávání k jedné chybě, neboť testované slovo bylo rozpoznáno s mezerou jako „*stří kočka*“. To je pravděpodobně způsobeno tím, že slovo „*kočka*“ je existující slovo ve slovníku, což vedlo k rozdělení na dvě slova. Na rozdíl od modelu CLARIN, model LM alespoň nenahradil slovo ve větě existujícím slovem „*stříkačka*“. Dále model CLARIN rozpoznal samostatné slovo jako tři slova, která existují ve slovníku, a tím přidal navíc dvě mezery. Výsledek rozpoznání je „*z tří kočka*“. Tento příklad znovu ukazuje, že pro účely webové aplikace je model CLARIN nejméně vhodnou volbou, za ním následuje wav2vec model s jazykovým modelem (LM), zatímco jako nejlepší volba se jeví wav2vec model (NO-LM), který je založený na grafémovém přístupu.

Shrnutí testování skupiny a), která se zaměřila hlavně na pozorování, jak se mění rozpoznávání samostatného slova a slova ve větě. Tento test odhalil několik zajímavých poznatků, které jsou užitečné pro vyhodnocování webové aplikace. Prvním zjištěním bylo, že při testování samostatného slova docházelo v některých případech k přidání nechtěných písmen na začátek slova, což bylo způsobeno švou nebo jinými rušivými zvuky. Je důležité prozkoumat, zda je to způsobeno výhradně nesprávným modelem syntézy, nebo zda by podobné chyby mohly vzniknout i při běžném užívání jazyka člověkem. Jelikož tento jev ve větě zmizel, problém by mohlo vyřešit uvození slova slovním spojením, například „*Toto je ...*“ nebo „*Na obrázku je ...*“, kde by tato úvodní slova mohla být při vyhodnocování výslovnosti snadno vyfiltrována. Další zjištění se týkají párových souhlásek, jejichž výslovnost se může vzájemně zaměňovat, což by nemělo být započítáváno jako chyba (LD). Stejně tak záměna měkkého a tvrdého i/y, které se vyslovují identicky, nebude zahrnována do počítání chyb. Posledním poznatkem je, že v některých případech může jazykový model způsobit,

že nesprávně vyslovená slova jsou rozdělena mezerou na dvě slova, pokud některá část slova obsahuje existující slovo ze slovníku. Na druhé straně, modely bez přidaného jazykového modelu někdy nemusí správně rozlišit dvě existující slova kvůli nedostatečné pauze mezi nimi, nebo naopak mohou rozdělit jedno slovo na dvě, pokud uživatel během slova udělá výraznější pauzu. Tento fakt je důležité zohlednit při výběru rozpoznávače, protože v některých případech je vhodné upozornit například na zbytečně dlouhou pauzu ve slově, zatímco v jiných případech může být pauza ve slově pro zdůraznění, například pomocí rázu, vhodná.

V následující Tabulce 6.11 jsou uvedeny výsledky testování skupiny a) pro všechny tři rozpoznávače. Model CLARIN se ukázal jako jednoznačně nejhorší, s celkovým počtem chyb 1461, což je téměř o polovinu více než u modelů wav2vec, které mají oba kolem 970 chyb (model LM má 971 a model NO-LM 965). Rozpoznávání samostatného slova bylo úspěšnější u modelu NO-LM, který má o 62 chyb méně než model LM. Tento rozdíl je způsoben větší tendencí modelu LM nahrazovat slova ze slovníku, což vede k zanedbávání nuancí ve výslovnosti. Naopak v rozpoznávání slov ve větě má model NO-LM o 56 chyb více než model LM, což je způsobeno hlavně špatnou syntézou TTS, která vede ke zkresleným výsledkům. Model NO-LM vykazuje celkově nejnižší počet chyb a nejlepší schopnost detekovat nesprávně vyslovená slova.

Tabulka 6.11: Výsledky testování ASR pomocí LD (počet chyb) - **a)**

	CLARIN		**LM**		**NO-LM**		**TTS suma**
hlas	slovo	věta	slovo	věta	slovo	věta	
Iva	94	136	61	62	51	72	476
Jan	99	99	73	77	63	86	497
Jiří	105	106	78	60	63	58	470
Kateřina	94	109	80	57	67	57	464
Radka	97	103	57	67	52	71	447
Stanislav	98	104	73	76	66	98	515
Alena	108	109	75	75	73	88	528
ASR suma	695	766	497	474	435	530	**3397**
	1461		**971**		**965**		

6.4.2 b) Vygenerované věty ChatGPT

Druhým testovaným scénářem jsou věty vygenerované pomocí ChatGPT. Tento test se zaměřuje na to, jak si jednotlivé rozpoznávače poradí s větami, které nemusí být strukturálně nebo optimálně sestavené. Hlavním cílem je zjistit, jak se různé modely vypořádají s těmito výzvami a jaký vliv na rozpoznávání bude mít jazykový model. Kompletní výsledky jsou k nalezení v samostatném PDF souboru[7] v Tabulkách 9, 10, 11. Jednotlivé syntetizované nahrávky ve formátu WAV lze přehrát pomocí čísla věty uvedeného v hlavičce tabulky, které odpovídá `number_string`, a pomocí konkrétního hlasu. První uvedený příklad je v následující Tabulce 6.12.

Tabulka 6.12: Výsledek testování 1. věty - **b)**

ASR	**Hlas**	**Slovo ve větě**	**LD**
1. věta	hlas	Rychlý hnědý lišák přeskakuje přes líného psa.	LD
LM	Iva	rychlý hnědý lišák přes kaku je přes líného psa	2
NO-LM	Iva	rychlý hnědý lyšák přes kaku je přes líného psa	3
CLARIN	Iva	rychlý hnědý lišák přeskakuje přes líného psa	0
LM	Jan	rychlý hnědý lišák přeskakuje přes líného psa	0
NO-LM	Jan	rychlý hnědý lyšák přeskakuje přes líného psa	1
CLARIN	Jan	rychlý hnědý lišák přeskakuje přes líného psa	0
LM	Jiří	rychlý hnědý lišák přeskakuje přes líného psa	0
NO-LM	Jiří	rychlý hnědý lyšák přeskaku je přes líného psa	2
CLARIN	Jiří	rychlý hnědý lišák přeskakuje přes líného psa	0
LM	Kateřina	rychlý hnědý lišák přeskakuje přes líného psa	0
NO-LM	Kateřina	rychlý hnědý lyšák přes skakuje přes líného psa	3
CLARIN	Kateřina	rychlý hnědý lišák přeskakuje přes líného psa	0
LM	Radka	rychlý hnědý lišák přeskakuje přes líného psa	0
NO-LM	Radka	rychlý hnědý lyšák přeskakuje přes líného psa	1
CLARIN	Radka	rychlý hnědý lišák přeskakuje přes líného psa	0
LM	Stanislav	rychlý hnědý lišák přeskakuje přes líného psa	0
NO-LM	Stanislav	rychlý hnědý lyšák přesskakuje přes líného psa	2
CLARIN	Stanislav	rychlý hnědý lišák přeskakuje přes líného psa	0
LM	Alena	rychlý hnědý lyšák přeskakuje přes líného obsa	3
NO-LM	Alena	rychlý hnědý lyšák přeskakuje přes líného opsa	2
CLARIN	Alena	rychlý hnědý lišák přeskakuje přes líného psa	0

V Tabulce 6.12 jsou uvedeny kompletní výsledky všech hlasů a rozpoznávačů. Podle počtu chyb (LD) by model CLARIN vypadal jako nejlepší rozpoznávač s nulovými chybami, následovaný modelem LM s 5 chybami a modelem NO-LM s 15 chybami. Přestože se na první pohled zdá, že model CLARIN zvládl situaci nejlépe a model NO-LM nejhůře, realita je odlišná. Z celkových 15 chyb modelu NO-LM, sedm chyb připadá na záměnu tvrdého a měkkého i/y, jako například „*lyšák*“ místo „*lišák*“, které by se měly považovat za nulové chyby, protože se vyslovují identicky. Čtyři další chyby jsou způsobeny přidanými mezerami, zejména ve slově „*přeskakuje*“, kde bylo jednou rozpoznáno jako „*přes kaku je*“. Toto zachycení je výsledkem nesprávného hláskování syntézy, ale je žádoucí, protože odráží skutečnou výslovnost. Dvě další chyby pochází ze zdvojení písmene „*s*“ ve slově „*přesskakuje*“, které může být také přípustnou variantou výslovnosti. Poslední chyba modelu NO-LM je přidání písmene „*o*“ u slova „*opsa*“, což může být způsobeno chybou syntézy, ale přidané „*o*“ není zřetelné při poslechu. Po úpravě by měl model NO-LM mít maximálně jednu chybu. Naopak model CLARIN by mohl mít více chyb, pokud by mu byly přičteny chyby za neschopnost správně rozpoznat nuance, jako jsou zdvojená písmena nebo chybné hláskování. Tento fakt je důležité zohlednit při závěrečném číselném vyhodnocení, protože výsledky mohou být velmi zkreslené.

Dalším příkladem je 3. věta, která není strukturálně správně sestavená:

„*V srdci města se každé ráno ožívá rušný trh.*“

Tato věta by mohla být upravena odstraněním slova „*se*“ nebo nahrazením slova „*ožívá*“ slovem „*ozývá*“, aby byla jak strukturálně, tak významově správná. Možné správné varianty vět by mohly znít:

„*V srdci města každé ráno ožívá rušný trh.*“
„*V srdci města se každé ráno ozývá rušný trh.*“

Ačkoliv věta nebyla při testování optimálně strukturovaná, výsledky rozpoznávání byly bez chyb pro všechny tři rozpoznávače a skrze všech sedm hlasů. Jedině u hlasu Aleny, který se ukazuje jako nejméně přesný model TTS, byl zaznamenán problém se syntézou, kde bylo ve dvou případech místo slova „*města*“ rozpoznáno „*město*“. Ačkoliv se tento výsledek liší od referenčního textu, považuje se za správnou variantu, protože toto zkreslení je způsobeno chybou výslovnosti v syntéze. Chyba by měla být přičtena wav2vec modelu LM, který pravděpodobně vlivem jazykového modelu rozpoznal slovo „*města*“ nesprávně, aby věta byla syntakticky správná.

V rozpoznávání nebyl problém dokonce ani u 6. věty:

„*Teplá šálek čaje je dokonalý lék na chladný zimní den.*“

Tato věta obsahuje gramatickou nesrovnalost na začátku, kde je přídavné jméno „*teplá*“ ve špatném rodu. Správně by mělo být „*teplý*“, aby odpovídalo mužskému rodu slova „*šálek*“. Rozpoznávače s jazykovým modelem tento jazykový nedostatek nekorigovaly.

S čím ale rozpoznávače měly větší problém, byla slova, která si umělá inteligence vymyslela, tedy slova neobsažená v jazykovém modelu. Dále pak slova, která se v běžné řeči nepoužívají často nebo jsou zastaralá, jako například přechodníky. Prvním příkladem je 9. věta obsahující přechodník „*lákajíc*“:

„*Vůně čerstvě upečeného chleba se nese vzduchem, lákajíc kolemjdoucí.*“

Model CLARIN rozpoznal 6krát ze 7 případů nesprávně přechodník jako slovo „*lákající*“, což je pravděpodobně způsobeno vyšším zastoupením tohoto slova v jazykovém modelu, zatímco správně přechodník rozpoznal pouze v jednom případě u hlasu Ivy. Druhý rozpoznávač s jazykovým modelem, wav2vec model LM, dokázal správně rozpoznat přechodník 5krát ze 7 případů a 2krát nesprávně nahradil slovem „*lákají*“. Nejlépe si s touto výzvou poradil rozpoznávač NO-LM bez přidaného jazykového modelu, který správně rozpoznal přechodník 6krát ze 7 případů a v 1 případě odhalil výslovnostní chybu způsobenou syntézou, kdy správně rozpoznal slovo „*lákajít*“.

Dalším příkladem je přechodník „*vodíc*“, který je použit v 11. větě:

„*Hvězdy jasně svítí na noční obloze, vodíc cestovatele na jejich cestě.*“

Model CLARIN tento přechodník nesprávně rozpoznal v každém případě, místo toho nabídl alternativy jako „*vévodí*“, „*vodítek*“, „*vodík*“ a „*vodí*“. Modely wav2vec si vedly lépe, neboť byly schopné přechodník správně rozpoznat v několika případech. Model LM často zaměňoval slovo „*vodíc*“ za „*vodí*“, zatímco model NO-LM toto slovo rozpoznal správně častěji, ale dvakrát ho rozpoznal jako „*vodí z*“, což by mohlo být považováno za možnou variantu správného rozpoznání díky velmi podobné výslovnosti. Dalším příkladem je neexistující slovo „*proudučného*“, které si umělá inteligence vymyslela a je obsaženo ve 32. větě:

„*Zvuk proudučného potoka je hudbou pro milovníky přírody.*“

Model CLARIN toto slovo nerozpoznal ani jednou správně a místo toho rozpoznal vždy slova „*proudu černého*“, což jsou existující slova ve slovníku, která jsou vymyšlenému slovu nejblíže. Naopak wav2vec modely slovo „*proudučného*“ rozpoznaly vždy správně. Model LM však ve třech případech slovo nesprávně rozdělil na dvě slova „*proudu čného*“.

V následující Tabulce 6.13 jsou číselné výsledky testu **b)**. Na první pohled se může zdát, že nejlepším rozpoznávačem je model LM, který výrazně předčí oba zbývající rozpoznávače. Toto však neodráží skutečnou situaci. Model CLARIN je opravdu nejméně úspěšný, protože není schopen správně rozpoznat slova mimo svůj slovník a místo toho je nahrazuje jinými slovy z jeho slovníku. Oba modely wav2vec výkonnostně výrazně převyšují model CLARIN. Přestože počet chyb u modelu NO-LM je na první pohled poměrně vysoký, je důležité poznamenat, že většina chyb přičítaných tomuto modelu ve skutečnosti nejsou pravé chyby. Jedná se o přípustné výslovnostní varianty, které nejsou v tomto testu brány v potaz, například chybějící či přebývající mezery vlivem syntézy („*barvilistů*“/„*barvy listů*“), výslovnostní varianta („*bombonů*“/„*bonbonů*“) nebo („*vůňje*“/„*vůně*“) a záměna i/y („*melodyi*“/„*melodii*“). Těchto chyb je ve skutečnosti mnohem více, což vede k nespravedlivě vysokému počtu chyb přičítaných modelu NO-LM, přestože neodrážejí skutečnou kvalitu rozpoznávání. Pro webové aplikace je právě tento způsob rozpoznávání nejvhodnější. Test dále odhalil, že špatná stavba věty není pro rozpoznávače velkou překážkou.

Tabulka 6.13: Výsledky testování ASR pomocí LD (počet chyb) - **b)**

hlas	**CLARIN**	**LM**	**NO-LM**	**TTS suma**
Iva	59	31	52	142
Jan	102	53	96	251
Jiří	61	37	47	145
Kateřina	103	55	90	248
Radka	57	25	57	139
Stanislav	95	38	57	190
Alena	101	68	90	259
ASR suma	**578**	**307**	**489**	**1374**

6.4.3 c) Homonyma

V poslední skupině c) se testují homonyma, což jsou slova, která jsou výslovnostně velmi blízká, ale liší se významem. Testování se zaměřuje na schopnost jednotlivých rozpoznávačů rozpoznat výslovnostní nuance a na vliv jazykového modelu na rozpoznávání. Testování zahrnovalo celkem 7 skupin homonym s dohromady 16 slovy. Kompletní výsledky jsou k nalezení v samostatném PDF souboru[7] v Tabulce 12.

První skupinou homonym byla slova „*oběd*“ a „*objet*“, kde výsledky testování jsou uvedeny v Tabulce 6.14.

Tabulka 6.14: Výsledek testování homonyma 1 - **c)**

hlas	**CLARIN**		**LM**		**NO-LM**	
0:homonyma1	oběd	LD_1	oběd	LD_2	oběd	LD_3
Alena	oběd	0	opět	2	opět	2
ostatní hlasy	oběd	0	oběd	0	oběd	0
1:homonyma1	objet	LD_1	objet	LD_2	objet	LD_3
Alena	oběd	3	opět	3	opět	3
ostatní hlasy	oběd	3	oběd	3	oběd	3

Testované slovo „*oběd*“ bylo pro všechny hlasy, s výjimkou hlasu Aleny, rozpoznáno správně. U hlasu Aleny wav2vec model rozpoznal slovo jako „*opět*“, což by nemělo být považováno za chybu, jelikož se jedná o párové souhlásky (b-p) a (t-d), které se ve výslovnosti mohou zaměňovat. Navíc ve skutečné nahrávce hlas Aleny vyslovuje písmeno „*p*“ místo „*b*“. Výsledky rozpoznávání pro slovo „*objet*“ jsou shodné s výsledky slova „*oběd*“. Vzhledem k tomu, že „*ě*“ se vyslovuje jako „*je*“ a záměna (t-d) je znovu způsobena párovými souhlásky, by se v tomto případě pro všechny výsledky měla počítat nulová chyba. Jelikož testování nezahrnovalo žádné jiné varianty výslovnosti než je referenční vzor, jsou tyto výsledné chyby nesprávně započítávány.

V druhé skupině homonym, kde byla testována slova „*svát*“ a „*sval*“, se slovo „*svát*“ u modelů často rozpoznávalo jako „*zvát*“, což je způsobeno zaměňováním párových souhlásek (s-z). U hlasu Ivy a modelu CLARIN bylo slovo „*svát*“ nesprávně rozpoznáno jako „*svátek*“. Naopak slovo „*sval*“ bylo modelem CLARIN vždy správně rozpoznáno jako referenční vzor. U modelů wav2vec se slovo někdy rozpoznalo jako „*sválo*“, což v některých případech reflektovalo přesnější variantu způsobenou chybnou syntézou, která přidávala koncové písmeno. Další zajímavostí je, že někdy bylo

písmeno „*a*“ nesprávně zaznamenáno jako „*á*“, což bylo v některých případech považováno za přesnější variantu, i když číselné výsledky to neodrážely. Při hodnocení se jak chybějící, tak přebývající čárka nad „*a*“ považuje za stejnou chybu (LD) s hodnotou jedna. Tento způsob hodnocení by měl být revidován, protože chyba ve výslovnosti, kde je špatně uvedeno „*a*“ místo „*á*“, by měla být ohodnocena jako menší chyba než například záměna „*a*“ za „*e*“.

Dalším příkladem uvedeným v následující Tabulce 6.15 jsou homonyma 4, která zahrnují tři testovaná slova: „*rada*“, „*ráda*“ a „*řada*“.

Tabulka 6.15: Výsledek testování homonyma 4 - **c)**

hlas	**CLARIN**		**LM**		**NO-LM**	
6:homonyma4	rada	LD_1	rada	LD_2	rada	LD_3
Jiří	rada	0	porada	2	porada	2
Jan, Radka	ráda	1	rada	0	rada	0
ostatní hlasy	rada	0	rada	0	rada	0
7:homonyma4	ráda	LD_1	ráda	LD_2	ráda	LD_3
všechny hlasy	ráda	0	ráda	0	ráda	0
8:homonyma4	řada	LD_1	řada	LD_2	řada	LD_3
všechny hlasy	řada	0	řada	0	řada	0

Se slovy „*ráda*“ a „*řada*“ neměl žádný ze tří rozpoznávačů u všech sedmi hlasů žádný problém a vždy byly rozpoznány správně. U slova „*rada*“ však model CLARIN ve dvou případech chybně rozpoznal slovo jako „*ráda*“, tedy s přidanou čárkou nad „*a*“. U modelů wav2vec se u hlasu Jiří slovo „*rada*“ rozpoznalo jako „*porada*“, což se po přehrání nahrávky skutečně ukázala jako přesnější varianta. V tomto případě budou opět číselné výsledky zkreslené.

Posledním příkladem uvedeným v následující Tabulce 6.16 jsou homonyma 5, která obsahují tři testovaná slova: „*plast*“, „*plást*“, a „*plášt*“. Model CLARIN v jednom případě nahradil slovo „*plast*“ slovem „*vlast*“ a „*oblast*“. Zatímco slovo „*vlast*“ může být považováno za přípustnou variantu s ohledem na výslovnost, slovo „*oblast*“ je od syntetizované nahrávky značně odlišné. U modelů wav2vec bylo slovo „*plast*“ v jednom případě zaměněno za „*past*“, což je s ohledem na syntézu také přípustná varianta. Slovo „*plást*“ bylo u modelů wav2vec někdy rozpoznáno bez dlouhé čárky nad „*a*“. Slovo „*plášt*“ bylo ve všech případech správně rozpoznáno.

Tabulka 6.16: Výsledek testování homonyma 5 - **c)**

hlas	**CLARIN**		**LM**		**NO-LM**	
9:homonyma5	plast	LD_1	plast	LD_2	plast	LD_3
Iva	plast	0	past	1	past	1
Kateřina	vlast	1	plast	0	plast	0
Radka	oblast	2	plast	0	plast	0
ostatní hlasy	plast	0	plast	0	plast	0
10:homonyma5	plást	LD_1	plást	LD_2	plást	LD_3
Stanislav	plást	0	plást	0	plást	0
Jan, Radka	plást	0	plast	1	plast	1
ostatní hlasy	plást	0	plast	1	plást	0
11:homonyma5	plášť	LD_1	plášť	LD_2	plášť	LD_3
všechny hlasy	plášť	0	plášť	0	plášť	0

V následující Tabulce 6.17 je shrnuto testování skupiny **c)**. Nové vhodné poznatky, které by se odlišovaly od těch v předchozích skupinách **a)** a **b)**, se v této skupině neobjevují. Ačkoliv model CLARIN vykazuje nejlepší číselné hodnocení, není považován za nejlepší volbu rozpoznávače. Důvodem je zkreslení výsledků vlivem nesprávné syntézy a absence přípustných výslovnostních variant oproti referenčním slovům, což negativně ovlivňuje model wav2vec bez přidaného jazykového modelu.

Tabulka 6.17: Výsledky testování ASR pomocí LD (počet chyb) - **c)**

hlas	**CLARIN**	**LM**	**NO-LM**	**TTS suma**
Iva	5	9	6	20
Jan	4	8	8	20
Jiří	3	7	7	17
Kateřina	7	9	5	21
Radka	9	5	5	19
Stanislav	3	5	7	15
Alena	4	10	6	20
ASR suma	**35**	**53**	**44**	**132**

6.5 Závěrečné vyhodnocení

Testování rozpoznávačů nabízí komplexní a objektivní hodnocení, které je zajištěno nejen použitím tří rozdílných rozpoznávačů řeči, ale také zahrnutím sedmi různých hlasů z text-to-speech (TTS) systémů. Díky tomuto přístupu lze posoudit celkovou vhodnost každého rozpoznávače, jeho flexibilitu a adaptabilitu na různé hlasové profily a výslovnostní charakteristiky, což je pro účely aplikace zásadní. Rovněž rozmanitý výběr hlasů umožňuje hodnotit adaptabilitu a efektivitu každého rozpoznávače ve vztahu k různým typům hlasového výstupu a poskytuje příležitost vyhodnotit, která z těchto syntéz je „nejlepší“ ve smyslu její schopnosti být správně rozpoznána rozpoznávači řeči. Porovnání referenčního textu a výsledného přepisu z ASR modulů se provádí na základě Levenshteinovy vzdálenosti, což poskytuje kvantifikovatelný způsob měření přesnosti rozpoznávání tím, že se vyhodnocuje počet nutných editačních operací pro převedení jednoho řetězce na druhý. Tento přístup zajišťuje, že hodnocení efektivity rozpoznávačů řeči je založeno na robustním a objektivním základě. Je však důležité poznamenat, že jednotlivé výsledky mohou být zkreslené vlivem špatné kvality syntézy, což je nutné zohlednit při celkovém vyhodnocení.

Rozsáhlé testování rozdělené do tří skupin poskytlo důležité informace o výběru rozpoznávače vhodného pro webovou aplikaci určenou k automatické kontrole výslovnosti. V následující Tabulce 6.18 jsou prezentovány celkové výsledky testů jednotlivých rozpoznávačů vyjádřené pomocí Levenshteinovy vzdálenosti (LD), která indikuje chyby v rozpoznávání.

Tabulka 6.18: Vyhodnocení jednotlivých rozpoznávačů řeči

pořadí	**model ASR**	**LD**
1.	LM	1337
2.	NO-LM	1500
3.	CLARIN	2085

Model CLARIN je jednoznačně nejhorším rozpoznávačem ve všech třech scénářích testování. Naproti tomu oba modely wav2vec vykazují výrazně lepší výsledky a jsou podle počtu chyb o hodně vzdáleny od modelu CLARIN. Ačkoliv model LM má celkově nejméně chyb, doporučeným modelem pro automatickou kontrolu výslovnosti je model NO-LM, založený na grafémovém přístupu. Přestože se na první pohled může zdát, že se jeho rozpoznávání často odchyluje od referenčního textu, ve skutečnosti

nejlépe odhaluje výslovnostní nuance ve všech testovacích situacích. Tento zdánlivý rozdíl je způsoben nekvalitní syntézou a neuznáváním různých výslovnostních variant, což vede ke zkreslení výsledků. Pro webovou aplikaci je schopnost modelu NO-LM odhalit jednotlivé výslovnostní chyby zásadní.

Zároveň je pro správné použití modelu NO-LM ve webové aplikaci důležité upravit vyhodnocení tak, aby zahrnovalo následující poznatky:

- zohlednění chybějících či přebývajících mezer,
- uznání všech výslovnostních variant, například:
  - záměna měkkého a tvrdého i/y,
  - záměna „u“ s čárkou a s kroužkem ú/ů,
  - záměna párových souhlásek (p-b, v-f, t-d, atd.),
- snížení chyb v důsledku chybějící diakritiky (a/á, e/é, atd.).

V následující Tabulce 6.19 je vyhodnocení kvality syntézy jednotlivých hlasů.

Tabulka 6.19: Vyhodnocení syntézy jednotlivých hlasů

pořadí	**hlas**	**LD**
1.	Radka	605
2.	Jiří	632
3.	Iva	638
4.	Stanislav	720
5.	Kateřina	733
6.	Jan	768
7.	Alena	807

Lze si všimnout, že první tři hlasy – Radka, Jiří a Iva – jsou si výsledkově velmi blízké, přičemž jsou považovány za jednoznačně nejlepší. Následuje je Stanislav, za ním Kateřina a Jan a jednoznačně nejhorší syntézu vykazuje hlas Aleny. Ačkoliv hlas Radky má z pohledu skóre nejlepší hodnocení, ve webové aplikaci bude použit hlas Ivy. Skóre obou hlasů jsou velmi podobná s rozdílem pouhých 33 chyb, avšak hlas Ivy je považován za příjemnější na poslech. Tento údaj potvrzuje i hodnocení poskytovatele jednotlivých syntéz, který uvádí syntézu Ivy jako nejlépe hodnocenou.

Shrnutí výsledků testování ASR je zobrazeno v Tabulce 6.20.

Tabulka 6.20: Souhrnné výsledky testování ASR pomocí LD (počet chyb) - **a)**, **b)**, **c)**

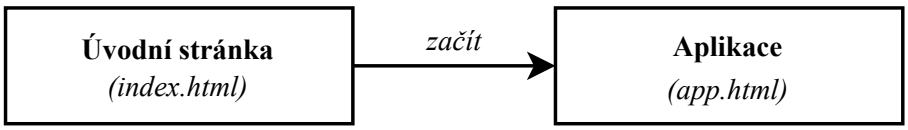
	CLARIN				**LM**				**NO-LM**				**TTS suma**
hlas	**a)**	**b)**	**c)**	**vše**	**a)**	**b)**	**c)**	**vše**	**a)**	**b)**	**c)**	**vše**	
Iva	230	59	5	294	123	31	9	163	123	52	6	181	638
Jan	198	102	4	304	150	53	8	211	149	96	8	253	768
Jiří	211	61	3	275	138	37	7	182	121	47	7	175	632
Kateřina	203	103	7	313	137	55	9	201	124	90	5	219	733
Radka	200	57	9	266	124	25	5	154	123	57	5	185	605
Stanislav	202	95	3	300	149	38	5	192	164	57	7	228	720
Alena	217	101	4	322	150	68	10	228	161	90	6	257	807
ASR suma	1461	578	35	**2074**	971	307	53	**1331**	965	489	44	**1498**	**4903**

7 Návrh webové aplikace

Cílem této práce je navrhnout a realizovat webovou aplikaci pro automatickou kontrolu výslovnosti uživatele. Princip aplikace by měl spočívat tom, že systém uživateli předkládá sérii obrázků a uživatel musí vyslovit název předmětu, který se na obrázku vyskytuje. V případě potřeby bude moci uživatel využít textovou či hlasovou nápovědu. Aplikace pak průběžně hodnotí jeho výslovnost a poskytuje zpětnou vazbu na základě procentuální shody vysloveného slova s referenčním textem. Během návrhu aplikace je kladen důraz na responzivní design, což zajišťuje správné zobrazení na různých typech zařízení a velikostech displejů. Současně bude kladen důraz na jednoduchost přidávání nových obrázků do aplikace, což uživatelům umožní procvičování výslovnosti na dalších slovech.

7.1 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní (GUI - Graphical User Interface) je rozděleno do dvou webových stránek (znázorněné na Obrázku 7.1).



Obrázek 7.1: Schéma struktury webové aplikace

7.1.1 Úvodní stránka

Úvodní stránka poskytuje základní *informace* o aplikaci, ukázky *vyhodnocení* výslovnosti a příklady použitých *obrázků*. Návrh začátku úvodní stránky je zobrazen na Obrázku 7.2, pod nímž se postupně umístí zmíněné *sekce*.

Obrázek 7.2: Návrh úvodní stránky

7.1.2 Aplikace

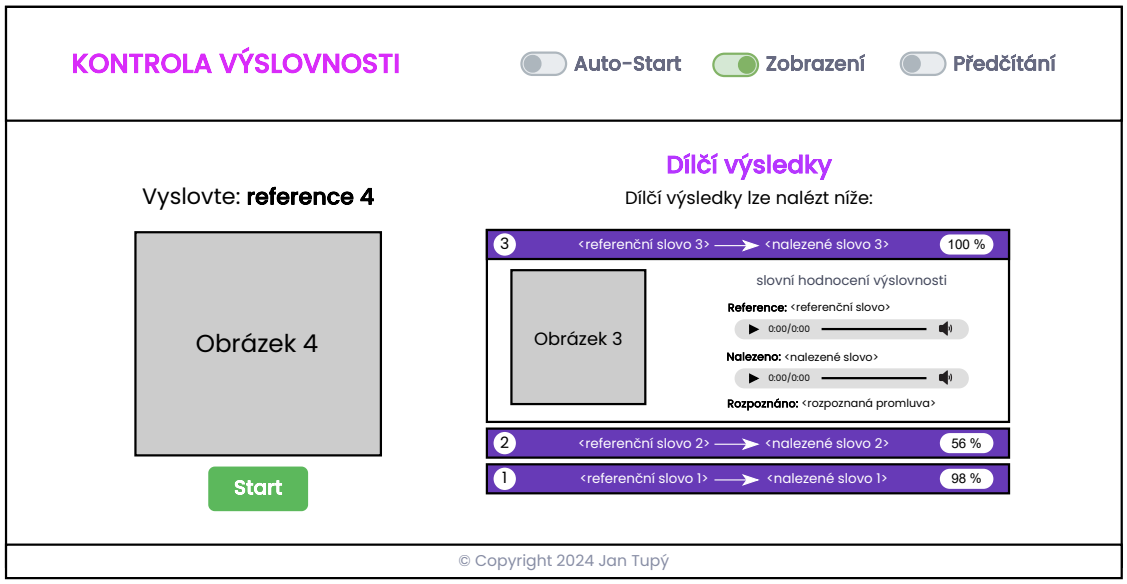
Do aplikace je možné se dostat pomocí tlačítka „*Začít s kontrolou.*“ na úvodní stránce. Po vstupu do aplikace se uživateli předloží výběr verze:

- **Testovací verze**, která slouží pro účely testování aplikace.
- **Produkční verze**, která je určena na běžný provoz aplikace.

Po výběru verze se uživateli postupně zobrazí vyskakovací okna (tzv. *pop-up okna*), která uživatele připravují na použití aplikace:

- **Nastavení**: Zde je možné si nastavit automatické spouštění následujících slov nebo nastavit, zda má být aktivovaná textová či hlasová nápověda.
- **Upozornění a doporučení**: Okno, které uživateli oznamuje, co všechno by si měl zkontrolovat, před zahájením výslovnosti a poskytuje přehled doporučení.
- **Instrukce**: Okno, kde jsou k nalezení jednotlivé instrukce k použití aplikace.

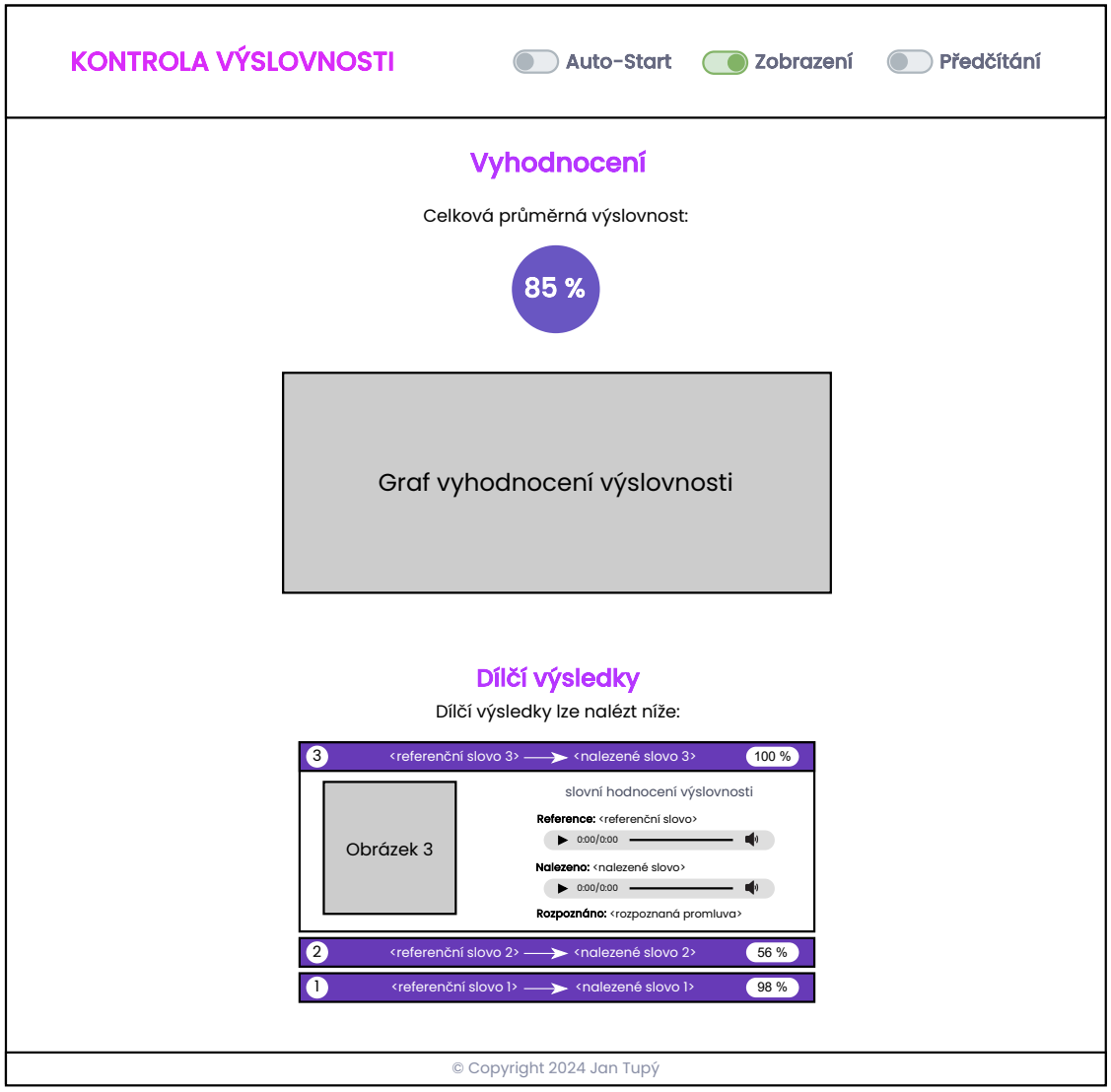
Po instrukcích následuje hlavní část aplikace, jejíž návrh je na Obrázku 7.3.



Obrázek 7.3: Návrh průběhu aplikace

V hlavičce stránky je možné i v průběhu kontroly měnit jednotlivá nastavení pomocí přepínačů. V levé části stránky je umístěn obrázek obsahující předmět, jehož název by měl uživatel vyslovit. Pravá část stránky zobrazuje předchozí výsledky, kde je textový přepis rozpoznané promluvy a její hlasová nahrávka, a také nalezené slovo, což je nejshodnější slovo s referencí, v rozpoznané promluvě. Shoda těchto dvou slov je vyjádřena v procentech vždy v hlavičce jednotlivých výsledků.

Po zkontrolování všech připravených slov, se zobrazí celkové vyhodnocení. Návrh vzhledu stránky s konečným vyhodnocením je zobrazen na Obrázku 7.4.



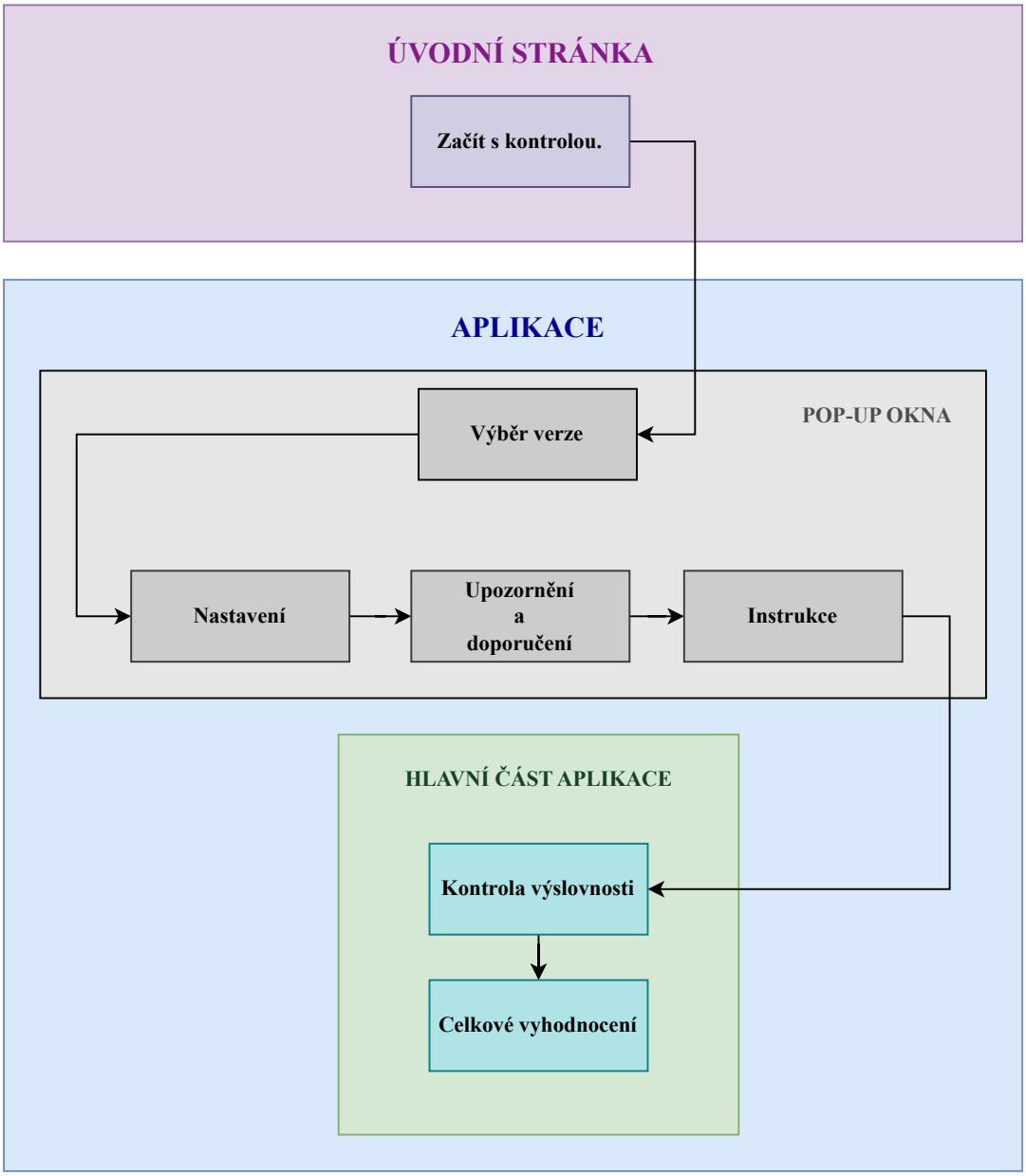
Obrázek 7.4: Návrh vyhodnocení aplikace

V úvodu stránky s vyhodnocením je celková průměrná výslovnost, která je procentuálně číselně vyjádřena jako průměr jednotlivých dílčích výsledků. Dále je na stránce umístěn graf vyhodnocení výslovnosti, který by měl sloužit jako přehledné shrnutí. Pod grafem na stránce jsou potom opět umístěny jednotlivé výsledky, které se zobrazují již v průběhu kontroly výslovnosti.

Je potřeba upozornit, že tyto návrhy, zejména pak jejich rozvržení, se můžou lišit v závislosti na různých typech zařízení a velikostech displejů.

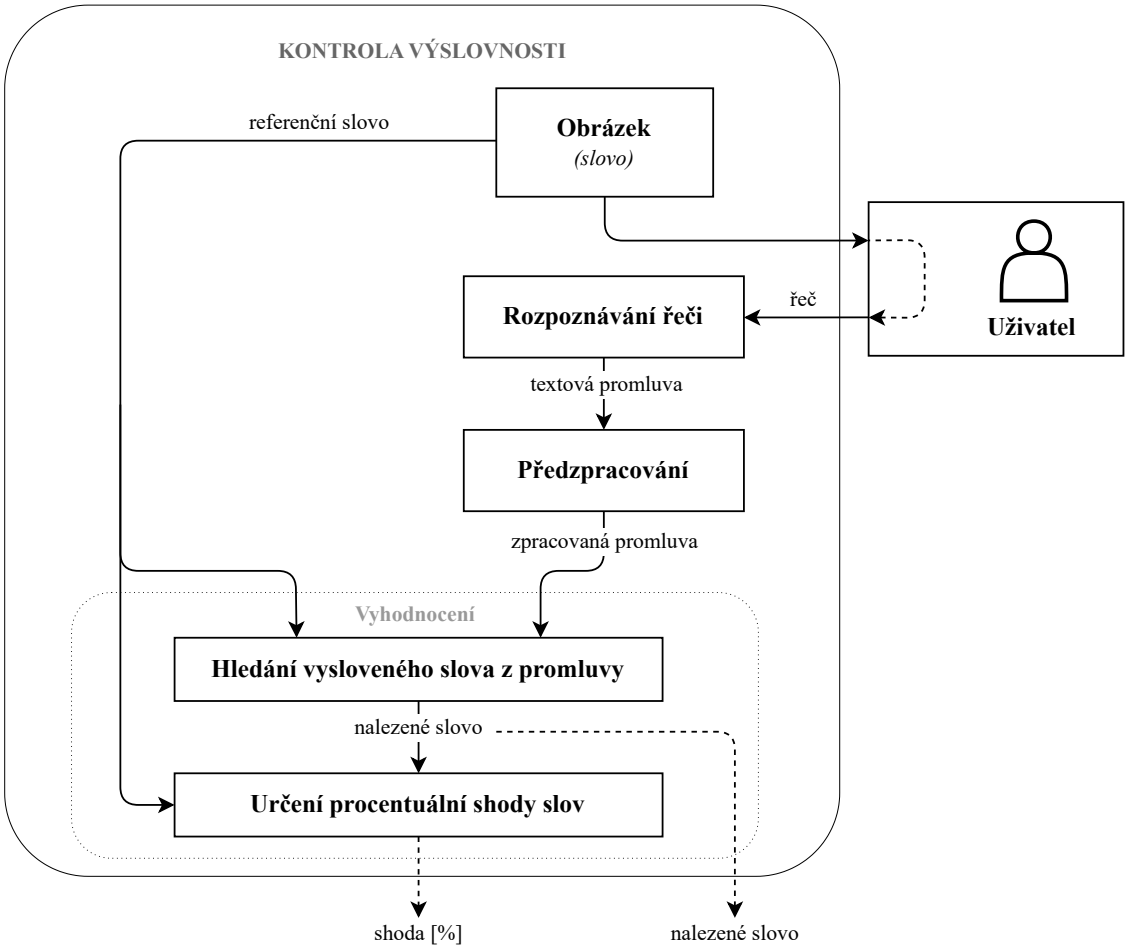
7.2 Průběh aplikace

Uživateli se jako první zobrazí **ÚVODNÍ STRÁNKA**, ze které se pomocí tlačítka „*Začít s kontrolou.*“ přesměruje do **APLIKACE**. Zde se mu nejprve zobrazí série *pop-up oken* (výběr verze, nastavení, upozornění a doporučení, instrukce). Následně se přechází do **hlavní část aplikace**, kde nejprve probíhá kontrola výslovnosti, zahrnující průběžné hodnocení, a na závěr se objeví celkové vyhodnocení. Na Obrázku 7.5 je ilustrován obecný průběh webovou aplikací.



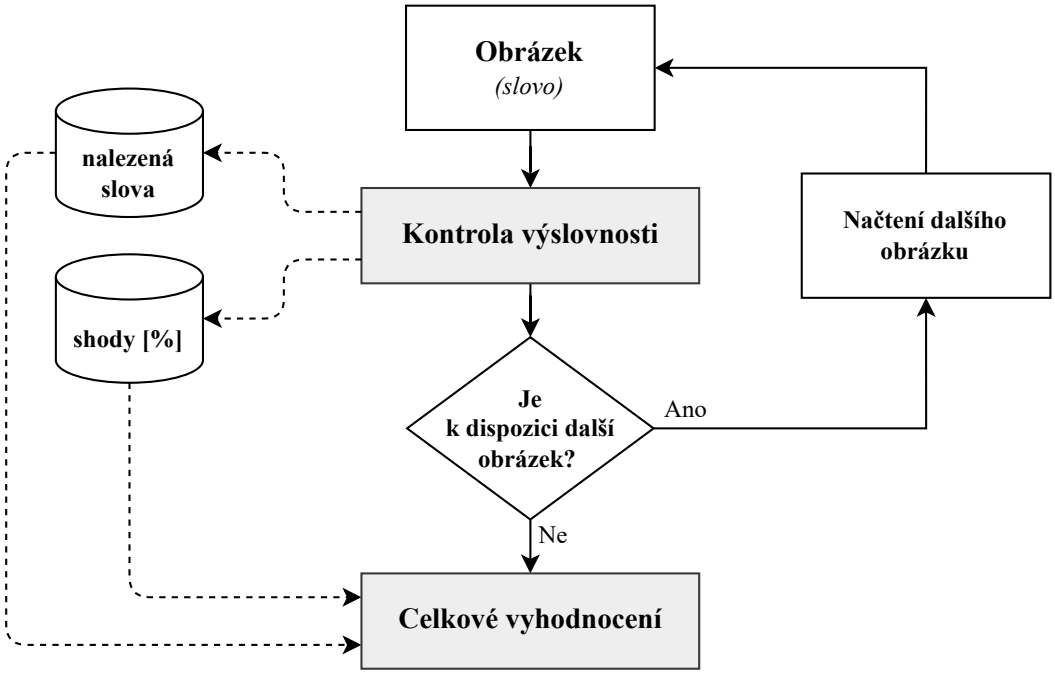
Obrázek 7.5: Obecný průběh webové aplikace

Na Obrázku 7.6 je zobrazen průběh kontroly výslovnosti jednoho slova. Nejdříve je uživateli předložen obrázek, znázorňující nějaké slovo. Uživatel následně vysloví co nejsrozumitelněji dané slovo z obrázku. Má na výběr, zda slovo vysloví samostatně nebo ve větě, ale musí být vysloveno vždy v 1. pádě. Vyslovená promluva jde do rozpoznávače řeči a vrací rozpoznaný textový přepis této promluvy. Tento text je následně předzpracován a pokračuje dále s referenční textem daného slova do vyhodnocení. Jelikož uživatel nemusí vyslovit pouze samostatné slovo, ale i slovo ve větě, které navíc nemusí být vysloveno správně, musí být nejprve z předzpracované textové promluvy vyhledáno nejvíce odpovídající slovo. Slovo se hledá pomocí upravené Levenshteinovy metody, která podporuje libovolný začátek a konec textového řetězce. Výsledkem automatické kontroly je číslo, které vyjadřuje procentuální shodu vysloveného nalezeného slova s referenčním popisem.



Obrázek 7.6: Průběh kontroly výslovnosti jednoho slova (obrázku)

Při běžném používání aplikace se bude kontrolovat více než jedno slovo za sebou. Tento průběh kontroly je znázorněn na Obrázku 7.7. Proces začíná prezentací prvního slova v podobě obrázku, následuje kontrola výslovnosti, jejíž průběh je znázorněný na předchozím Obrázku 7.6. Tento cyklus, předložení a následná kontrola výslovnosti, se opakuje, dokud jsou k dispozici další slova (obrázky). Nalezená slova a jejich procentuální shoda s referenčním textem je průběžně ukládána a na závěr je použita při celkovém vyhodnocení. Toto vyhodnocení je vypočítáno jako průměr dílčích procentuálních shod.



Obrázek 7.7: Průběh kontroly výslovnosti více slov (obrázků)

7.3 Vyhodnocení

Při návrhu vyhodnocení se vycházelo ze závěrečných poznatků z testování rozpoznávačů v kapitole 6.5. Z těchto poznatků vyplývá, že v Levenshteinově metodě cena operace *vložení* a *odstranění* zůstává **rovna 1**, ale je potřeba upravit následující případy cen operace *substituce*:

- záměna měkkého a tvrdého i/y → **cena = 0** (*nepočítáno jako chyba*);
- záměna „u“ s čárkou a s kroužkem ú/ů → **cena = 0** (*nepočítáno jako chyba*);
- záměna párových souhlásek (p-b, v-f, t-d, h-ch, atd.) = → **cena = 0.1**;
- snížení chyb v důsledku chybějící diakritiky (a/á, e/é, atd.). → **cena = 0.5**.

Všechny ostatní případy *substituce* zůstávají **rovny 1**.

8 Realizace webové apliace

V této kapitole bude prezentována realizace webové aplikace určené k automatické kontrole výslovnosti. Budou popsány kroky, které byly provedeny během vývojového procesu, od výběru technologií po finální implementaci. Proces vývoje zahrnoval několik důležitých fází, včetně realizace uživatelského rozhraní, integrace řečových technologií a vytváření backendových služeb, které zajišťují zpracování a vyhodnocování dat. Speciální důraz byl kladen na možnost jednoduchého přidávání nových slov (obrázků) do aplikace, což umožňuje její snadné rozšiřování. Popis použitých technologií a implementačních rozhodnutí bude proveden v dalších podkapitolách.

8.1 Použité technologie

Při realizace aplikace jsou využity technologie popsané v kapitole 4. Integrace řečových modulů, jako jsou ASR a TTS, je realizována prostřednictvím interní platformy SpeechCloud, která je popsána v kapitole 5. Její struktura, využitá i v této práci, se skládá ze tří hlavních komponent:

1. **Uživatelské rozhraní** (*SpeechCloud client*),
2. **Dialogový manažer** (*Dialog managerWebSocket server*),
3. **SpeechCloud** (*SpeechCloud API server*).

Grafické uživatelské rozhraní (GUI) vychází z připravené šablony ThemeSINE[8], která využívá framework Bootstrap[9]. Bootstrap usnadňuje stylování stránek a zajišťuje jejich responzivitu. Tato šablona slouží jako základ, na kterém je grafický design dále rozvíjen s využitím vlastních CSS stylů. Dynamiku stránky pak zajišťuje JavaScript s knihovnou jQuery[10].

Dialogový manažer (DM) je implementován pomocí asynchronního programování, což mu umožňuje obsluhovat řečové technologie a současně zajišťovat funkčnost uživatelského rozhraní. DM je realizován v programovacím jazyce Python.

SpeechCloud (SC) využívá pro rozpoznávání řeči (ASR) model **NO-LM**, jenž se ukázal v kapitole o testování rozpoznávačů řeči jako nejvíce vhodný pro účely automatické kontroly výslovnosti. Konkrétně se jedná o model wav2vec 2.0, který využívá grafémový přístup bez přidaného jazykového modelu na výstupu.

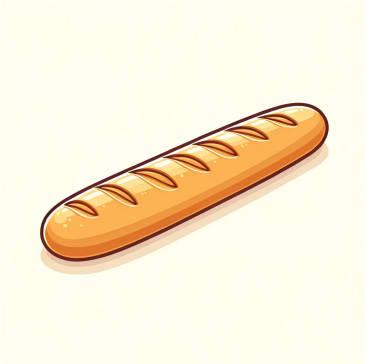
[8] `https://themesine.com`
[9] `https://getbootstrap.com`
[10] `https://jquery.com`

Generování obrázků, které fungují jako vizuální předlohy pro slova určená k výslovnosti, je prováděno pomocí technologie DALL·E 3, popsané v kapitole 4.3. Tyto obrázky jsou vytvářeny z textových popisů, tzv. *promptů*, a mají jednoduchý, kreslený styl, aby byly pro uživatele snadno rozpoznatelné. Generování těchto obrázků probíhá podle následujícího *promtu*:

Generate me an image of _____, create in a simple, cartoon-like style suitable for educational purposes.

Vygeneruj mi obrázek _____, tvoř v jednoduchém, kresleném stylu vhodném pro vzdělávací účely.

Na Obrázcích 8.1, 8.2 a 8.3 jsou výsledky generování pro slova *bageta*, *sůl* a *ponožka*.



Obrázek 8.1: Bageta

Obrázek 8.2: Sůl

Obrázek 8.3: Ponožka

8.1.1 Komunikace

Aplikace je rozčleněna do tří hlavních komponent, které mezi sebou musí spolupracovat, což vyžaduje systém komunikace. Komunikační proces je zajištěn různými technologiemi v závislosti na funkci každé komponenty. Dialogový manažer používá Websocket server pro správu komunikace v reálném čase, zatímco uživatelské rozhraní komunikuje skrze SpeechCloud clienta, který je implementován v JavaScriptu. Tyto dvě části jsou propojeny prostřednictvím platformy SpeechCloud, kde veškerá data procházejí skrze SpeechCloud API server. Tento server nejenže zprostředkovává data, ale také interaguje s moduly pro zpracování řeči (ASR a TTS).

Pro výměnu řídících zpráv je používán formát JSON, který je ideální pro strukturované datové výměny přes protokol Websocket, jak je diskutováno v kapitole 4.5. Pro přenos řečových signálů, ať už se jedná o řeč uživatele nebo syntetizovanou řeč, jsou využívány protokoly SIP/RTP, což umožňuje streamování mediálního obsahu.

V následující části bude uveden příklad komunikace komponent během kontroly výslovnosti, bez aktivované hlasové nápovědy. Pro lepší pochopení složitosti této komunikace je situace znázorněna také graficky na Obrázku 8.4. Komunikace je rozdělena do následujících bodů:

(1) Uživatel klikne na tlačítko *Start*, které pošle řídící zprávu (požadavek o aktivaci ASR) přes SC API server do DM.

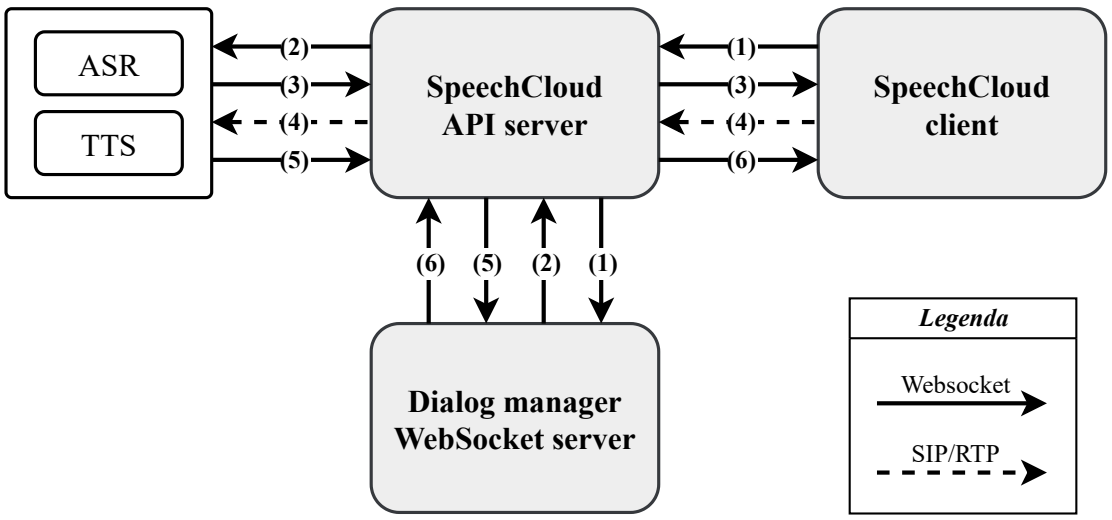
(2) DM pošle řídící zprávu do SC API serveru, který aktivuje modul ASR.

(3) Po aktivaci modul ASR vygeneruje událost o tom, že je aktivní a pošle zpět přes SC API server k SC clientovi.

(4) Uživatel vysloví slovo na obrázku a zvuková stopa se odešle přes SC API server do modulu ASR.

(5) Modul ASR rozpozná příchozí řeč a textovou podobu pošle přes SC API server do DM.

(6) DM z rozpoznané textové podoby nalezne vyslovené slovo, provede vyhodnocení a výsledky odešle přes SC API server k SC clientovi, který je prezentuje uživateli.



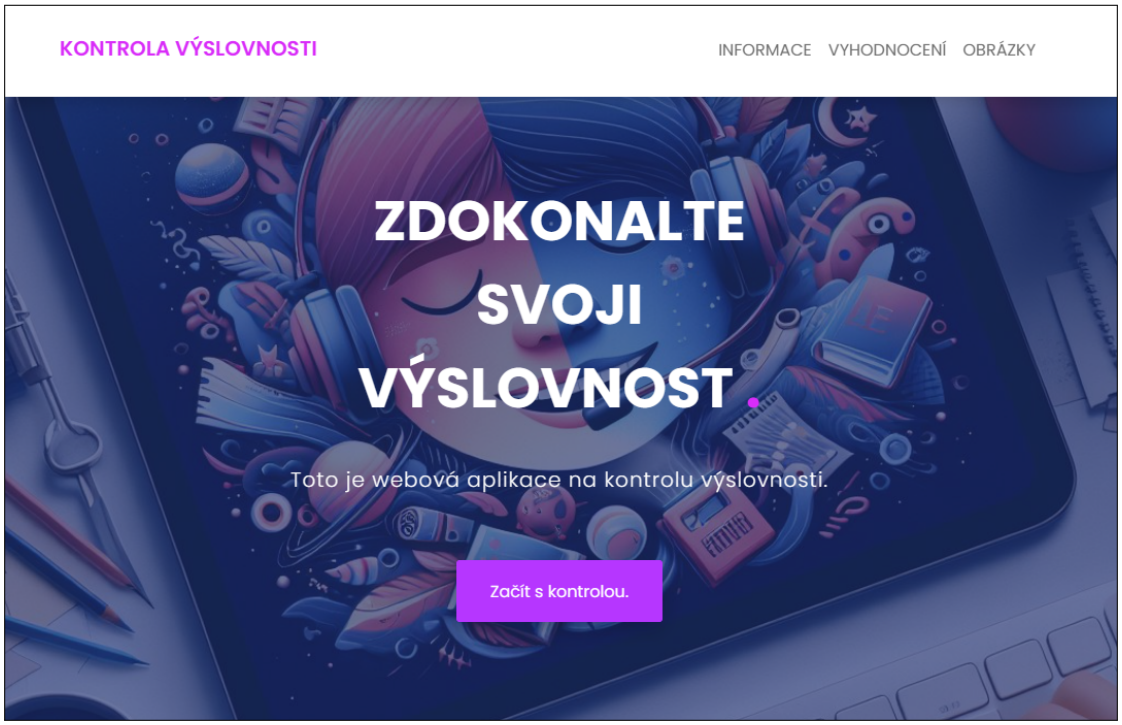
Obrázek 8.4: Ukázka průběhu komunikace

8.2 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní bylo realizováno podle návrhu uvedeného v kapitole 7.1. Skládá se ze dvou samostatných HTML stránek: úvodní stránky a aplikace.

8.2.1 Úvodní stránka

Začátek úvodní stránky, zobrazen na Obrázku 8.5, slouží jako uvítací část před vstupem do aplikace pro kontrolu výslovnosti. V pozadí je obrázek vygenerovaný pomocí umělé inteligence[11]. V popředí je pak uvítací heslo a tlačítko, které uživatele přesměruje do samotné aplikace. Pod uvítací části jsou pak dále *informace* o aplikaci, ukázky *vyhodnocení* výslovnosti a přehled vybraných *obrázků*.



Obrázek 8.5: Ukázka úvodní stránky na počítači

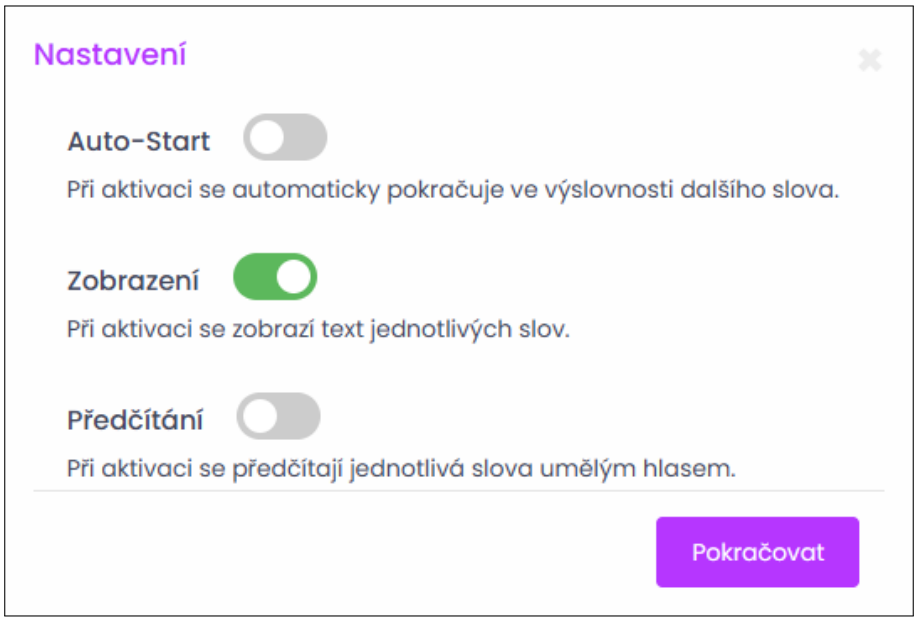
8.2.2 Aplikace

Ihned po vstupu do aplikace se objeví několik *pop-up oken*, která umožňují výběr verze, nastavení aplikace, poskytují upozornění a doporučení, a obsahují instrukce. Po instrukcích přechází aplikace k dialogu kontroly výslovnosti, který je zakončen závěrečným celkovým vyhodnocením. Podrobný popis a obsah těchto pop-up oken bude představen v následující části této práce.

[11] Promt: „*Generate me a welcome image for website with automatic pronunciation evaluation.*“

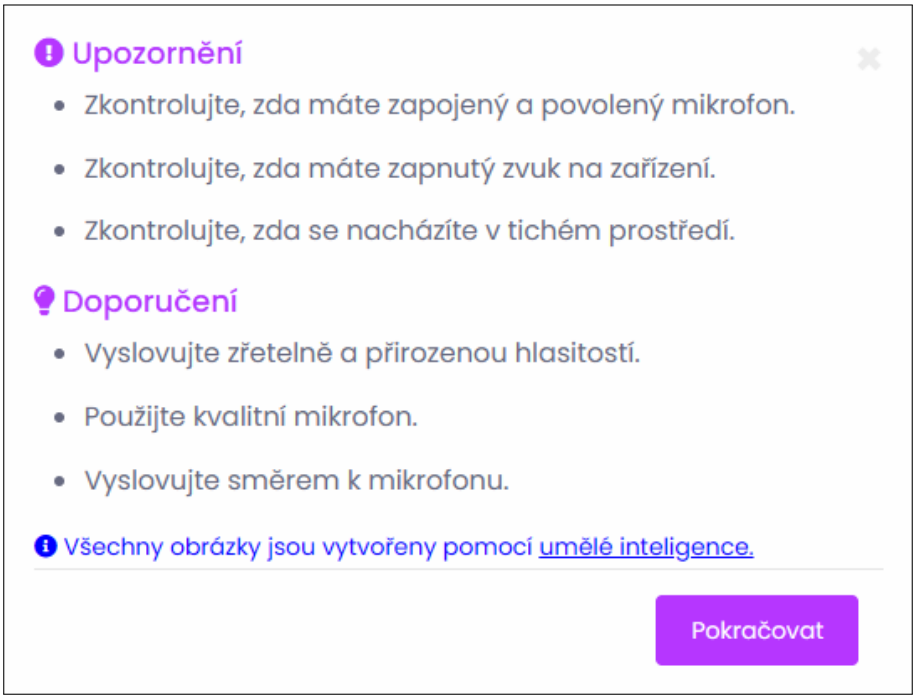
Výběr verze:
Testovací verze slouží k účelům vývoje aplikace, je zde postupně představeno vždy 15 stejných obrázků, které se v každé relaci zobrazí ve stejném pořadí. V produkční verzi je při každém spuštění vybráno náhodně n obrázků (slov), což umožňuje uživateli trénovat výslovnost na různých slovech. Počet obrázků n se nastavuje v konfiguračním souboru aplikace na straně serveru.

Nastavení:
Aplikace poskytuje několik možností nastavení, které lze upravit v pop-up okně při spuštění (viz Obrázek 8.6) nebo kdykoliv později v hlavičce aplikace. První možnost nastavení je *Auto-Start*, který umožňuje automatické pokračování ve výslovnosti dalšího slova bez nutnosti manuálního zahájení. Druhou možností nastavení aplikace je *Zobrazení*, které ovládá, zda bude nápověda v podobě textového přepisu slova nad obrázkem viditelná. Pokud je tato funkce vypnuta, text zůstává rozmazaný, naopak zapnutí zase zaručuje jeho čisté zobrazení. Pro jednorázovou nápovědu stačí kliknout na rozmazané slovo, aby se zobrazilo ostře, při dalším načtení obrázku se zobrazí tento text opět rozmazaně. Poslední třetí možností je *Předčítání*, což je hlasová nápověda. Před každou výslovností se syntetizuje dané slovo, čímž uživatel získává příklad vzorové výslovnosti.



Obrázek 8.6: Možnosti nastavení aplikace - *pop-up okno*

Upozornění a doporučení:
Po výběru verze se zobrazí okno (viz Obrázek 8.7). Toto okno uživatele upozorňuje, že je nutné zkontrolovat zapojení mikrofonu a zároveň jeho povolení ve webovém prohlížeči. Dále upozorňuje na potřebu ověřit si úroveň zvuku na zařízení, aby si mohl přehrát například hlasovou nápovědu. Doporučuje se mluvit zřetelně a přirozenou hlasitostí. Uživatelé by měli také provádět kontroly v tichém prostředí, používat kvalitní mikrofon a mluvit směrem k mikrofonu pro co nejlepší zachycení řeči. Na konci tohoto okna je pak informace o tom, že použité obrázky v aplikace jsou vytvořeny pomocí umělé inteligence.

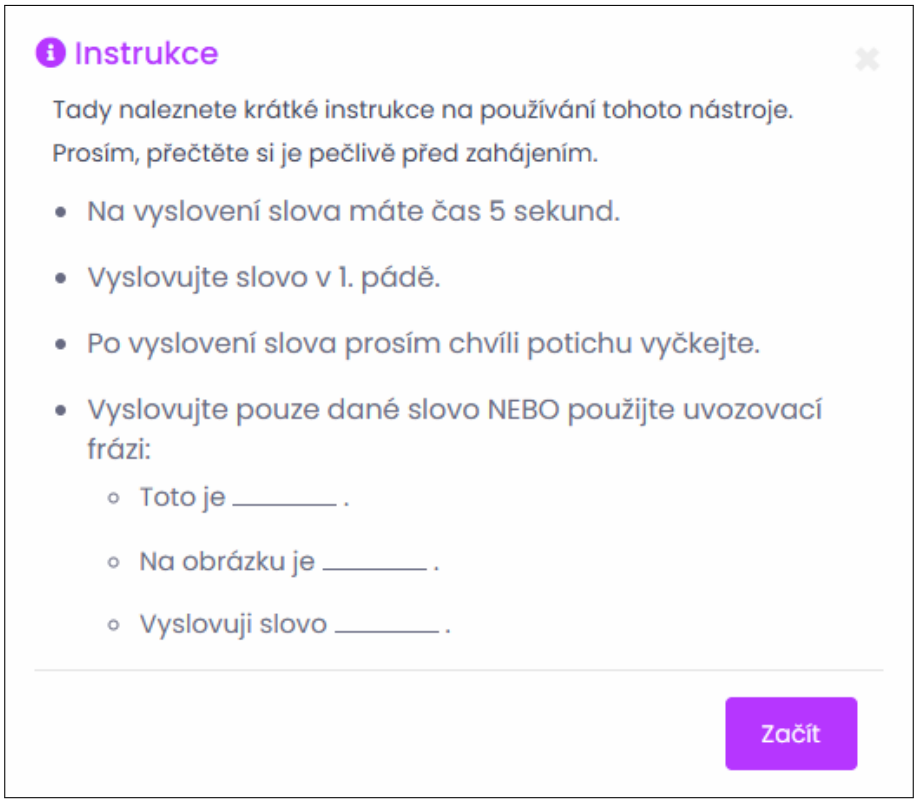


Obrázek 8.7: Upozornění a doporučení aplikace - *pop-up okno*

Instrukce:
Posledním oknem před vstupem do aplikace je okno s instrukcemi (viz Obrázek 8.8). Tyto instrukce uživatele seznamují s průběhem kontroly. Uživatel má na vyslovení slova k dispozici 5 sekund, během kterých by měl být kromě výslovnosti slova v ideálním případě potichu. V tomto časovém okně by měl uživatel vyslovit dané slovo samostatně nebo použít některou z doporučených uvozovacích frází:

- Toto je _____.
- Na obrázku _____.
- Vyslovuji slovo _____.

Tyto fráze slouží pouze jako návrh. Uživatel však může slovo použít v libovolné větě, musí ale dodržet jeho výslovnost v 1. pádě. Použití frází může být prospěšné pro správnou výslovnost a v některých případech to zabraňuje problémům s mikrofony, které někdy nemusí správně zachytit začátek slova, což by mohlo vést k chybnému vyhodnocení.



Obrázek 8.8: Instrukce aplikace - *pop-up okno*

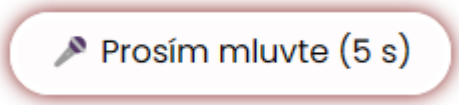
HLAVNÍ ČÁST APLIKACE:
Po instrukcích se uživatel dostává do hlavní části aplikace, která je realizována podle návrhu v kapitole 7.1.2. Tato část je rozdělena do dvou sekcí. První z nich zahrnuje kontrolu výslovnosti a druhá se věnuje závěrečnému vyhodnocení.

Na Obrázku 8.9 je zobrazena aktivovaná textová nápověda, kde je slovo vidět ostře. Naopak na Obrázku 8.10 je textová nápověda vypnuta a slovo je rozmazané.

Obrázek 8.9: Zapnuté *‚Zobrazení‘*

Obrázek 8.10: Vypnuté *‚Zobrazení‘*

Během kontroly výslovnosti jsou na stránce implementovány různé indikační prvky. Mezi ně patří indikátor úrovně zvukového signálu (viz Obrázek 8.12), indikace probíhající syntézy řeči nebo rozpoznávání řeči (viz Obrázek 8.11) a informace o probíhajícím zpracování, která uživatele vyzývá k vyčkání na vyhodnocení.

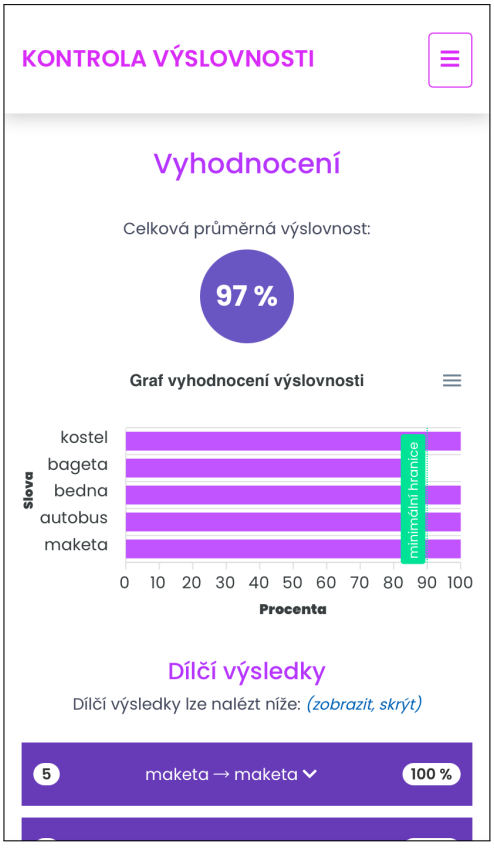


Obrázek 8.11: Indikace rozpoznávání řeči

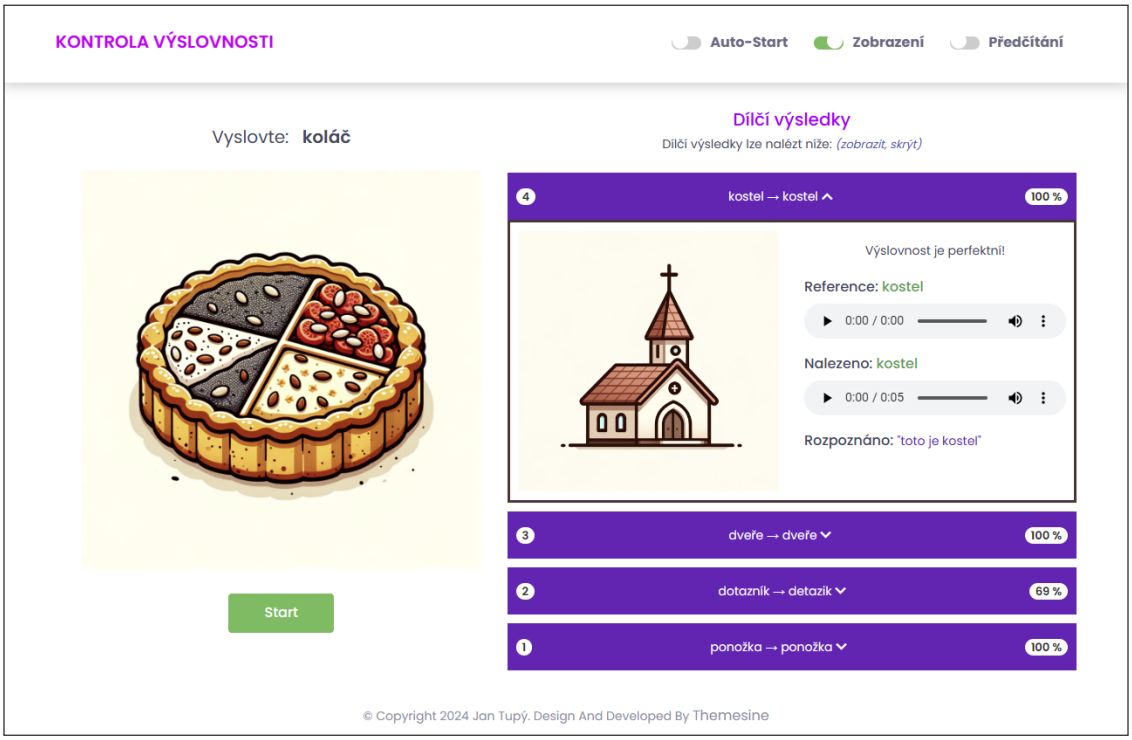
Obrázek 8.12: Indikace úrovně signálu zvuku

Na Obrázcích 8.13 a 8.14 jsou ukázky kontroly výslovnosti a závěrečného vyhodnocení v mobilní verzi webové aplikace. Pro srovnání, Obrázky 8.15 a 8.16 ilustrují tentýž proces na desktopové verzi webové stránky.

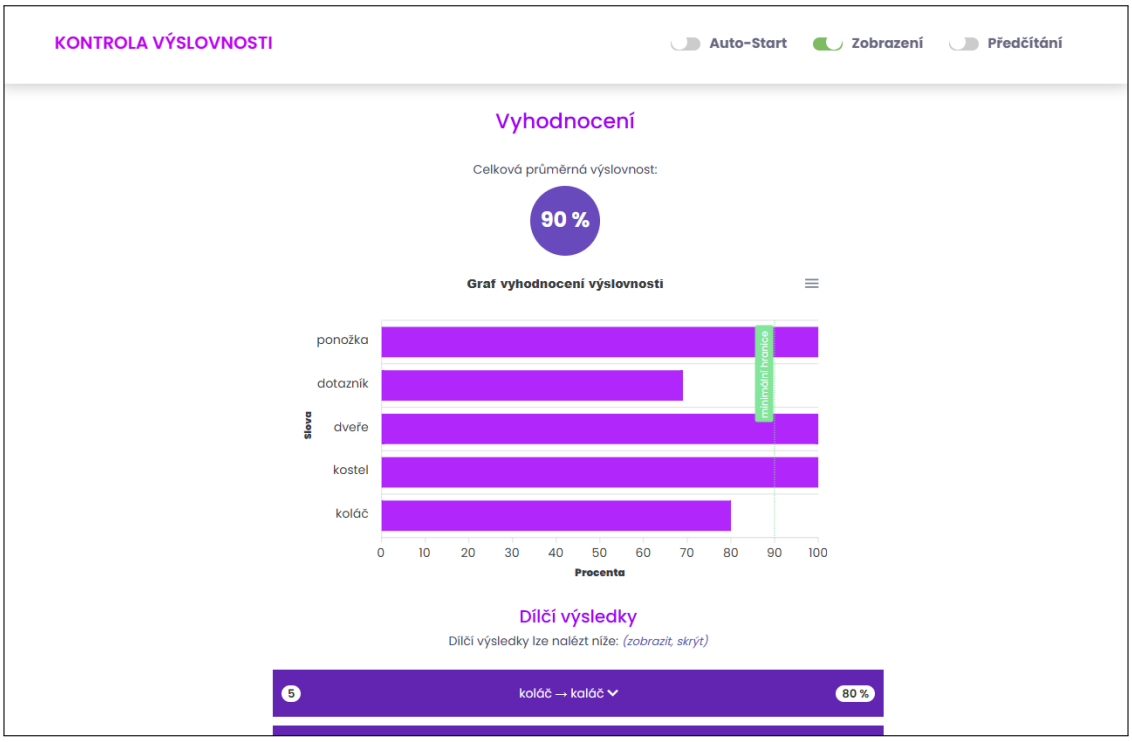
Obrázek 8.13: Kontrola výslovnosti na mobilním zařízení



Obrázek 8.14: Závěrečné vyhodnocení na mobilním zařízení



Obrázek 8.15: Kontrola výslovnosti na počítači

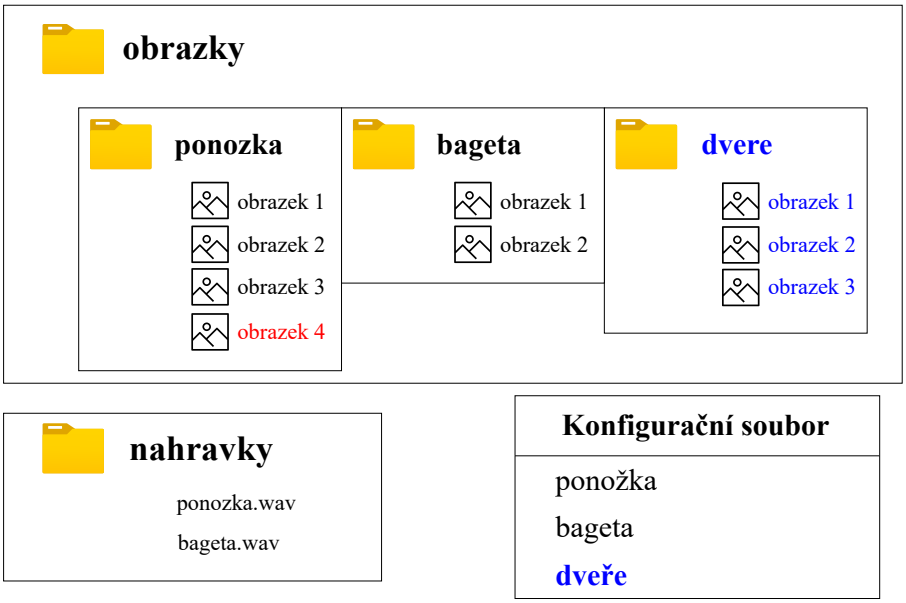


Obrázek 8.16: Závěrečné vyhodnocení na počítači

8.3 Server a konfigurace

Na serveru, kde běží dialogový manažer, jsou uložena veškerá data nezbytná pro chod aplikace. Obrázky jsou systematicky uložené ve složkách pojmenovaných podle příslušných slov bez diakritiky. Každá složka zpravidla obsahuje několik variant těchto obrázků. Kromě toho jsou na serveru uloženy audio nahrávky, které poskytují vzorovou výslovnost slov. Tyto nahrávky jsou generovány syntézou řeči předem při spouštění DM, aby nebylo nutné je generovat znovu pro každou novou relaci (session).

Přidávání nových slov do aplikace je velmi jednoduchý proces. Stačí přidat obrázky do příslušné složky a zapsat referenční text tohoto slova do konfiguračního souboru. Obrázky jsou ukládány do složek podle názvu slova bez diakritiky. Názvy jednotlivých obrázků v příslušných složkách mohou být libovolné. Přidáním tohoto slova do seznamu v konfiguračním souboru se slovo automaticky přidá do dané aplikace po restartu DM. Pro lepší představivost je vhodné uvést příklad, vycházející z Obrázku 8.17.



Obrázek 8.17: Ukázka přidávání nových slov do aplikace

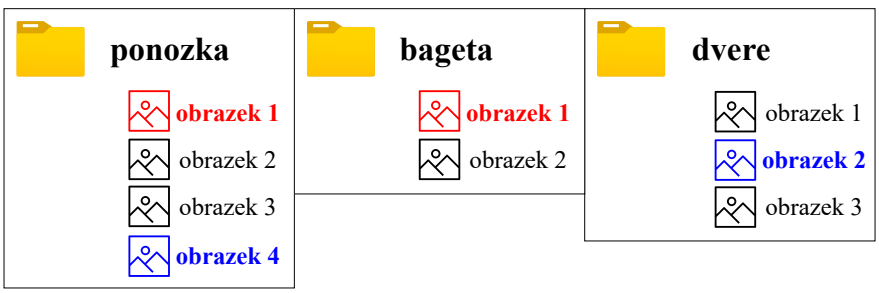
V aplikaci jsou již používány pro kontrolu výslovnosti dvě slova: ponožka a bageta. Tyto slova jsou zapsána v konfiguračním souboru a odpovídající obrázky jsou uloženy ve složkách pojmenovaných **ponozka** a **bageta**, bez použití diakritiky. K těmto slovům existují také audio nahrávky, které byly vytvořeny pomocí syntézy řeči. Pro přidání čtvrtého obrázku slova *ponožka* (znázorněné červeně) stačí obrázek umístit pouze do již existující složky **ponozka** a obrázek se automaticky zahrne

po restartování DM do možného výběru při kontrole výslovnosti. Pro přidání nového třetího slova dveře (znázorněné modře) je potřeba vytvořit novou složku **dvere**, do které se umístí jednotlivé obrázky. Aby bylo slovo zařazeno do kontroly výslovnosti, musí být také přidáno do konfiguračního souboru. Po restartu DM se znovu načtou všechna slova z konfiguračního souboru a zkontroluje se, zda existují odpovídající audio nahrávky. V tomto případě chybí nahrávka pro nově přidané slovo *dveře*, a proto bude generována nová nahrávka pomocí syntézy řeči a následně uložena k ostatním nahrávkám.

V konfiguračním souboru je možné **změnit konkrétní hlas** pro syntézu. Pro tyto účely je k dispozici možnost automatického **přegenerování referenčních nahrávek**. Pokud tato možnost není aktivována, DM při každém spouštění kontroluje existenci všech potřebných audio nahrávek a generuje pouze ty, které chybí. Dále konfigurační soubor umožňuje nastavit **timeout**, tedy maximální dobu, po kterou má uživatel čas vyslovit slovo ve webové aplikaci.

Pomocí konfiguračního souboru lze také nastavit výběr obrázků, jež se zobrazují ve webové aplikaci. Jedním z možných nastavení je **výběr počtu *n* slov**, která se mají zobrazit. Další možností je **náhodného výběru slov**. Například při nastavení $n = 2$ se mohou pro jednu relaci náhodně vybrat slova *bageta* a *dveře*, zatímco pro jinou relaci *dveře* a *ponožka*. Pokud je náhodný výběr vypnutý, vybírají se první dvě slova v přesném pořadí, jak jsou zapsána v konfiguračním souboru, což v tomto případě znamená *ponožka* a *bageta*.

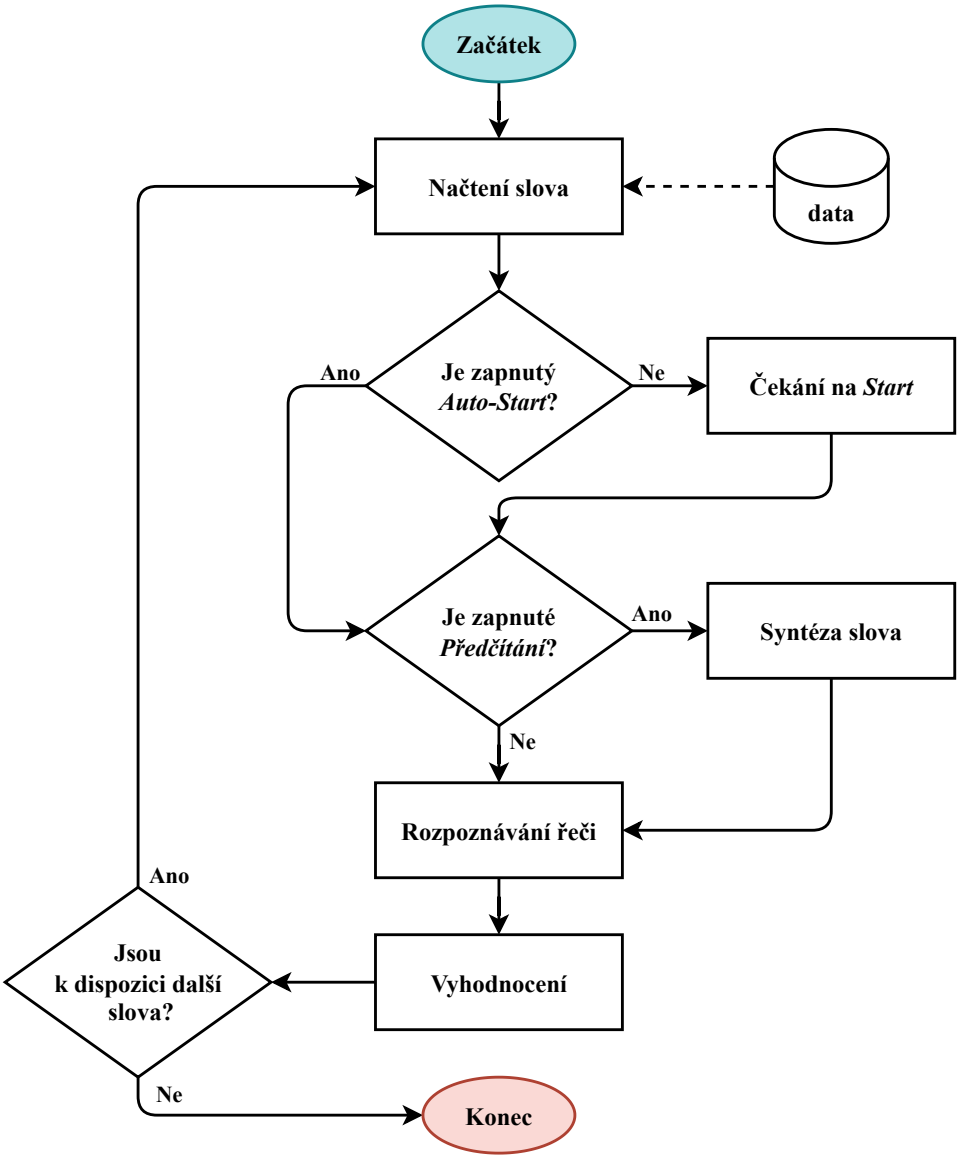
Poslední možností je **náhodný výběr obrázků** (viz ilustrace na Obrázku 8.18). Pro slova *ponožka* a *bageta* není aktivován náhodný výběr a vybere se tedy vždy první obrázek v dané složce (znázorněné červeně). U slov *ponožka* a *dveře* je zapnutý náhodný výběr obrázků (znázorněno modře).



Obrázek 8.18: Ukázka náhodného výběru obrázků

8.4 Průběh aplikace

Průběh aplikace je realizován podle návrhu z kapitoly 7.2. Průběh kontroly výslovnosti, včetně uvažování možností nastavení *Předčítání* a *Auto-Start*, je znázorněn na vývojovém diagramu na Obrázku 8.19. Dialog začíná načtením slova a předložením obrázku uživateli. Uživatel má buď zapnutý *Auto-Start*, a tak se kontrola spustí automaticky, nebo musí kontrolu spustit manuálně kliknutím na tlačítko. V případě aktivované hlasové nápovědy se dané slovo syntetizuje uživateli. Poté probíhá rozpoznávání řeči, během kterého uživatel vyslovuje slovo. Následně probíhá vyhodnocení výslovnosti. Pokud jsou k dispozici další slova, dialog pokračuje, jinak se kontrola ukončí.



Obrázek 8.19: Vývojový diagram průběhu kontroly výslovnosti

8.5 Vyhodnocení

Před samotným vyhodnocením je nezbytné provést předzpracování textové promluvy. Tento proces zahrnuje několik kroků, které jsou zaměřeny na úpravu a normalizaci vstupního textu. Jednotlivé kroky předzpracování jsou ilustrovány na příkladu promluvy „*Toto je asi ČR.*“. Kroky zahrnují:

1. Převedení textu na malá písmena. → „*toto je asi čr.*“
2. Odstranění speciálních znaků. → „*toto je asi čr*“
3. Odstranění balastních slov. → „~~*toto je*~~ *asi čr*“

Po dokončení předzpracování je výsledná textová promluva „*asi čr*“. Tento řetězec se následně porovnává s referenčním textem, v tomto případě slovem „*hasič*“, pomocí upravené Levenshteinovy metody. Tato modifikovaná metoda umožňuje zahrnout libovolný začátek a konec textového řetězce při výpočtu. Nejlepší nalezená shoda pro podřetězec „*asi č*“ vyžaduje pouze jedno vložení na začátku a jedno odstranění mezery, což dává Levenshteinovu vzdálenost $LD = 2$ (cena operací *vložení* a *odstranění* je 1). Procentuální shoda je poté vypočítána pomocí vzorce (8) z kapitoly 3.3:

$$LR = 1 - \frac{levenshtein\_distance(s1, s2)}{max(len(s1), len(s2))} = 1 - \frac{2}{5} = 0.6 \ , \tag{12}$$

což znamená, že slova „*asi čr*“ a „*hasič*“ mají 60 % shodu.

Pro operaci substituce byla v kapitole 7.3 pro některé záměny navržena odlišná cena než 1, což lépe odráží skutečné vyhodnocení. Ceny za záměnu jednotlivých znaků jsou implementovány pomocí ztrátové matice, která je zobrazena na Obrázku 8.20.

Dalším příkladem vyhodnocení bude referenční slovo „*kostel*“ a k němu rozpoznaná promluva „*kestel*“. V tomto případě dochází k záměně (substituci) jednoho znaku *e* místo *o*. Tato substituce je ve ztrátové matice standardně hodnocena hodnotou 1, což odpovídá Levenshteinově vzdálenosti pro tento případ. Výsledný Levenshteinův poměr (LR) je pak:

$$LR = 1 - \frac{levenshtein\_distance(s1, s2)}{max(len(s1), len(s2))} = 1 - \frac{1}{6} \doteq 0.83 \ , \tag{13}$$

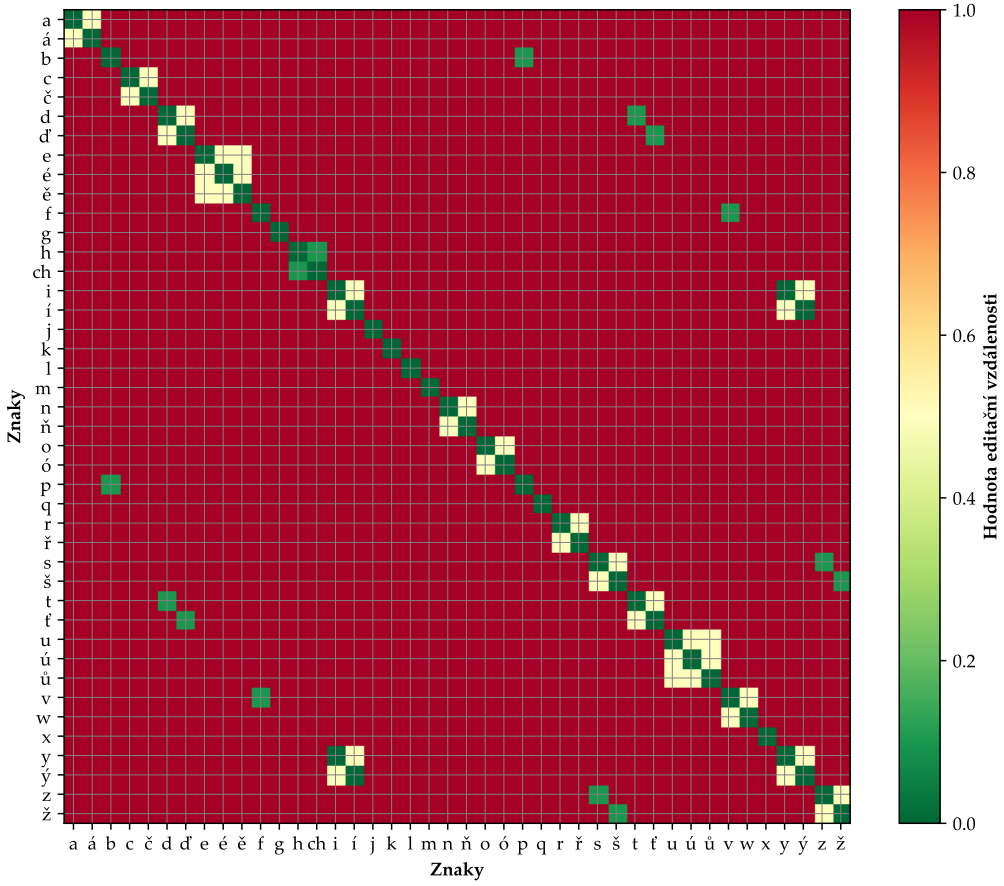
což odpovídá přibližně 83 % shodě.

V případě rozpoznané promluvy „*kóstel*“, kde chyba spočívá pouze v chybějící diakritice (*ó* místo *o*), je substituce ohodnocena nižší cenou 0.5 ve ztrátové matici. Výsledná LD zůstává 0.5. Poměr pro tento příklad je následně:

$$LR = 1 - \frac{levenshtein\_distance(s1, s2)}{max(len(s1), len(s2))} = 1 - \frac{0.5}{6} \doteq 0.92 \ , \tag{14}$$

což představuje přibližnou shodu 92 %.

Přestože obě varianty slova („*kestel*“ a „*kóstel*“) obsahují chybu pouze v jednom znaku, celkové procentuální hodnocení se liší. Tento přístup poskytuje jemnější detail pro hodnocení výslovnosti, což umožňuje přesnější posouzení míry shody.



Obrázek 8.20: Ztrátová matice - ceny substituce

Při implementaci je nutná výjimka pro znak ‚*ch*‘, který je v češtině považován za jeden znak, i když se programátorsky se skládá ze dvou znaků ‚*c*‘ a ‚*h*‘. Aby bylo možné tento problém obejít a správně vypočítat LD, je ‚*ch*‘ dočasně nahrazováno symbolem ‚*$*‘, což umožňuje považování ‚*ch*‘ za jednotlivý znak. Po dokončení výpočtu je symbol ‚*$*‘ znovu nahrazen zpět na ‚*ch*‘.

9 Testování webové aplikace

Testování je rozděleno do dvou částí. První část se zaměřuje na testování samotné webové aplikace, které provedlo 20 uživatelů. Následně druhá část zahrnuje dotazníkové šetření, jež bylo provedeno se stejnými uživateli.

9.1 Testování

Testování webové aplikace probíhalo v rámci „Testovací verze“, kde bylo připraveno 15 slov, které jednotliví uživatelé museli postupně vyslovit. Výběr těchto 15 slov byl proveden na základě 5 výchozích slov **original**. Od těchto slov byly následně odvozeny vždy 3 skupiny slov **reference**:

1. **Stejné slovo**: Slovo je shodné se slovem **original** (např. *ponožka*).
2. **Podobné slovo**: Slovo je podobné slovu **original** (např. *stonožka*).
3. **Jiné slovo**: Slovo má minimální nebo žádnou shodu se slovem **original** (např. *telefon*).

V Tabulce 9.1 jsou zobrazena všechna slova vybraná pro testování. Je zde uvedeno celkem 5 skupin, kde každá skupina obsahuje výchozí slovo **original** a k němu přiřazená **shodná**, **podobná** a **jiná slova**.

Tabulka 9.1: Přehled testovaných slov - *(shodné, podobné, jiné slovo)*

Original	Reference
ponožka	ponožka
	stonožka
	telefon
kostel	kostel
	postel
	židle
bedna	bedna
	kedna
	dveře
bageta	bageta
	maketa
	koláč
stůl	stůl
	sůl
	prase

Na následujících Tabulkách 9.2, 9.3, 9.4 a 9.5 jsou výsledky testování prvních 4 uživatelů. Tyto tabulky ukazují data získaná během testování. Každý uživatel nejprve vyslovil slovo (**reference**) odvozené od výchozího slova **original**. Rozpoznávač řeči vrátí **rozpoznanou promluvu**, ze která se pomocí upravené Levenshteinovy metody získá **nalezené slovo**. Toto slovo by mělo být v ideálním případě shodné (tj. 100 % shoda) s **referencí**. Je důležité, zda uživatel slovo vyslovil správně, a také, zda bylo správně rozpoznáno rozpoznávačem řeči. Tato shoda je v tabulkách označena jako **LR reference**. Poslední sloupec v těchto tabulkách je označován jako **LR original**, což vyjadřuje procentuální shodu **nalezeného slova** se slovem **original**. Tento údaj je vždy pro „stejné slovo“ označen pomlčkou, protože odpovídá údaji **LR reference**. Pro „podobná slova“ by tento údaj měl být ve vysoký, zatímco pro „jiná slova“ by měl být tento údaj nízký.

V Tabulce 9.2 byla úspěšnost rozpoznání slov téměř ve všech případech 100 %. Pouze u slova „koláč“, které bylo rozpoznáno jako „kol“, byla shoda pouze 60 %. Podle audio nahrávek byl tento neúplný výsledek způsoben tím, že uživatel nestihl slovo vyslovit celé během pětisekundového limitu. Jeho průměrná úspěšnost výslovnosti dosahuje **97.3 %**. Uživatel 2 dosáhl v Tabulce 9.3 dokonalé výslovnosti **100 %** ve všech případech. V Tabulce 9.4 jsou některé chyby v rozpoznaných slovech, jako například „položka“ místo „ponožka“ nebo „stomoška“ místo „stonožka“. Celkově má úspěšnost **94.6 %**. U uživatele 4 se v Tabulce 9.5 často objevily chyby na začátku slov, jako „vedna“ místo „bedna“, „hostel“ místo „postel“ a „vegeta“ místo „bageta“. V tomto případě by uživatelovi mohla pomoct uvozovací fráze, např. „*Toto je* ____“. Jeho průměrná shoda s referencí činí **94.9 %**.”

V Tabulkách 9.6, 9.7 a 9.8 jsou výsledky rozděleny do tabulek podle **referencí** (shodné, podobné a jiné slovo). Údaj $\mathbf{LR}_{\mathrm{ASR}}$ vyjadřuje průměr shod **nalezených slov** se slovem **original** všech 20 uživatelů. Údaj $\mathbf{LR}_{\mathrm{reference}}$ vyjadřuje shodu **reference** a slova **original**. K tomuto údaji by se měla shoda $\mathbf{LR}_{\mathrm{ASR}}$ v ideálním případě co nejvíce přiblížit. Posledním údajem v tabulce je $\mathbf{LR}_{\mathrm{diff}}$, který vyjadřuje právě rozdíl těchto údajů $\mathbf{LR}_{\mathrm{reference}} - \mathbf{LR}_{\mathrm{ASR}}$. Tyto tabulky poskytují užitečný vhled do toho, jak dobře systém rozpoznává slova v kontextu jejich podobnosti s původním slovem. Je patrné, že u slov shodných s originálem dosahuje průměrná úspěšnost **98.8 %**. U slov, která mají větší vizuální nebo zvukovou podobnost s původním slovem, jsou hodnoty stále relativně vysoké, a to **73.6 %**. Naopak u slov, která se od originálu zcela liší, jsou tyto hodnoty nízké nebo dokonce nulové, což dokládá **3.6 %**.

Tabulka 9.2: Výsledky testování - **uživatel 1**

Original	Reference	Rozpoznaná promluva	Nalezené slovo	LR reference	LR original
ponožka	ponožka	ponožka	ponožka	100 %	-
	stonožka	stonožka	stonožka	100 %	75.0 %
	telefon	toto je telefon	telefon	100 %	0.0 %
kostel	kostel	tuto kostel	kostel	100 %	-
	postel	také postel	postel	100 %	83.3 %
	židle	židle	židle	100 %	16.7 %
bedna	bedna	tuto bedna	bedna	100 %	-
	kedna	kedna	kedna	100 %	80.0 %
	dveře	také dveře	dveře	100 %	0.0 %
bageta	bageta	tuto je bageta	bageta	100 %	-
	maketa	a to to maketa	maketa	100 %	66.7 %
	koláč	velikej kole	kol	60 %	0.0 %
stůl	stůl	stůl	stůl	100 %	-
	sůl	tuto sůl	sůl	100 %	75.0 %
	prase	velký prase	prase	100 %	0.0 %
Průměr:				**97.3 %**	

Tabulka 9.3: Výsledky testování - **uživatel 2**

Original	Reference	Rozpoznaná promluva	Nalezené slovo	LR reference	LR original
ponožka	ponožka	tohle je ponožka	ponožka	100 %	-
	stonožka	tohle je stonožka	stonožka	100 %	75.0 %
	telefon	tohle je telefon	telefon	100 %	0.0 %
kostel	kostel	tohle je kostel	kostel	100 %	-
	postel	tohle je postel	postel	100 %	83.3 %
	židle	tohle je židle	židle	100 %	16.7 %
bedna	bedna	tohle je bedna	bedna	100 %	-
	kedna	tohle je kedna	kedna	100 %	80.0 %
	dveře	tohle jsou dveře	dveře	100 %	0.0 %
bageta	bageta	tohle je bageta	bageta	100 %	-
	maketa	tohle je maketa	maketa	100 %	66.7 %
	koláč	tohle je koláč	koláč	100 %	0.0 %
stůl	stůl	tohle je stůl	stůl	100 %	-
	sůl	tohle je sůl	sůl	100 %	75.0 %
	prase	tohle je prase	prase	100 %	0.0 %
Průměr:				**100 %**	

Tabulka 9.4: Výsledky testování - **uživatel 3**

Original	Reference	Rozpoznaná promluva	Nalezené slovo	LR reference	LR original
ponožka	ponožka	toto je položka	položka	86 %	-
	stonožka	toto je stomoška	stomoška	86 %	61.3 %
	telefon	tohle je telefon	telefon	100 %	0.0 %
kostel	kostel	tohle je kostel	kostel	100 %	-
	postel	tohle je postel	postel	100 %	83.3 %
	židle	tohle je židle	židle	100 %	16.7 %
bedna	bedna	tohle je brdna	brdna	80 %	-
	kedna	tohle je kedna	kedna	100 %	80.0 %
	dveře	tohle soudveře	dveře	100 %	0.0 %
bageta	bageta	tohle je bageta	bageta	100 %	-
	maketa	tohle je maketa	maketa	100 %	66.7 %
	koláč	tohle je koláč	koláč	100 %	0.0 %
stůl	stůl	tohle je stůl	stůl	100 %	-
	sůl	tohle je sů	sů	67 %	50.0 %
	prase	tohle je prase	prase	100 %	0.0 %
Průměr:				**94.6 %**	

Tabulka 9.5: Výsledky testování - **uživatel 4**

Original	Reference	Rozpoznaná promluva	Nalezené slovo	LR reference	LR original
ponožka	ponožka	ponožka	ponožka	100 %	-
	stonožka	stonožka	stonožka	100 %	75.0 %
	telefon	telefon	telefon	100 %	0.0 %
kostel	kostel	kostel	kostel	100 %	-
	postel	hostel	hostel	83 %	83.3 %
	židle	židlo	židl	80 %	31.7 %
bedna	bedna	vedna	vedna	80 %	-
	kedna	kedna	kedna	100 %	80.0 %
	dveře	dveře	dveře	100 %	0.0 %
bageta	bageta	vageta	vageta	83 %	-
	maketa	maketa	maketa	100 %	66.7 %
	koláč	koláč	koláč	100 %	0.0 %
stůl	stůl	stůl	stůl	100 %	-
	sůl	sůl	sůl	100 %	75.0 %
	prase	v praze	praze	98 %	0.0 %
Průměr:				**94.9 %**	

Tabulka 9.6: Srovnání shody LR pro „stejné slovo“

Original	**Stejné slovo**	**LR_{ASR}**	**$LR_{reference}$**	**LR_{diff}**
ponožka	ponožka	97.9 %	100.0 %	-2.1 %
kostel	kostel	100.0 %	100.0 %	-0.0 %
bedna	bedna	97.0 %	100.0 %	-3.0 %
bageta	bageta	99.2 %	100.0 %	-0.8 %
stůl	stůl	100.0 %	100.0 %	-0.0 %
Průměr abs. hodnot:		**98.8 %**	**100.0 %**	**1.2 %**

Tabulka 9.7: Srovnání shody LR pro „podobné slovo“

Original	**Podobné slovo**	**LR_{ASR}**	**$LR_{reference}$**	**LR_{diff}**
ponožka	stonožka	71.5 %	75.0 %	-3.5 %
kostel	postel	84.1 %	83.3 %	0.8 %
bedna	kedna	75.0 %	80.0 %	-5.0 %
bageta	maketa	66.7 %	66.7 %	-0.0 %
stůl	sůl	70.6 %	75.0 %	-4.4 %
Průměr abs. hodnot:		**73.6 %**	**76.0 %**	**2.7 %**

Tabulka 9.8: Srovnání shody LR pro „jiné slovo“

Original	**Jiné slovo**	**LR_{ASR}**	**$LR_{reference}$**	**LR_{diff}**
ponožka	telefon	0.0 %	0.0 %	0.0 %
kostel	židle	18.2 %	16.7 %	1.5 %
bedna	dveře	0.0 %	0.0 %	0.0 %
bageta	koláč	0.0 %	0.0 %	0.0 %
stůl	prase	0.0 %	0.0 %	0.0 %
Průměr abs. hodnot:		**3.6 %**	**3.3 %**	**0.3 %**

V další části bude v Tabulce 9.9 a 9.10 jednotlivé výsledky rozpoznání a nalezených slov pro všech 20 uživatelů. První tabulka se zaměřuje na výsledky slova „sůl“ a druhá tabulka na výsledky slova „stonožka“.

První tabulka 9.9 ukazuje, že ve většině případů (15 z 20) dosáhl systém 100 % úspěšnosti rozpoznání slova „sůl“, což svědčí o vysoké účinnosti systému. Nicméně, data také ukazují, že systém může mít problémy s rozpoznáváním krátkých slov. V některých případech bylo slovo „sůl“ nesprávně rozpoznáno jako „fůl“, „osum“, „sůň“, nebo „stůl“. Zlepšení by mohlo přinést lepší uživatelova výslovnost nebo použití uvozovací fráze před slovem.

Celkový průměr přesnosti **91.3 %** je poměrně vysoký, ale přesto naznačuje, že je stále prostor pro zlepšení, zvláště pro tyto krátké slova, kde systém nesprávně interpretuje slova nebo nedostatečně rozlišovat nuance ve výslovnosti.

Tabulka 9.9: Výsledky testování slova **sůl**

Rozpoznaná promluva	**Nalezené slovo**	**LR_{find}**
sůl	sůl	100 %
fůl	fůl	67 %
osum	su	50 %
tuto sůl	sůl	100 %
sůl	sůl	100 %
sůl	sůl	100 %
sůl	sůl	100 %
sůň	sů	67 %
sůl	sůl	100 %
tohle je sů	sů	67 %
sůl	sůl	100 %
stůl	stůl	75 %
sůl	sůl	100 %
sůl	sůl	100 %
sůl	sůl	100 %
sůl	sůl	100 %
sůl	sůl	100 %
sůl	sůl	100 %
tohle je sůl	sůl	100 %
sůl	sůl	100 %
Průměr:		**91.3 %**

Druhá tabulka 9.10 ukazuje, že většina záznamů (16 z 20) dosáhla perfektní shody 100 %, což naznačuje, že slovo „stonožka“ je dobře rozpoznávané. Přesto se vyskytly i méně přesné výsledky, jako „stanožka“, „stoložka“ a „stomoška“, které, i když jsou blízké originálu, ukazují na potenciální problémy systému s rozpoznáváním slov s mírně odlišnou výslovností nebo v přítomnosti šumu v pozadí. Nejnižší shoda 61 % byla zaznamenána u slova „tomáška“. Celková průměrná shoda nalezených slov s referenčním slovem „stonožka“ činí **95.6 %**.”

Tabulka 9.10: Výsledky testování slova **stonožka**

Rozpoznaná promluva	**Nalezené slovo**	**LR_{find}**
stonožka	stonožka	100 %
stonožka	stonožka	100 %
stonožka	stonožka	100 %
stonožka	stonožka	100 %
stonožka	stonožka	100 %
stanožka	stanožka	88 %
stonožka	stonožka	100 %
stoložka	stoložka	88 %
stonožka	stonožka	100 %
toto je stomoška	stomoška	86 %
tomáška	tomáška	61 %
stonožka	stonožka	100 %
stonožka	stonožka	100 %
stonožka	stonožka	100 %
stonožka	stonožka	100 %
stonožka	stonožka	100 %
stanožka	stanožka	88 %
stonožka	stonožka	100 %
tohle je stonožka	stonožka	100 %
stonožka	stonožka	100 %
Průměr:		**95.6 %**

Tabulka 9.11 shrnuje průměrné hodnoty správné výslovnosti referenčních slov u testovaných osob. Z dat je patrné, že většina slov má vysokou míru správné výslovnosti, s průměrem **97.58 %**. To ukazuje na vysokou úroveň výslovnosti mezi testovanými osobami a správné rozpoznávání řeči.

Slova jako „telefon“, „kostel“, „dveře“, „maketa“ a „stůl“ dosáhla dokonalého skóre 100 %, což naznačuje, že tato slova jsou mezi účastníky dobře známá nebo snadno vyslovitelná. Na druhé straně, slovo „sůl“ s nejnižším skórem 91.3 % může naznačovat určitou úroveň obtížnosti s rozpoznáváním tohoto slova nebo na problémy s výslovností mezi některými testovanými osobami.

Tabulka 9.11: Průměrná výslovnost jednotlivých slov

Reference	**LR_{find}**
ponožka	97.9 %
stonožka	95.5 %
telefon	100 %
kostel	100 %
postel	97.5 %
židle	98 %
bedna	97 %
kedna	93 %
dveře	100 %
bageta	99.2 %
maketa	100 %
koláč	96.5 %
stůl	100 %
sůl	91.3 %
prase	97.9 %
Průměr:	**97.58 %**

Tabulka 9.12 prezentuje průměrné skóre výslovnosti referenčních slov mezi dvaceti testovanými osobami, s celkovým průměrným výsledkem **97,58 %**. To svědčí o celkově vysoké úrovni schopnosti správné výslovnosti mezi účastníky.

Zvláště pozoruhodné jsou výkony uživatelů 1, 7, 9, 15, 18, 19 a 20, kteří dosáhli dokonalého skóre 100 %, což naznačuje jejich schopnost přesné výslovnosti testovaných slov. Na druhé straně, uživatelé 10 a 11, zaznamenali nejnižší skóre 94.6 % a 93.8 %.

Tabulka 9.12: Průměrná výslovnost referenčních slov u testovaných osob

Testovaná osoba	**LR_{find}**
uživatel 1	97.3 %
uživatel 2	100 %
uživatel 3	94.6 %
uživatel 4	94.9 %
uživatel 5	95.1 %
uživatel 6	95.9 %
uživatel 7	100 %
uživatel 8	95.7 %
uživatel 9	100 %
uživatel 10	95.3 %
uživatel 11	93.8 %
uživatel 12	95.9 %
uživatel 13	97.7 %
uživatel 14	97.5 %
uživatel 15	100 %
uživatel 16	98.7 %
uživatel 17	99.2 %
uživatel 18	100 %
uživatel 19	100 %
uživatel 20	100 %
Průměr:	**97.58 %**

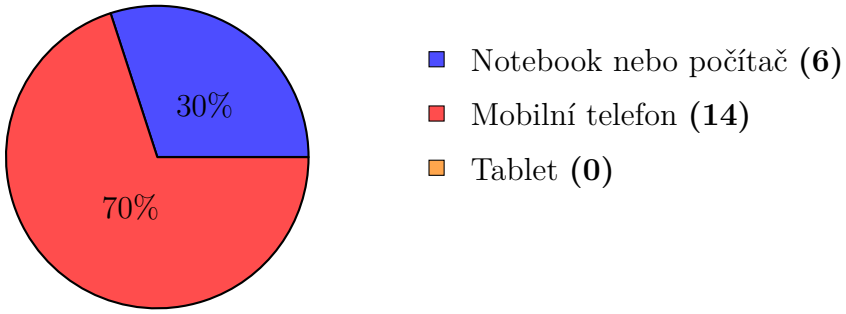
9.2 Dotazníkové šetření

V rámci praktické části diplomové práce bylo provedeno dotazníkové šetření zaměřené na získání zpětné vazby od uživatelů webové aplikace, které byla tato práce věnována. Dotazník byl distribuován mezi stejných 20 respondentů, kteří se podíleli na testování funkcionality a uživatelské přívětivosti zmíněné aplikace.

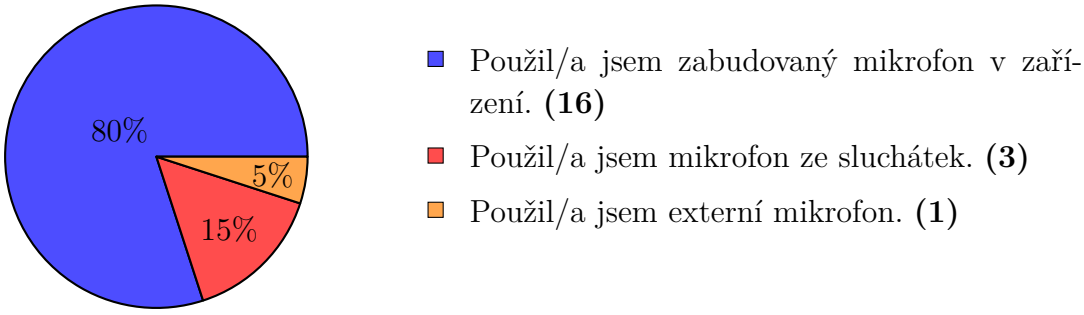
Celkově bylo v dotazníku položeno 11 otázek, které byly formulovány tak, aby odpovídaly na konkrétní aspekty použitelnosti a efektivity aplikace. Respondenti měli možnost vybírat ze zadaných odpovědí, což umožnilo kvantifikovat a efektivně analyzovat jejich zpětnou vazbu. Cílem tohoto šetření bylo nejen posoudit celkovou spokojenost uživatelů, ale také identifikovat specifické silné a slabé stránky aplikace, což přispěje k jejímu dalšímu vývoji a zlepšení.

9.2.1 Výsledky jednotlivých otázek:

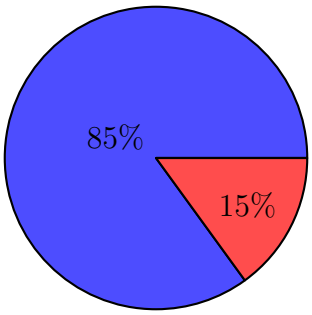
Otázka 1: Jaké jste použil/a zařízení?



Otázka 2: Jaký jste použil/a mikrofon?

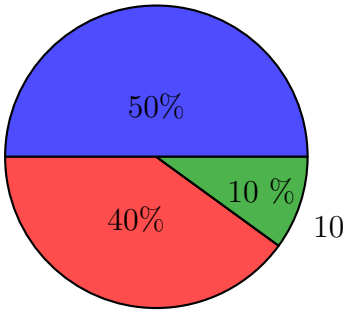


Otázka 3: V jakém jste se nacházel/a prostředí?



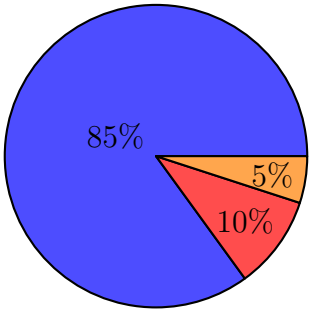
- **Tiché** - Prostředí bylo klidné bez žádných rušivých zvuků. **(17)**
- **Mírně rušivé** - V prostředí se nacházely mírně rušivé zvuky *(např. zapnutá televize).* **(3)**
- **Rušné** - V prostředí se nacházely silně rušivé zvuky *(např. hlasitá hudba nebo konverzace).* **(0)**

Otázka 4: Vyzkoušel/a jsem si aktivovat režim 'Auto-Start' a automatické 'Předčítání'?



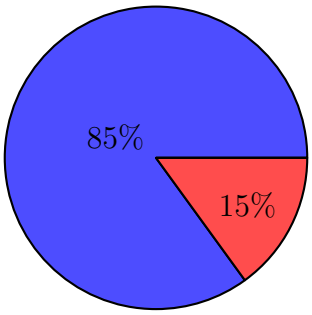
- Ne ani jedno. **(10)**
- Ano, pouze *Auto-Star.* **(8)**
- Ano, pouze *Předčítání.* **(0)**
- Ano, obě možnosti. **(2)**

Otázka 5: Používal/a jsem při vyslovování slova uvozovací frázi?
(Toto je ... , Na obrázku je ... , atd.)



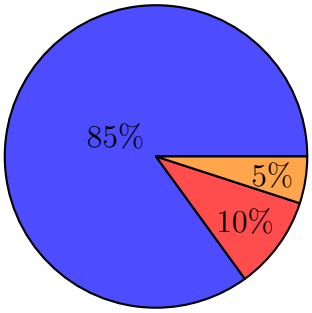
- Ne, říkal/a jsem vždy pouze samostatné slovo. **(17)**
- Ano, občas jsem používal/a uvozující frázi. **(2)**
- Ano, vždy jsem používal/a uvozující frázi. **(1)**

Otázka 6: Jaké byly úvodní informace ohledně fungování webové aplikace?



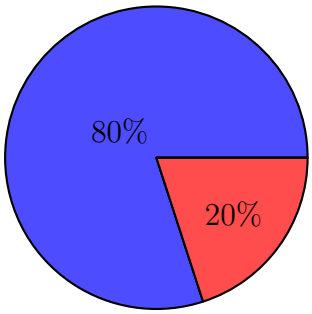
- Informace byly jasné a srozumitelné, hned jsem věděl/a, co mám dělat. **(17)**
- Informace byly složité, ale vše jsem nakonec pochopil/a. **(3)**
- Informace byly nesrozumitelné, nevěděl/a jsem, co mám dělat. **(0)**

Otázka 7: Jak byste popsali intuitivnost používání webové aplikace na kontrolu výslovnosti?



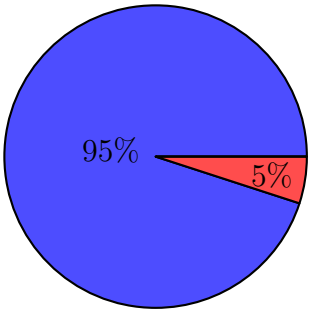
- **Intuitivní** - Intuitivně jsem chápal/a, jak používat webovou aplikaci bez potřeby podrobného návodu. **(17)**
- **Neutrální** - Některé funkce webové aplikace byly intuitivní, zatímco jiné vyžadovaly trochu zkoumání. **(2)**
- **Zmatená** - Měl/a jsem problémy porozumět způsobu fungování některých funkcí webové aplikace. **(1)**

Otázka 8: Jak byste ohodnotil/a vzhled webové aplikace na kontrolu výslovnosti?



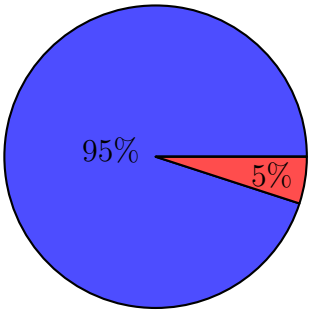
- **Příjemný** - Webová aplikace má moderní a atraktivní design podrobného návodu. **(16)**
- **Neutrální** - Vzhled webové aplikace není výrazně pozitivní ani negativní, je průměrný. **(4)**
- **Nepříjemný** - Vzhled webové aplikace není zrovna atraktivní a mohl by být vylepšen. **(0)**

Otázka 9: Jak byste hodnotil/a přehlednost webové aplikace na kontrolu výslovnosti?



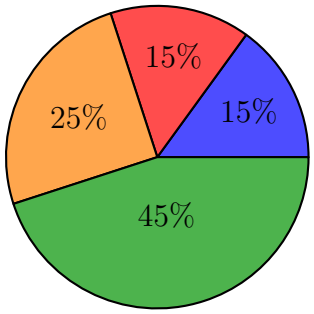
- **Přehledná** - Webová aplikace má jasnou strukturu a je jednoduché se v ní orientovat. **(19)**
- **Neutrální** - Přehlednost webové aplikace je průměrná, není zvlášť obtížné se v ní orientovat, ale ani jednoduché. **(1)**
- **Nepřehledná** - Webová aplikace má komplikovanou strukturu a je složité se v ní orientovat. **(0)**

Otázka 10: Jak byste ohodnotil/a responzivitu webové aplikace na kontrolu výslovnosti? Zobrazovala se vám webová aplikace vždy správně a jednotlivé prvky byly uspořádány přehledně?



- **Vždy ano** - Webová aplikace se zobrazovala vždy správně a její prvky byly uspořádány přehledně. **(19)**
- **Většinou ano** - Webová aplikace se výjimečně zobrazovala nesprávně nebo její prvky nebyly uspořádány přehledně. **(1)**
- **Spíše ne** - Webová aplikace se často zobrazovala nesprávně nebo její prvky nebyly uspořádány přehledně. **(0)**

Otázka 11: Uvítali byste novější a přirozenější hlas syntézy ve webové aplikaci?



- Ano, uvítal/a bych novější a přirozenější hlas syntézy. **(3)**
- Ano, ale současný hlas je pro mě dostačující. **(3)**
- Ne, jsem spokojen/a s aktuálním umělým hlasem. **(5)**
- Nelze posoudit, umělý hlas jsem na webové stránce neslyšel/a *(nepřehrál/a jsem si referenční výslovnosti ani jsem neměl/a aktivovaný režim ‚Předčítání')*. **(9)**

9.2.2 Shrnutí

V následující části je shrnutí dotazníkového šetření, které se zúčastnilo 20 respondentů.

1. Použité zařízení:
Většina respondentů (70 %) použila mobilní telefon k otestování aplikace, zatímco menší část (30 %) preferovala notebook nebo počítač. Tablety nebyly použity.

2. Použitý mikrofon:
Drtivá většina (80 %) účastníků používala vestavěný mikrofon ve svých zařízeních. Mikrofony ze sluchátek byly použity méně často (15 %), a pouze jedna osoba (5 %) použila externí mikrofon.

3. Prostředí:
Většina uživatelů (85 %) byla v tichém prostředí bez rušivých zvuků, zatímco menší část (15 %) zaznamenala mírné rušení, jako je zapnutá televize. Žádní respondenti nehlásili použití aplikace v hlasitém prostředí.

4. Funkce aplikace:
Polovina respondentů nevyužila ani 'Auto-Start' ani 'Předčítání'. Osm uživatelů aktivovalo pouze 'Auto-Start', zatímco kombinace obou funkcí byla využita pouze dvěma osobami.

5. Uvozovací fráze:
Většina respondentů (85 %) nepoužívala žádné uvozovací fráze při vyslovování slov, zatímco malý počet (10 %) je používal občas a pouze jeden respondent (5 %) je používal vždy.

6. Instrukce:
Většina účastníků (85 %) považovala instrukce za jasné a srozumitelné. Trocha zmatku (15 %) však byla u menšího počtu respondentů, kteří potřebovali více času na pochopení informací.

7. Intuitivnost aplikace:
Většina (85 %) uživatelů intuitivně pochopila, jak aplikaci používat bez potřeby podrobného návodu. Dva respondenti (10 %) měli smíšené zkušenosti a jeden (5 %) měl problémy s pochopením některých funkcí.

8. Vzhled aplikace:
Vzhled aplikace byl hodnocen kladně většinou uživatelů (80 %), zatímco někteří (20 %) ho hodnotili jako neutrální. Negativní hodnocení nebylo zaznamenáno.

9. Přehlednost aplikace:
Webová aplikace byla považována za velmi přehlednou většinou uživatelů (95 %) s jednoduchou orientací. Pouze jeden respondent (5 %) měl průměrné hodnocení přehlednosti.

10. Responzivita aplikace:
Většina (95 %) uživatelů potvrdila, že aplikace byla vždy správně zobrazena a prvky byly přehledně uspořádány. Pouze jeden respondent (5 %) zaznamenal výjimečné problémy s responzivitou.

11. Hlas syntézy:
Někteří respondenti (15 %) by uvítali modernější a přirozenější hlas syntézy, zatímco stejný počet je spokojen s aktuálním hlasem. Velká část (45 %) nevyužila hlas syntézy kvůli neaktivaci relevantních funkcí.

10 Závěr

Cílem této práce bylo navržení hlasového dialogového systému (HDS), který bude uživateli automaticky kontrolovat výslovnost a následná realizace dialogu ve formě webové aplikace.

První část práce se zaměřuje na teoretický popis problematiky HDS, především na automatické rozpoznávání řeči. V současné době je tato technologie založena převážně na neuronových sítích a transformerech, zejména pak na modelu wav2vec 2.0. Dále jsou představeny klíčové moduly systému, které zahrnují porozumění řeči, řízení dialogu, generování odpovědí a syntézu řeči. V závěrečné části je diskutováno vyhodnocení dialogu v rámci HDS.

Druhá teoretická část je věnována základním metodám porovnávání textových řetězců. Nejvhodnější se ukázala být upravená Levenshteinova metoda, která umožňuje využít libovolný začátek a konec podřetězce. Výsledná shoda vysloveného slova se pak hodnotí pomocí Levenshteinova poměru na procentuální bázi.

Další část práce podrobně popisuje hlasovou platformu SpeechCloud, která byla využita pro realizaci systému v této práci. Je zde popsána její architektura a funkčnost. Následně jsou představeny technologie nezbytné pro realizaci webové aplikace a celého systému. Mezi ně patří především HTML, CSS, DALL·E 3, DOM a WebSocket. Technologie DALL·E 3 je pak dále využita v praktické části pro generování všech obrázků určených pro kontrolu výslovnosti.

Následné testování se zaměřilo na tři rozpoznávače řeči, z nichž každý využívá odlišné technologie a přístupy k rozpoznávání řeči. Prvním z nich je tradiční rozpoznávač založený na architektuře TDNN s akustickým modelem, kde dekódování probíhá nad fonémovým výstupem s využitím slovníku a jazykového modelu. Dva další modely jsou založeny na modernější architektuře wav2vec 2.0. Jeden z nich využívá jazykový model, zatímco druhý aplikuje grafémový přístup bez přidaného slovního jazykového modelu. Testování ASR probíhalo tak, že byl nejprve vygenerován text, který se následně syntetizoval pomocí sedmi různých hlasů - Unit selection. Nahrávky byly poté pomocí jednotlivých rozpoznávačů převedeny zpět na text a porovnány s referenčním textem. Výsledky testování ukázaly, že jako nejlepší se jevil rozpoznávač wav2vec NO-LM bez přidaného jazykového modelu, který nejlépe odhalil různé nuance ve výslovnosti.

Aplikace je implementována jako webová stránka s důrazem na responzivitu tak, aby byla správně zobrazena na různých typech zařízení. Přidávání dalších slov je navrženo tak, aby bylo co nejjednodušší: stačí vložit obrázky do příslušné složky a přidat textový název. Po spuštění aplikace se uživateli postupně zobrazují vyskakovací okna pro nastavení aplikace, upozornění, doporučení a instrukce. Následně se otevře hlavní část aplikace, kde je možné začít s tréninkem výslovnosti. Uživatelé mohou využít různé formy nápovědy, jako je hlasové předčítání slov nebo vizuální přepis textu. Slova lze vyslovovat samostatně nebo ve větě, ale vždy v prvním pádě. Vyhodnocení probíhá kontinuálně a na konci je celkové vyhodnocení. Hodnocení je založeno na procentuální shodě vysloveného slova s referenčním textem pomocí Levensteinovy vzdálenosti (LD). U LD je možné nastavit libovolné ceny substituce v rámci tzv. *ztrátové matice*, která umožňuje různé ceny pro různé znaky (například cena za záměnu znaků *b* za *h* bude vyšší než za záměnu *a* za *á*).

Na závěr byla aplikace testována na 15 slovech a test byl proveden na 20 osobách. Tito lidé nejprve použili samotnou webovou aplikaci a následně vyplnili přiložený dotazník. Celková úspěšnost rozpoznávání, tedy shoda nalezených vyslovených slov s referenčními slovy, dosáhla **97.58 %**. Testování rovněž ukázalo, že aplikace dokáže rozlišit, zda uživatel řekl podobné nebo zcela jiné slovo ve srovnání s referencí. Uživatelé hodnotili webovou aplikaci jako přehlednou a intuitivní s funkčním responzivním designem. Většina uživatelů prováděla tento test na mobilním zařízení, a někteří by uvítali modernější syntézu řeči.

V současné implementaci aplikace udržuje stav pouze v rámci jedné relace. Jedním z možných vylepšení je řešení tohoto problému, které by uživatelům umožnilo ukládat jednotlivá slova k opakování, poskytovat chytřejší výběr slov pro výslovnost v různých relacích a umožnit porovnávání pokroku ve výslovnosti. Další možností zlepšení je použití dvou rozpoznávačů pro porovnávání na dvou úrovních. K současnému rozpoznávači bez přidaného jazykového modelu by se přidal rozpoznávač s jazykovým modelem. Posledním možným rozšířením je úprava pevného času pro rozpoznávání jednotlivých slov. Pokud by uživatel slovo vyslovil dříve, rozpoznávání by se ukončilo. Naopak, pokud by uživatel potřeboval více času, byl by mu nabídnut delší časový úsek. Tyto poslední dvě možnosti však nejsou v současné chvíli realizovatelné, protože je neumožňuje poskytnutá platforma SpeechCloud.

Literatura

[1] ROE, David B.; WILPON, Jay G. (ed.). *Voice Communication Between Humans and Machines*. Washington, DC: The National Academies Press, 1994. ISBN 978-0-309-04988-7. Dostupné z DOI: 10.17226/2308.

[2] ZUE, Victor; SENEFF, Stephanie. Spoken Dialogue Systems. In: *Springer Handbook of Speech Processing*. Ed. BENESTY, Jacob; SONDHI, M. Mohan; HUANG, Yiteng Arden. Berlin: Springer Berlin Heidelberg, 2008, s. 705–722. ISBN 978-3-540-49127-9. Dostupné z DOI: 10.1007/978-3-540-49127-9_35.

[3] TUPÝ, Jan. *Hlasová asistentka při vaření* [bakalářská práce]. Západočeská univerzita v Plzni, 2022. [cit. 2024-01-12]. Dostupné z: https://dspace5.zcu.cz/bitstream/11025/49359/1/BP_Hlasova_asistentka_pri_vareni_final_elektro_verze.pdf.

[4] ŠVEC, Jan. *Hlasové dialogové systémy* [učební text]. Západočeská univerzita v Plzni, 2023.

[5] YU, Dong; DENG, Li. Automatic Speech Recognition. In: *Automatic Speech Recognition*. Springer London, 2015, s. 705–722. ISBN 978-1-4471-5779-3. Dostupné z DOI: 10.1007/978-1-4471-5779-3.

[6] IRCING, Pavel. *Rozpoznávání řeči , akustické a jazykové modelování* [učební text]. Západočeská univerzita v Plzni, 2021.

[7] MARKOWITZ, Judith A. *Using speech recognition*. Upper Saddle River, NJ, Prentice Hall, 1996. ISBN 0131863215.

[8] JELINEK, Frederick. *Statistical methods for speech recognition*. 1997.

[9] VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N.; KAISER, Lukasz; POLOSUKHIN, Illia. *Attention Is All You Need*. 2017. Dostupné z DOI: 10.48550/ARXIV.1706.03762.

[10] BAEVSKI, Alexei; ZHOU, Henry; MOHAMED, Abdelrahman; AULI, Michael. *wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations*. 2020. Dostupné z DOI: 10.48550/ARXIV.2006.11477.

[11] PSUTKA, Josef. *Analýza a zpracování řečového signálu, parametrizace řeči* [učební text]. Západočeská univerzita v Plzni, 2021.

[12] IBE, Oliver C. 14 - Hidden Markov Models. In: IBE, Oliver C. (ed.). *Markov Processes for Stochastic Modeling (Second Edition)*. Second Edition. Oxford: Elsevier, 2013, s. 417–451. ISBN 978-0-12-407795-9. Dostupné z DOI: 10.1016/B978-0-12-407795-9.00014-1.

[13] MONTAVON, Grégoire; SAMEK, Wojciech; MÜLLER, Klaus-Robert. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*. 2018, roč. 73, s. 1–15. ISSN 1051-2004. Dostupné z DOI: 10.1016/j.dsp.2017.10.011.

[14] LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. *Nature*. 2015, s. 436–444. ISSN 1476-4687. Dostupné z DOI: 10.1038/nature14539.

[15] RADOVÁ, Vlasta. *Neuronové sítě* [učební text]. Západočeská univerzita v Plzni, 2023.

[16] LEDERER, Johannes. Activation Functions in Artificial Neural Networks: A Systematic Overview. 2021. Dostupné z DOI: 10.48550/ARXIV.2101.09957.

[17] SPOTFIRE. *What is a neural network?* [online]. [cit. 2024-04-20]. Dostupné z: https://www.spotfire.com/glossary/what-is-a-neural-network.

[18] ŠVEC, Jan. *Pokročilé neuronové sítě pro HDS* [učební text]. Západočeská univerzita v Plzni, 2023.

[19] WAIBEL, Alex. Modular Construction of Time-Delay Neural Networks for Speech Recognition. *Neural Computation*. [B.r.], roč. 1, č. 1, s. 39–46. ISSN 0899-7667. Dostupné z DOI: 10.1162/neco.1989.1.1.39.

[20] O'SHEA, Keiron; NASH, Ryan. *An Introduction to Convolutional Neural Networks*. 2015. Dostupné z DOI: 10.48550/ARXIV.1511.08458.

[21] HRÚZ, Marek. *Convolutional Neural Network* [učební text]. Západočeská univerzita v Plzni, 2023.

[22] *Cross-resolution face identification using deep-convolutional neural network - Scientific Figure on ResearchGate* [online]. [cit. 2024-04-28]. Dostupné z: https://www.researchgate.net/figure/Max-pooling-and-average-pooling_fig2_349921480.

[23] PARVEEN, Shahla; QADEER, Abdul; GREEN, Phil. Speaker recognition with recurrent neural networks. In: 2000. Dostupné z DOI: 10.21437/ICSLP.2000-270.

[24] HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long Short-Term Memory. *Neural Computation*. 1997, s. 1735–1780. ISSN 0899-7667. Dostupné z DOI: 10.1162/neco.1997.9.8.1735.

[25] RAVANELLI, Mirco; BRAKEL, Philemon; OMOLOGO, Maurizio; BENGIO, Yoshua. Light Gated Recurrent Units for Speech Recognition. 2018, s. 92–102. ISSN 2471-285X. Dostupné z DOI: 10.1109/tetci.2017.2762739.

[26] TOSHNIWAL, Shubham; KANNAN, Anjuli; CHIU, Chung-Cheng; WU, Yonghui; SAINATH, Tara N; LIVESCU, Karen. A Comparison of Techniques

for Language Model Integration in Encoder-Decoder Speech Recognition. In: 2018, s. 369–375. Dostupné z DOI: 10.1109/SLT.2018.8639038.

[27] GRAVES, Alex; FERNÁNDEZ, Santiago; GOMEZ, Faustino; SCHMIDHUBER, Jürgen. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural 'networks. In: 2006, s. 369–376. Dostupné z DOI: 10.1145/1143844.1143891.

[28] ŠVEC, Jan. *Porozumění řeči pro HDS* [učební text]. Západočeská univerzita v Plzni, 2023.

[29] PSUTKA, J.; MÜLLER, L.; MATOUŠEK, J.; RADOVÁ, V. *Mluvíme s počítačem česky*. Prague: Academia, 2006. ISBN 80-200-1309-1.

[30] DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 2018. Dostupné z DOI: 10.48550/ARXIV.1810.04805.

[31] ŠVEC, Jan. *Řízení dialogu pro HDS* [učební text]. Západočeská univerzita v Plzni, 2023.

[32] RADFORD, Alec; NARASIMHAN, Karthik; SALIMANS, Tim; SUTSKEVER, Ilya. *Improving Language Understanding by Generative Pre-Training* [online]. [cit. 2024-04-29]. Dostupné z: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

[33] RAFFEL, Colin; SHAZEER, Noam; ROBERTS, Adam; LEE, Katherine; NARANG, Sharan; MATENA, Michael; ZHOU, Yanqi; LI, Wei; LIU, Peter J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. 2019. Dostupné z DOI: 10.48550/ARXIV.1810.04805.

[34] MATOUŠEK, Jindřich. *Syntéza řeči: Zpracování textu a syntéza řeči z textu* [učební text]. Západočeská univerzita v Plzni, 2021.

[35] ŘEZÁČKOVÁ, Markéta. *Zpracování přirozeného jazyka* [učební text]. Západočeská univerzita v Plzni, 2023.

[36] SAGISAKA, Y. Speech synthesis from text. *IEEE Communications Magazine*. 1990, roč. 28, č. 1, s. 35–41. Dostupné z DOI: 10.1109/35.46669.

[37] MATOUŠEK, Jindřich; TIHELKA, Daniel; ŠMÍDL, Luboš. On the Impact of Annotation Errors on Unit-Selection Speech Synthesis. In: SOJKA, Petr; HORÁK, Aleš; KOPEČEK, Ivan; PALA, Karel (ed.). *Text, Speech and Dialogue*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, s. 456–463. ISBN 978-3-642-32790-2. Dostupné z DOI: 10.1007/978-3-642-32790-2_55.

[38] TAYLOR, P. Unit-selection synthesis. In: *Text-to-Speech Synthesis*. Cambridge Univer. Press, 2009, s. 474–516. Dostupné z DOI: 10.1017/CBO9780511816338.018.

[39] HANZLÍČEK, Zdeněk. *Statistická parametrická syntéza* [učební text]. Západočeská univerzita v Plzni, 2023.

[40] MATOUŠEK, Jindřich. *Neurální syntéza řeči* [učební text]. Západočeská univerzita v Plzni, 2023.

[41] WANG, Yuxuan; SKERRY-RYAN, RJ; STANTON, Daisy; WU, Yonghui; WEISS, Ron J; JAITLY, Navdeep; YANG, Zongheng; XIAO, Ying; CHEN, Zhifeng; BENGIO, Samy et al. Tacotron: Towards End-to-End Speech Synthesis. 2017. Dostupné z DOI: 10.48550/ARXIV.1703.10135.

[42] ARIK, Sercan Ö; CHRZANOWSKI, Mike; COATES, Adam; DIAMOS, Gregory; GIBIANSKY, Andrew; KANG, Yongguo; LI, Xian; MILLER, John; NG, Andrew; RAIMAN, Jonathan et al. Deep voice: Real-time neural text-to-speech. In: *International Conference on Machine Learning*. PMLR, 2017, s. 195–204. Dostupné z DOI: 10.48550/ARXIV.1702.07825.

[43] REN, Yi; RUAN, Yangjun; TAN, Xu; QIN, Tao; ZHAO, Sheng; ZHAO, Zhou; LIU, Tie-Yan. Fastspeech: Fast, robust and controllable text to speech. In: *NeurIPS*. 2019. Dostupné z DOI: 10.48550/ARXIV.1905.09263.

[44] OORD, Aaron van den; DIELEMAN, Sander; ZEN, Heiga; SIMONYAN, Karen; VINYALS, Oriol; GRAVES, Alex; KALCHBRENNER, Nal; SENIOR, Andrew; KAVUKCUOGLU, Koray. Wavenet: A generative model for raw audio. 2016. Dostupné z DOI: 10.48550/ARXIV.1609.03499.

[45] KUMAR, Kundan; KUMAR, Rithesh; BOISSIERE, Thibault de; GESTIN, Lucas; TEOH, Wei Zhen; SOTELO, Jose; BRÉBISSON, Alexandre de; BENGIO, Yoshua; COURVILLE, Aaron. Melgan: Generative adversarial networks for conditional waveform synthesis. In: *NeurIPS*. 2019. Dostupné z DOI: 10.48550/ARXIV.1910.06711.

[46] PENG, Kainan; PING, Wei; SONG, Zhao; ZHAO, Kexin. Non-autoregressive neural text-to-speech. In: *International Conference on Machine Learning*. PMLR, 2020, s. 7586–7598. Dostupné z DOI: 10.48550/ARXIV.1905.08459.

[47] PING, Wei; PENG, Kainan; CHEN, Jitong. ClariNet: Parallel Wave Generation in End-to-End Text-to-Speech. In: *International Conference on Learning Representations*. 2018. Dostupné z DOI: 10.48550/ARXIV.1807.07281.

[48] KIM, Jaehyeon; KONG, Jungil; SON, Juhee. Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech. 2021. Dostupné z DOI: 10.48550/ARXIV.2106.06103.

[49] DONAHUE, Jeff; DIELEMAN, Sander; BIŃKOWSKI, Mikołaj; ELSEN, Erich; SIMONYAN, Karen. End-to-end adversarial text-to-speech. In: *ICLR*. 2021. Dostupné z DOI: 10.48550/ARXIV.2006.03575.

[50] CAMBRIDGE UNIVERSITY PRESS & ASSESSMENT. *String - Cambridge Dictionary* [online]. [cit. 2024-04-12]. Dostupné z: https://dictionary.cambridge.org/dictionary/english/string.

[51] NAVARRO, Gonzalo. A guided tour to approximate string matching. *ACM Comput. Surv.* 2001, s. 31–88. ISSN 0360-0300. Dostupné z DOI: 10.1145/375360.375365.

[52] NEHA, Seth. *A Simple Guide to Metrics for Calculating String Similarity* [online]. [cit. 2024-04-14]. Dostupné z: https://www.analyticsvidhya.com/blog/2021/02/a-simple-guide-to-metrics-for-calculating-string-similarity.

[53] DOAN, AnHai; HALEVY, Alon; IVES, Zachary. 4 - String Matching. In: *Principles of Data Integration*. Boston: Morgan Kaufmann, 2012, s. 95–119. ISBN 978-0-12-416044-6. Dostupné z DOI: 10.1016/B978-0-12-416044-6.00004-1.

[54] GEHRMANN, Sebastian; URKE, Lauren; AMIR, Ofra; GROSZ, Barbara J. Deploying AI Methods to Support Collaborative Writing: a Preliminary Investigation. In: *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2015, s. 917–922. ISBN 9781450331463. Dostupné z DOI: 10.1145/2702613.2732705.

[55] WHATWG. *HTML Living Standard* [online]. [cit. 2024-05-11]. Dostupné z: https://html.spec.whatwg.org/.

[56] BETKER, James; GOH, Gabriel; JING, Li; BROOKS, Tim; WANG, Jianfeng; LI, Linjie; OUYANG, Long; ZHUANG, Juntang; LEE, Joyce; GUO, Yufei; MANASSRA, Wesam; DHARIWAL, Prafulla; CHU, Casey; JIAO, Yunxin; RAMESH, Aditya. *Improving Image Generation with Better Captions* [online]. [cit. 2024-04-29]. Dostupné z: https://cdn.openai.com/papers/dall-e-3.pdf.

[57] ŠVEC, Jan; NEDUCHAL, Petr; HRÚZ, Marek. Multi-modal communication system for mobile robot. 2022.

[58] ŠVEC, Jan. *Knihovna SpeechCloud.dialog* [interní dokument]. Západočeská univerzita v Plzni, 2020.

[59] TIHELKA, Daniel; HANZLÍČEK, Zdeněk; JŮZOVÁ, Markéta; VÍT, Jakub; MATOUŠEK, Jindřich; GRŮBER, Martin. Current State of Text-to-Speech System ARTIC: A Decade of Research on the Field of Speech Technologies: 21st International Conference, TSD 2018, Brno, Czech Republic, September 11-14, 2018, Proceedings. In: 2018, s. 369–378. ISBN 978-3-030-00793-5. Dostupné z DOI: 10.1007/978-3-030-00794-2_40.

[60] COMMON CRAWL. *Common Crawl* [online]. [cit. 2024-05-13]. Dostupné z: https://commoncrawl.org.

[61] LEHEČKA, Jan; ŠVEC, Jan; PRAŽÁK, Aleš; PSUTKA, Josef. Exploring Capabilities of Monolingual Audio Transformers using Large Datasets in Automatic Speech Recognition of Czech. In: *Proc. Interspeech 2022*. 2022, s. 1831–1835. ISSN 2308-457X. Dostupné z DOI: 10.21437/Interspeech.2022-10439.

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

Zkratka	Popis (Poznámka)
API	Application Programming Interface, aplikační programovací rozhraní
ASR	Automatic Speech Recognition, automatické rozpoznávání řeči
CNN	Convolutional Neural Network, konvoluční neuronová síť
CSS	Cascading Style Sheets, kaskádové styly
DM	Dialog Manager, dialogový manažer (správce)
DNN	Deep Neural Network, hluboké neuronové sítě
DOM	Document Object Model, objektový model dokumentu
GUI	Graphical User Interface, grafické uživatelské rozhraní
HDS	Hlasový Dialogový Systém
HMM	Hidden Markov Model, skryté Markovské modely
HTML	HyperText Markup Language , hypertextový značkovací jazyk
JS	JavaScript (programovací jazyk)
JSON	JavaScript Object Notation (způsob zápisu dat)
LD	Levenshtein Distance, Levenshteinova vzdálenost
LM	Language Model, jazykový model
LPC	Linear Predictive Coding, lineární prediktivní kódování
LR	Levenshtein Ratio, Levenshteinův poměr
LSTM	Long Short-Term Memory, dlouhá krátkodobá paměť (neuronová síť)
MFCC	Mel-Frequency Cepstral Coefficient, mezifrekvenční kepstrální koef.
NLG	Natural Language Generation, generování přirozeného jazyka
NLP	Natural Language Processing, zpracování přirozeného jazyka
PLP	Perceptual Linear Prediction, percepční lineární predikce
RNN	Recurrent Neural Network, rekurentní neuronová síť
RTP	Real-Time Transport Protocol (protokol)
SC	SpeechCloud (platforma)
SED	Semantic Entity Detection, detekce sémantické entity
SIP	Session Initiation Protocol (protokol)
SLU	Spoken Language Understanding, porozumění mluvené řeči
TCP	Transmission Control Protocol (protokol)
TDNN	Time Delay Neural Network, neuronová síť s časovým zpožděním
TTS	Text to Speech, syntéza řeči z textu
URL	Uniform Resource Locator, jednotný lokátor zdroje