Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

# BAKALÁŘSKÁ PRÁCE

## Rozpoznávání řeči pomocí neuronových sítí s navazujícím sequence-to-sequence modelem

Plzeň, 2023/2024

Matěj Šulc

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

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

| | |
|---|---|
| Jméno a příjmení: | **Matěj ŠULC** |
| Osobní číslo: | **A21B0403P** |
| Studijní program: | **B0714A150005 Kybernetika a řídicí technika** |
| Specializace: | **Umělá inteligence a automatizace** |
| Téma práce: | **Rozpoznávání řeči pomocí neuronových sítí s navazujícím sequence-to-sequence modelem** |
| Zadávající katedra: | **Katedra kybernetiky** |

## Zásady pro vypracování

1. Nastudujte problematiku neuronových sítí, architekturu Wav2Vec 2.0, vybraných sequence-to-sequence modelů a techniku jejich fine-tuningu.
2. Seznamte se s veřejnými českými datasety pro úlohu automatického rozpoznávání řeči.
3. Natrénujte Wav2Vec 2.0 model z vhodného datasetu na úlohu rozpoznávání řeči.
4. Navrhněte rozšíření rozpoznávače řeči o navazující sequence-to-sequence model.
5. Modely vyhodnoťte a porovnejte.

Rozsah bakalářské práce: **30-40 stránek A4**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:


1. VASWANI, Ashish, et al. Attention is all you need. Advances in neural information processing systems, 2017, 30.
2. BAEVSKI, Alexei, et al. wav2vec 2.0: A framework for self-supervised learning of speech representations. Advances in neural information processing systems, 2020, 33: 12449-12460.
3. LEHEČKA, Jan, et al. Exploring capabilities of monolingual audio transformers using large datasets in automatic speech recognition of Czech. arXiv preprint arXiv:2206.07627, 2022.
4. RAFFEL, Colin, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research, 2020, 21.1: 5485-5551.


Vedoucí bakalářské práce: **Ing. Jan Lehečka, Ph.D.**
Výzkumný program 1

Datum zadání bakalářské práce: **17. října 2023**
Termín odevzdání bakalářské 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 17. října 2023

# Prohlášení

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

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

V Plzni, dne

.....
*Matěj Šulc*

# Poděkování

Rád bych tímto poděkoval vedoucímu své bakalářské práce panu Ing. Janu Lehečkovi, Ph.D. za jeho odborné vedení, ochotu a cenné připomínky, které mi pomohli v průběhu psaní práce.

Dále bych také rád poděkoval své rodinně za jejich neustálou podporu a trpělivost během studia a psaní této práce.

MetaCentrum:
Computational resources were provided by the e-INFRA CZ project (ID:90254),
supported by the Ministry of Education, Youth and Sports of the Czech Republic.

# Abstrakt

Tato bakalářská práce se zaměřuje na vývoj a optimalizaci systému rozpoznávání řeči využívající kombinaci dvou hlavních technologických nástrojů z domén zpracování akustických signálů a přirozeného jazyka, a to neuronové sítě Wav2Vec2.0, která je doplněna o sequence-to-sequence model T5. Cílem je zvýšit přesnost převodu mluveného slova na textový formát.

V první části práce je proveden rozbor metody Wav2Vec2.0, která slouží k extrakci významných akustických informací z audio nahrávek a vytvoření textové reprezentace. Následně je popsán sequence-to-sequence model T5, jenž slouží k úpravě získaného přepisu řeči, jelikož na rozdíl od Wa2Vec2.0 modelu vychází ze znalostí o přirozeném jazyce. Tyto dva klíčové prvky jsou následně integrovány do funkčního systému pro rozpoznávání řeči.



**Klíčová slova:** automatické rozpoznání řeči, ASR, STT, Transformers, Wav2Vec2.0, seq2seq, T5



# Abstract

This bachelor thesis focuses on the development and optimization of a speech recognition system utilizing a combination of two main technological tools from the domains of acoustic signal processing and natural language processing: the Wav2Vec2.0 neural network, complemented by the sequence-to-sequence model T5. The goal is to enhance the accuracy of converting spoken words into textual format.

In the first part of the thesis, an analysis of the Wav2Vec2.0 method is conducted, which serves for extracting significant acoustic information from audio recordings and creating a textual representation. Subsequently, the sequence-to-sequence model T5 is described, which is used to refine the obtained speech transcription, as it differs from the Wa2Vec2.0 model by incorporating knowledge about natural language. These two key elements are then integrated into a functional system for speech recognition.



**Key words:** automatic speech recognition, ASR, STT, Transformers, Wav2Vec2.0, seq2seq, T5

# Obsah

# 1 Úvod

Tato práce se zaměřuje na problematiku automatického rozpoznávání řeči s využitím moderních metod strojového učení, konkrétně na přístupu založeném na Wav2Vec2.0 modelu [1, 2] popisovaném v kapitole 3. Tento model, vyvinutý v Meta AI (Facebook) v roce 2020, představuje významný pokrok v oblasti automatického rozpoznávání řeči, nabízející vylepšenou schopnost převodu akustických dat na textovou podobu. Využívá se schopnosti předtrénování modelu na velkém množství neoznačených dat, tedy modelu není předkládána dvojice audio nahrávky s jeho textovou podobou, ale předkládají se pouze audio nahrávky. Pro následné dotrénování modelu stačí značně menší část již označených dat, na kterých se naučí přikládat k nahrávce její textový přepis. Tedy dochází k využití možnosti přenesení znalostí o reprezentaci řeči v nahrávkách, jenž byly získány během předtrénování.

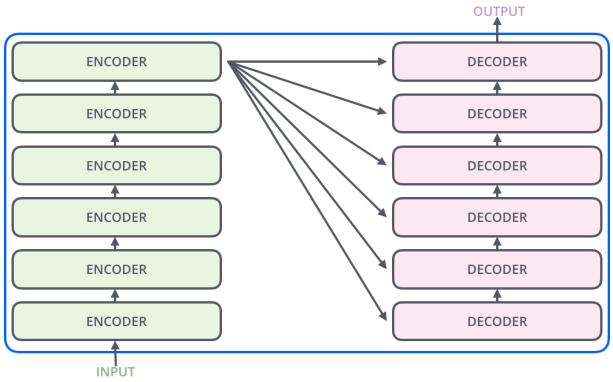
Předchozí odstavec napovídá, že model Wav2Vec2.0 je předtrénován na velkém množství audia a textová podoba řeči je mu předkládána až ve finální fázi dotrénování, kde se teprve učí přikládat znaky k získaným významným charakteristikám audio nahrávek. Z tohoto důvodu je v rámci této práce navrhnut přístup, který doplní model Wav2Vec2.0 o navazují sequence-to-sequence model T5. Model T5 [3], vyvinutý firmou Google v roce 2019, slouží jako přídavná informace o znalosti přirozeného jazyka (jazykový model, Language Model - LM), díky které by mělo dojít k opravám chyb typických pro přepis záznamu řečového signálu. Dále blíže popsáno v kapitolách týkajících se modelu T5 4 a analýze chyb 7.

Motivací pro navržené řešení vychází z požadavků současného digitálního světa, kde technologie automatického rozpoznávání řeči (Automatic Speech Recognition - ASR) stále postupně proniká do každodenního života a stává se stále důležitějším a žádanějším prvkem. Tato technologie nám umožňuje překonat komunikační bariéry a vytvářet intuitivnější uživatelské rozhraní pro širokou škálu aplikací, od virtuálních asistentů po rozpoznávání hlasu v mobilních zařízeních či počítačích. Automatické rozpoznávání řeči tedy tvoří první blok v tomto jinak komplexním systému a je zapotřebí, aby chyby zanesené samotným rozpoznáním řeči byly minimální a neovlivňovali navazující systémy. Navržené řešení, společně s vyhodnocením a analýzou chyb, lze nalézt v kapitolách 6 a 7.

# 2 Architektura Transformer modelu

Transformer [4] je architektura neuronových sítí, jejíž klíčovým prvkem je mechanismus attention [5, 6]. Attention mechanismus umožňuje modelu naučit se, jak velká pozornost (attention) se má věnovat tokenům v okolí pro právě zkoumaný token (základní jednotka textu, token například může reprezentovat celé slovo, znak nebo část slov). Tedy dochází k vytváření kontextu pro daný token, což nachází značné uplatnění v úlohách zpracování přirozeného jazyka (Natural Language Processing - NLP). Některé nejznámější současné modely, které se objevují na poli NLP, se skládají až z desítek transformerových bloků nebo některých jeho variant. Mezi tyto zástupce patří právě zkoumané modely Wav2Vec2.0 a T5 (kapitoly 3 a 4). Důležitou vlastností architektury transformer je možnost učení enkodéru a dekodéru bez učitele. Pro předtrénování modelu tedy lze využít velkého množství neoznačených dat a poté, kdy takto předtrénovaný model, lze v následující fázi dotrénovat na již mnohem menší množině označených dat a naučit tak plnit požadovanou úlohu.

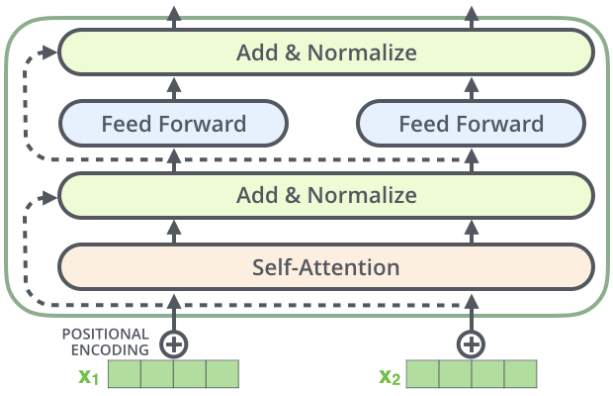
Tato architektura je tvořena ze dvou základních komponent, a to sekvence bloků enkodérů a dekodérů (sekvence nemusejí mít stejný počet bloků, ale typicky tomu tak bývá, jako je například uvedeno v původní práci [4], kde model obsahuje sekvenci enkodérů a dekodérů o délce 6-ti bloků), které jsou vzájemně propojeny (detail viz. obrázek 1).




Obrázek 1: Obecný náhled na architekturu transformer ([7], upraveno).

## 2.1 Enkodér, self-attention mechanismus a vstup

Enkodér a jeho mechanismus self-attention v architektuře transformer hraje klíčovou roli při zpracování vstupních sekvencí. Na obrázku níže (2) je vyobrazen detailnější náhled do jednoho z bloků sekvence enkodérů.




Obrázek 2: Detailní náhled na architekturu bloku enkodéru ([7], upraveno).

Před samotným vstupem do prvního bloku enkodéru je nutné vstupní sekvenci rozdělit na jednotlivé tokeny, kdy je využit tokenizer, mechanismus převádějící vstupní text na zmíněné tokeny. Následně se z nich pomocí embeddingu vytvářejí příznakové vektory, jenž jsou dimenze o velikosti 512. Tyto příznakové vektory jsou dále sčítány s *positional encoding* vektorem, který reprezentuje informace o jejich pozici v dané vstupní sekvenci. V práci [4] popisují využití sinusových a kosinových funkcí s lišícími se frekvencemi pro vytvoření *positional encoding* vektorů, které vycházejí z aktuální pozice tokenů, dimenze jejich příznakových vektorů a velikosti modelu.

Self-attention mechanismus potřebuje pro každý vstupní příznakový vektor $x_i$ s poziční informací, kde $i$ označuje počet tokenů ve vstupní sekvenci (dále jako parametr *input_length*), vytvořit trojici vektorů query $q$, key $k$ a value $v$, každý vektor je dimenze 64. Toho je docíleno vytvořením váhových matic $W^Q$, $W^K$ a $W^V$, které jsou optimalizovány ve fázi trénování a mají dimenzi $512 \times 64$. Příznakové vektory $x_i$, pro paralelizaci výpočtu, jsou poskládány pod sebe do matice $X$ s dimenzí $input\_length \times 512$, kdy maticovým násobením matice $X$ s váhovými maticemi $W^Q$, $W^K$ a $W^V$ vzniknou příslušné matice $Q$, $K$ a $V$ (dimenze $input\_length \times 64$), jejichž řádky tvoří dříve zmiňované požadované vektory $q$, $k$ a $v$ pro každý vstupní token. Následně lze již použít vzorec pro výpočet self-attention:

$$z = \text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_{kv}}}\right)V \tag{1}$$

kde $d_{kv}$ je 64 (dimenze vektorů $q$, $k$, $v$) a využívá se jako škálovací faktor pro zajištění numerické stability.

Ve stejné práci [4] je dále tento přístup rozšířen o použití více self-attention mechanismů 1 s využitím různých matic $Q$, $K$ a $V$ zvaný "multi-headed"attention. Toto obohacení rozšíří možnosti modelu přidávat pozornost na více míst ve vstupní sekvenci, které je zapříčiněno právě použitím $h$ různých projekčních matic $W_i^Q$, $W_i^K$ a $W_i^V$, kde $i = 1, ..., h$ (ve zmiňované práci bylo použito $h = 8$). Výpočet multi-headed attention přechází na vzorec:

$$\begin{aligned} Z = \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \\ \text{kde head}_i &= \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right) \end{aligned} \tag{2}$$

kde matice $W^O$ je přidaná matice dimenze $h \cdot 64 \times 512$, která zařídí projekci z jednotlivých attention hlav na výstup tak, aby byly dodrženy jeho požadované dimenze.

Dalším hlavním blokem je plně propojená dopředná síť (Feed-Forward Network, FFN). Tato síť je aplikována na každý výstupní vektor zvlášť za účelem zachování pozičních informací. Skládá se ze dvou plně propojených lineárních vrstev vytvářející projekci do a ze skryté dimenze $d_{ff}$ (typicky bývá větší než je dimenze využitá v attention blocích, následující kapitoly označují dimenzi attention bloků jako $d_{model}$), kde se nachází ReLU (Rectified Linear Unit) aktivační funkce

$$\text{FFN}(x) = \max\left(0; xW_1 + b_1\right)W_2 + b_2 \tag{3}$$

a kde jsou využity $W_i$ váhové matice a $b_i$ prahové vektory.

Výstupy z bloků self-attention a FFN jsou dále sčítány s jejich vstupy a normalizovány. Zde je uveden případ pro výstup z bloku self-attention:

$$Z_{norm} = \text{LayerNorm}(X + Z) \tag{4}$$

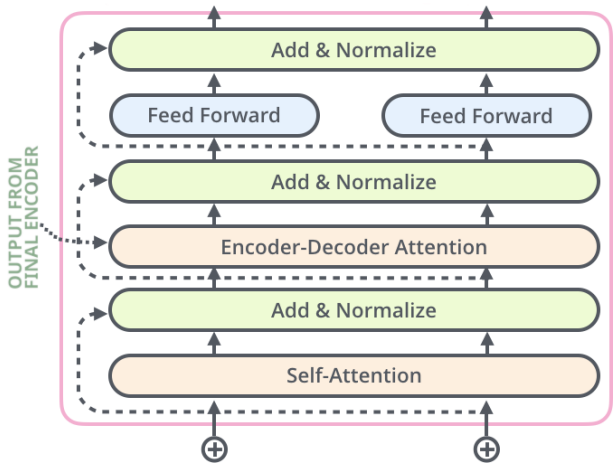
Implementace funkce normalizace vrstev ve frameworku PyTorch, který je využíván v této práci, vychází z práce [8] a nabývá formy

$$y = \text{LayerNorm}(x) = \frac{x - \text{E}[x]}{\sqrt{\text{Var}[x] + \epsilon}} \cdot \gamma + \beta \tag{5}$$

kde $\epsilon$ je konstanta přidávaná pro numerickou stabilitu, $\gamma$ a $\beta$ jsou volitelné a v průběhu učení laditelné parametry reprezentující afinní transformaci (škálování a posun).

## 2.2 Dekodér, propojení komponent a výstup

Dekodér je svojí strukturou a funkčností velmi podobný enkodéru popsaného v předchozí kapitole 2.1 s tím rozdílem, že obsahuje dva bloky attention mechanismu. První, vstupní self-attention blok, který zpracovává výstup od předchozího dekodéru. A druhý, který navíc zpracovává výstup od posledního bloku ze sekvence enkodérů svým attention mechanismem, což bývá označováno jako cross-attention nebo enkodér-dekodér attention. Náhled do struktury jednoho z bloků dekodéru je vyobrazena na následujícím obrázku 3.




Obrázek 3: Detailní náhled na architekturu bloku dekodéru ([7], upraveno).

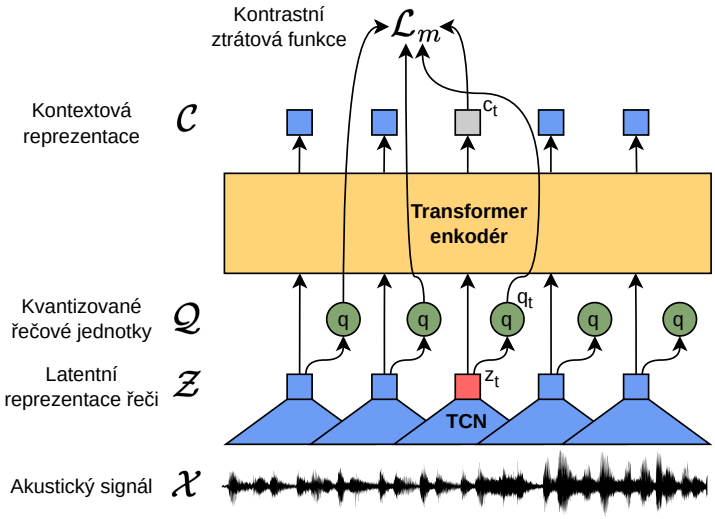
Výstup je poté vytvořen za pomoci plně propojené vrstvy, která vytváří projekci do vektoru o velikosti slovníku modelu (získané při trénování), jehož prvky jsou následně normalizovány vrstvou `softmax` 6 do intervalu <0; 1>. Samotný výběr slova (tokenu) ze slovníku může být realizovaný funkcí `argmax`, která vybere index slova s nejvyšším skóre.

$$\text{softmax}(\mathbf{x}) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \text{ kde } i = 1, ..., N \text{ a } \mathbf{x} = [x_1, x_2, ..., x_i, ..., x_N] \tag{6}$$

# 3 Model Wav2Vec2.0

Wav2Vec2.0, popisovaný v práci [1], značně těží z možnosti předtrénování založeném na učení bez učitele s velkou sadou neoznačených dat, ze kterých získá znalosti o reprezentaci akustických signálů v jejich obrazové (latentní) formě, podobně jako u zpracování NLP pomocí Bidirectional Encoder Representations from Transformers (BERT, [9]), jenž se ukázalo být velmi dobrým přístupem. Tento akustický model přistupuje ke zpracování signálu podobně jako jazykový model BERT, a to s využitím vytvořeného embeddingu (latentní reprezentaci), který po průchodu sítí transformerového enkodéru vytvoří kontextovou reprezentaci přes celou vstupní sekvenci. Cílem je poté pomocí kontrastního učení získat charakteristiky takové, aby pro vektory v latentním prostu patřící do různých kódových knih se co nejvíce lišily, a pro podobné si naopak byly velmi blízké.

Během předtréninku se v kvantizačním modulu získávají reprezentace diskrétních řečových jednotek z latentního prostoru řečového signálu. Tyto jednotky se poté využívají jako objekty v latentním prostoru u kontrastního učení, kdy je pro jejich výběr využita Gumbel softmax funkce (12). Na následujícím obrázku 4 je zobrazen diagram architektury Wav2Vec2.0 modelu.

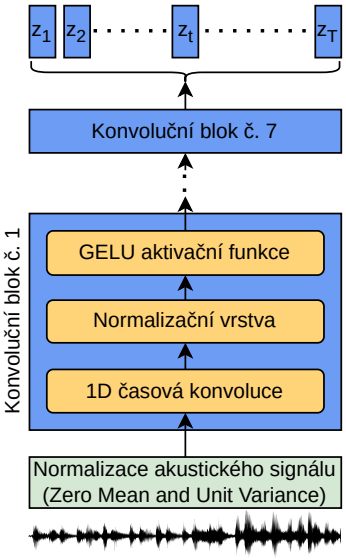



Obrázek 4: Obecný náhled na architekturu Wav2Vec2.0 modelu při fázi předtrénování ([1], upraveno).

## 3.1 Architektura Wav2Vec2.0 modelu

### 3.1.1 Příznakový enkodér

Příznakový enkodér 5 se sestává z několika bloků. Prvním je standardizace vstupního akustického signálu (Zero Mean and Unit Variance, tedy nulová střední hodnota a jednotkový rozptyl, 7), které je následováno sedmy, za sebou seřazenými, konvolučními bloky (viz. obrázek 5). Tyto konvoluční bloky se sestávají z jednorozměrné časové konvoluční vrstvy (Temporal Convolution Network - TCN), následované normalizační vrstvou ([8], 5) a GELU (Gaussian Error Linear Unit) aktivační funkcí ([10], 10).




Obrázek 5: Diagram příznakového enkodéru.

$$x_{norm}(t) = \frac{x(t) - \mathrm{E}(x)}{\mathrm{Std}(x)} \tag{7}$$

Tabulka 1: Parametry jednotlivých konvolučních bloků.

| Blok | # kanálů | Jádro | Krok |
|---|---|---|---|
| 1 | 512 | 10 | 5 |
| 2 | | 3 | 2 |
| 3 | | | |
| 4 | | | |
| 5 | | | |
| 6 | | 2 | |
| 7 | | | |

Každý konvoluční blok obsahuje na vstupu vrstvu TCN, kde každá má nastavené své parametry popsané v tabulce 1. S tímto nastavením bude na výstupu enkodéru $z_T$ vektorů s dimenzí 512, kde $T$ je přibližně (nutné vzít v potaz potřebné doplnění krajních elementů ve vstupním vektoru - padding, využití interního speciální tokenu [PAD]):

$$T \approx \frac{\mathrm{size}(\mathcal{X})}{5 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2} = \frac{\mathrm{size}(\mathcal{X})}{320} \tag{8}$$

tedy vstup $\mathcal{X}$, který byl navzorkován 16kHz ($sampling\_rate$), je po průchodu 7 konvolučními bloky transformován do latentního prostoru $\mathcal{Z}$, který jak již bylo zmíněno, má $z_T$ vektorů s dimenzí 512 obsahující zakódované audio s výstupní frekvencí $f_{out} \approx 49$Hz (přibližně 20ms mezi jednotlivými vzorky).

$$f_{out} = \frac{T \cdot sampling\_rate}{\mathrm{size}(\mathcal{X})} \tag{9}$$

Součástí těchto bloků je také GELU aktivační funkce, která je aplikována na výstupy z TCN. GELU je vyjádřena v následující rovnici:

$$\mathrm{GELU}(x) = x \cdot \mathrm{P}(X \leq x) = x \cdot \Phi(x) = x \cdot \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)\right] \tag{10}$$

kde $\mathrm{erf}(x)$ značí gaussovskou kumulativní distribuční funkci, díky níž má GELU, na rozdíl od ReLU, hladký průběh. Pro výpočet s reálnými argumenty bývá aproximována pomocí hyperbolických funkcí.

$$\begin{aligned} \mathrm{erf}(x) &= \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \mathrm{d}t \\ \text{aproximace } \mathrm{erf}(x) &\approx \tanh\left(\frac{2}{\sqrt{\pi}}\left(x + \frac{11}{123}x^3\right)\right) \end{aligned} \tag{11}$$

Příznakové vektory $z_t$ po průchodu enkodérem mají v latentním prostoru, jak již bylo zmíněno, dimenzi 512, což by nemuselo vyhovovat některým variantám modelu, jejichž transformerový enkodér má bloky o rozdílné dimenzi $d_{model}$ (více v následující kapitole). Proto je mezi tyto části přidána plně propojená lineární vrstva vytvářející projekci z dimenze latentního prostoru 512 do dimenze používané danou variantou modelu $d_{model}$.

### 3.1.2 Kontextová reprezentace

Kontextová síť, jenž sdílí podobnou architekturu s enkodérem transformeru, jako byl popsán v kapitole 2, zpracovává výstup z příznakového enkodéru a vytváří jeho kontextovou reprezentaci $\mathcal{C}$. Zobrazuje tedy vektory $z_t$ z latentního prostoru $\mathcal{Z}$ do jejich kontextové reprezentace $c_t$, které budou obsahovat informace o celé vstupní sekvenci. Dále bude pojem kontextová síť nahrazena pojmem transformerový enkodér z důvodů podobnosti architektury a již zavedeným pojmům.

Transformerový enkodér použitý v BASE variantě modelu Wav2Vec2.0 obsahuje 12 transformerových bloků s dimenzí $d_{model} = 768$, ve kterých je 8 self-attention hlav a FFN se skrytou dimenzí $d_{ff} = 3072$. Dále nejsou používány fixní *positional encodings* popsané v [4] a kapitole 2, ale byla využita konvoluční síť vytvářející relativní poziční informace (velikost jádra je 128 a 16 skupin), které byly popsány v práci [11]. Výstup z této sítě je následován GELU aktivační funkcí, který je poté sčítán s původním vstupem $z_t$ a je aplikována normalizace na tento součet pomocí vztahu definovaném na 5.

### 3.1.3 Kvantizační modul

Kvantizační modul vytváří diskrétní reprezentaci latentního prostoru $\mathcal{Z}$ na konečný počet řečových jednotek $\mathcal{Q}$ za použití shlukovacího algoritmu Product Quantization (dále jen PQ, [12]). Kvantizace se vytváří a je využívána jen ve fázi předtrénování pro učení bez učitele, kdy figuruje při výpočtu ztrátových funkcí 3.1.4 pro učení a optimalizaci kontextové reprezentace.

PQ spočívá ve výběru kvantizovaných reprezentací z $G$ kódových knih, kdy každá obsahuje $V$ záznamů $e \in \mathbb{R}^{V \times d/G}$ (pro BASE verzi modelu $G = 2, V = 320$), z čehož vyplývá, že model může mít až 102.4k záznamů. Z kódových knih vždy dojde k výběru jednoho ze záznamů $e_v$, které jsou nejbližší k latentní reprezentaci $z_t$. Spojením těchto záznamů a následnou aplikací lineární transformace lze získat kvantizovanou řečovou jednotku $q_t$.

Pro výběr záznamů $e_v$, které jsou nejbližší k latentní reprezentaci $z_t$, slouží Gumbelova softmax funkce [13], která je aplikována pro každou kódovou knihu, tedy $G$-krát. Poté výběr $v$-tého záznamu z kódové knihy $g$ se provádí pomocí:

$$p_{g,v} = \frac{\exp(l_{g,v} + n_v)/\tau}{\sum_{k=1}^{V} \exp(l_{g,k} + n_k)/\tau} \tag{12}$$

kde $n_v = -\log(-\log(u))$, $u$ je náhodná veličina vybraná z rovnoměrného rozdělení $\mathcal{U}(0, 1)$ a $\tau$ je nezáporný faktor omezující vliv dané náhodné veličiny.

Díky vhodné volbě funkce lze při dopředném chodu vybrat záznam $e_v$, jehož argument maximalizuje funkci, tedy $v = \underset{v \in V}{\operatorname{argmax}}(p_{g,v})$, a při zpětném chodu lze určit a použít její opravdový gradient.

### 3.1.4 Ztrátová funkce

Během předtrénování se model učí reprezentovat akustický signál řešením kontrastní ztrátové funkce $\mathcal{L}_m$, která nutí model identifikovat opravdovou kvantizovanou reprezentaci $q_t$ pro maskovaný úsek $z_t$ pomocí jeho kontextové reprezentace $c_t$ a známých $K$ distraktorů $\tilde{q} \in \mathcal{Q}_t$, které jsou náhodně vybírány (viz. obrázek 4, červeně označený maskovaný úsek $z_t$). Kontrastní ztrátové funkce je tedy definována jako

$$\mathcal{L}_m = -\log \frac{\exp(\mathrm{sim}(c_t, q_t)/\kappa)}{\sum_{\tilde{q} \sim \mathcal{Q}_t}^{K} \exp(\mathrm{sim}(c_t, \tilde{q})/\kappa)} \tag{13}$$

kde se podobnost počítá pomocí kosinové podobnosti $\text{sim}(a,b) = \frac{a^T b}{||a|| \cdot ||b||}$ a $\kappa$ je volitelný škálovací faktor (v práci [1] nastaven na $\kappa = 0.1$).

Kontrastní ztrátová funkce závisí na náhodném výběru záznamů z kódových knih, které by měly být rovnoměrně vybírány. Proto je přidána další ztrátová funkce $\mathcal{L}_d$, která má zaručit rovnoměrný výběr všech záznamů $V$ z každé kódové knihy $G$. Čehož je docíleno ztrátovou funkcí diverzity, která nutí model maximalizovat entropii vybráním nějakého záznamu $v$ ze všech kódových knih $G$, tato funkce má poté tvar:

$$\mathcal{L}_d = \frac{1}{GV} \sum_{g=1}^{G} -H(\bar{p}_g) = \frac{1}{GV} \sum_{g=1}^{G} \sum_{v=1}^{V} \bar{p}_{g,v} \log \bar{p}_{g,v} \tag{14}$$

Tedy celková ztrátová funkce s využitím informace od kontrastní a diverzitní funkce nabývá tvaru

$$\mathcal{L} = \mathcal{L}_m + \alpha \mathcal{L}_d \tag{15}$$

kde $\alpha$ je volitelný váhový parametr (v původní práci [1] byl zvolen $\alpha = 0.1$).

## 3.2 Předtrénování a použitý model

### 3.2.1 Předtrénování Wav2Vec2.0 modelu

Wav2Vec2.0 modelu ve fázi předtréninku, jehož schéma lze nalézt na obrázku 4, je maskována část vektorů latentní reprezentace $z_t$, v původní práci se uvádí přibližně 49% ze všech časových úseků. Maskování je parametrizováno náhodným výběrem počátečního úseku s $p = 0.065$ a zamaskování $M = 10$ následujících časových úseků, kdy je jejich překryv povolen (parametrizace a její nastavení je převzato z původní práce [1]).

Takto maskovaná reprezentace je dále předaná do bloku s transformerovým enkodérem, jenž vytváří kontextovou reprezentaci v podobě vektorů $c_t$. Zároveň dochází k určení kvantizované reprezentace všech úseků, kdy pro určení $q_t$ jsou všechny úseky odmaskovány, jelikož nejsou hlavním předmětem v předtrénování a figurují při výpočtu ztrátové funkce. Následně se určuje ztrátová funkce pro všechny $c_t$, které měly maskovaný vstup, založeném na dvou aspektech popisovaných v kapitole 3.1.4.

### 3.2.2 Implementace a použitý model

Výchozím modelem Wav2Vec2.0 byla jeho BASE varianta, jehož architektura a parametry jsou popsané v předchozí kapitole 3.1 a původní práci [1]. Byla použita jeho implementace ve frameworku PyTorch [14] a obalující abstrakci poskytnula platforma HuggingFace prostřednictvím knihovny `transformers` [15].

Wav2Vec2.0 model byl předtrénovaný na sadě přibližně 80k hodin neoznačené české řeči z mnoha oblastí, jako jsou například záznamy z rádiového a televizního vysílání, nahrávky od stínových řečníků, apod. Pro zpřístupnění jednoho z pozdějších uložených checkpointů modelu a jeho představení bych rád poděkoval panu Ing. Janu Lehečkovi, Ph.D. a odkázal se na jeho výzkumnou práci [2], kde lze získat více informací o předtrénování, vlastnostech a dosažitelných výsledcích na poli automatického rozpoznání řeči při použití Wav2Vec2.0 modelu.

## 3.3 Dataset využitý pro dotrénování

Model byl dotrénovaný na datasetu Common Voice [16], ačkoliv byl součástí datasetu i pro předtrénování, během něhož bylo použito pouze audio z *train* a *validation* porce, tak

tvořil pouze zlomek z celkových 80k hodin audio nahrávek. Z toho důvodu by neměl být problém je opětovně využít pro dotrénování, kdy dosahované výsledky by tímto faktem neměly být nijak ovlivněny.

Tento dataset je veřejně dostupná sbírka hlasových dat, kterou spravuje organizace Mozilla Foundation. Dataset byl vytvořen s cílem podporovat vývoj a vylepšování technologií automatického rozpoznávání řeči a dalších hlasových aplikací. Je k dispozici zdarma pro výzkumné účely a přispívající jednotlivci mohou nahrané hlasové vzorky sdílet s komunitou a verifikovat poskytnuté transkripce u záznamům. Pro trénování byla využita jeho verze 11 zveřejněná na platformě HuggingFace a pro manipulaci s daty byla využita knihovna `datasets` [17].

Dataset Common Voice 11 byl vybrán na základě rozsáhlé a verifikované české části, která obsahuje trénovací sadu s 14612 příklady, kdy průměrná délka nahrávky je přibližně 4.76 sekundy a celková délka nahrávek je 19.31 hodin. Dále obsahuje menší validační a testovací porce dat, které každé obsahují přibližně přes 7k příkladů.

Tabulka 2: Základní statistiky české porce dat z datasetu Common Voice 11.

| | *train* | *validation* | *test* |
|---|---|---|---|
| # příkladů [1] | 14612 | 7543 | 7714 |
| # hodin [hod.] | 19.31 | 9.65 | 9.71 |
| prům. délka [sek.] | 4.76 | 4.61 | 4.53 |

Za účely dotrénování (fine-tuningu) byla využita sada *train*, pro validaci během trénování byla využila sada *validation* a pro otestování funkčnosti a dosahovaných výsledků modelu byla využita sada *test* dosud neviděných dat během trénování. Běhům v tabulce 5 označených * byla navýšena trénovací sada o *validation* sadu a samotná validace během trénování se prováděla na sadě *test*. Verze modelů označené * nebyly v dalším průběhu práce nijak využívány.

## 3.4 Dotrénování modelu Wav2Vec2.0

Předtrénovaný model je ve fázi dotrénování doplněn o náhodně inicializovanou lineární vrstvu (plně propojenou). Tato vrstva slouží jako projekce výstupu z transformerového enkodéru, kdy BASE varianta modelu produkuje $t$ vektorů zvaných logits (kde $t = 1, ..., T$) a každý je dimenze o velikosti 768, do daného počtu tříd $C_{W2V2}$, jenž reprezentují znaky z použitého slovníku (dále popsáno v této kapitole 5.3). Tato klasifikační hlava modelu může být doplněna o jednoduchý jazykový model (n-gram, beam-search). Jelikož cílem této práce je přidání samostatného jazykového modelu T5 (popis funkce a dosažené výsledky spojením modelů v kapitole 6), který by měl dosáhnou vyšší přesnosti při korekci výstupu. Z tohoto důvodu byl zvolen prostý výběr nejpravděpodobnějšího tokenu pro daný časový úsek $t$ za pomocí funkce `argmax` z vektoru logits, kdy pro jeho získaný index je následně vyhledán odpovídající znak ve slovníku $C_{W2V2}$. Poskytnutým dekodérem se takto zpracuje celá sekvence a dojde k převodu do textové formy.

Během dotrénování modelu je cílem minimalizovat ztrátu danou vztahem 18 způsobenou při využití Connectionist Temporal Classification (dále jen CTC, [18, 19]). CTC určuje pravděpodobnost cílové sekvence $y$, uvážením všech možných zarovnání znaku $a$ při délce vstupu $T$. Pro výstup z lineární projekční vrstvy (klasifikační hlava, logits $x_L$)

a cílovou sekvenci $y$ je pravděpodobnost definována jako:

$$\mathrm{P}_{CTC}(y|x_L) = \sum_{a \in \beta(y)} \mathrm{P}(a|x_L) \tag{16}$$

kde $\beta(y)$ je množina všech možných zarovnání $a$ v cílové sekvenci $y$ o délce $T$ (včetně interně používaného tokenu [UNK]). Pravděpodobnost $\mathrm{P}(a|x_L)$ může být rozložena na faktory (časové úseky v časech $t$), jelikož jsou podmíněně nezávislé. Poté lze tedy rozepsat:

$$\mathrm{P}(a|x_L) = \prod_{t=1}^{T} \mathrm{P}(a(t)|x_L(t)) \tag{17}$$

kde $a(t)$ a $x_L(t)$ je $t$-tá reprezentace symbolu ve výstupní sekvenci a $t$-tým vektorem logits.

Tedy během dotrénování je minimalizována ztráta CTC funkcí:

$$\mathcal{L}_{CTC} = -\log \mathrm{P}_{CTC}(y|x_L) \tag{18}$$

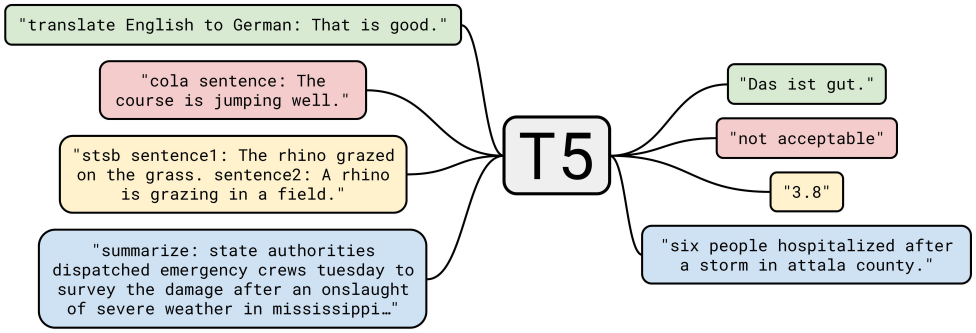
a při validaci nebo testování je využit "greedy"přístup, kdy je vybráno nejpravděpodobnější zarovnání pomocí funkce `argmax`.

V průběhu dotrénování jsou váhy výstupní lineární klasifikační vrstvy a transformerového enkodéru optimalizovány právě zmíněnou CTC ztrátovou funkcí, ale váhy příznakového enkodéru se typicky nechávají zafixované. Původní práce [1] také uvádí možnost trénování pouze klasifikační vrstvy v první fázi (prvních 10k kroků) a ve zbytku trénování se dále k optimalizaci přidávají i parametry transformerového enkodéru, avšak váhy příznakového enkodéru zůstávají stále zafixovány.

# 4 Model T5

Model T5, Text-To-Text Transfer Transformer, byl představen v práci [3], jako možnost převedení všech textově založených jazykových úloh do text-to-text formátu (například úlohy sumarizace, klasifikace, překlad, apod.). Vstupem do modelu je tedy nějaký textový kontext s úlohou a jeho výstupem bude vyprodukovaný výsledek opět v textové formě. Je využito předtrénování na značně velké sadě přirozeného textu, kdy se následně pro dílčí úlohy využívá možnosti přenesení získaných znalostí o přirozeném jazyce a dotrénování modelu pro danou úlohu. Tento přístup je označován jako transfer learning a ukázal se jako velmi účinným v doméně NLP.

Výsledkem práce [3] byl tedy průzkum a návrh tzv. Text-to-Text frameworku, jenž sjednocuje úlohy NLP převedených do formátu text-to-text a je tvořen jediným základním modelem, který má stejné váhy, hyperparametry a ztrátovou funkci (stejné předtrénování). Poté se tedy využívá možností transfer learningu a dotrénování pro specifické úlohy, kdy se během dotrénování přidává ke vstupní sekvenci $prefix$, kterým se rozlišují jednotlivé úlohy.




Obrázek 6: Diagram zobrazující několik možných úloh NLP převedených do text-to-text formátu (převzato z [3]).

## 4.1 Architektura T5 modelu

### 4.1.1 Prozkoumávané varianty architektury modelu v originální práci

Hlavním účelem původní práce [3] byly experimenty s různou variací původní architektury transformerů (popsaných v kapitole 2 a podrobněji v práci [4]) a způsobu maskování vstupů do attention mechanismu, které je hlavním rozlišujícím faktorem. Jejich předtrénování na neoznačených datech bylo prováděno na úlohách "odšumění" (*denoising*, dále v kapitole 4.2.1) a autoregrese (tato úloha byla v práci zahrnuta pouze z důvodu retrospektivního porovnání s dřívějšími přístupy pro standardní LM).

Operace self-attention mechanismu definované ve vztahu 1, zjednodušeně, vezme vstupní sekvenci a vytvoří novou výstupní sekvenci o stejné délce, jejíž hodnoty jsou vypočtené jako vážený průměr. Váhy jsou reprezentovány normalizovaným attention skóre daným maticemi $Q$ a $K$, které porovnávají daný token v sekvenci (řádek $Q$) s ostatními tokeny (sloupce $K^T$), a vážené hodnoty ze vstupní sekvence jsou reprezentovány řádky matice $V$. Poté si lze maskování představit jako vynulování těchto vah $W = \text{softmax}((QK^T)/\sqrt{d_k})$ pro takové vstupy $j$ ($j$-té řádky matice $V$), které nemají ovlivňovat výstup $i$ (uvedeno pro kontext k obrázku 7a a následujícím sekcím). Dále je zde pro úplnost uveden stručný popis prozkoumávaných variant architektur a jimi využívaných maskování.

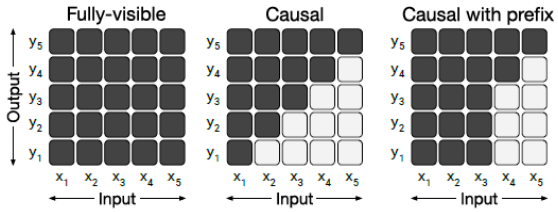
**Encoder-Decoder Transformer** vychází z původní architektury transformerů navržené v práci [4], jehož zjednodušené schéma je na obrázku 7b (vlevo).

Enkodér využívá *fully-visible* maskování (7a, vlevo), které dovoluje self-attention mechanismu využívat každý vstup $x_j$ při produkci libovolného výstupu $y_i$. Toto maskování je vhodné při využívání *prefix*u, který dodává vstupní sekvenci potřebný kontext pro danou úlohu.

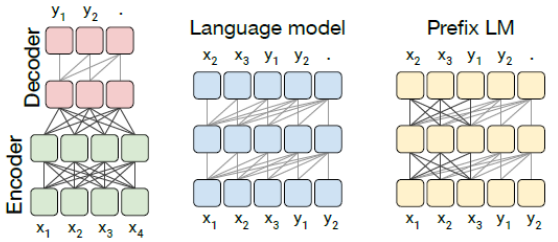
Dekodér užívá *causal* maskování (7a, uprostřed), jenž zabraňuje modelu při produkci $i$-tého výstupu využívat $j$-té vstupy, pro které platí $j > i$. Tedy při dekódování zabraňuje modelu "vidět do budoucnosti"a postupně produkuje výstupy $y_i$ pouze z předcházejících vstupů $x_j$. Což je typické a požadované chování pro LM.

**Language model** obsahuje pouze transformerový dekodér, který autoregresivně vytváří výstupní sekvenci, tedy každý následující krok je produkován ze vstupu a všech minulých predikovaných výstupů. Tato varianta, zobrazená na obrázku 7b (uprostřed), je využívána jako LM, který je trénován za účely predikce dalšího kroku (autoregrese). LM varianta modelu opět využívá *causal* maskování (7a, uprostřed).

**Prefix LM** je svojí strukturou velmi podobný předchozímu LM (vyobrazen na obrázku 7b, vpravo). Využívá pouze transformerový dekodér, ale je aplikováno modifikované maskování *causal with prefix* (7a, vpravo). Toto maskování odstraňuje některé limitace *causal* maskování u sekvencí, kde je žádoucí věnovat zvýšenou pozornost počáteční části sekvence, označované jako *prefix*. Tedy při zpracovávání *prefix* části je využito *fully-visible* maskování a na zbytek sekvence je aplikována *causal* maska.




(a) Matice reprezentující různé nastavení maskování pro self-attention mechanismus. Světlé buňky označují zamaskované vstupy pro self-attention mechanismus.




(b) Zkoumané varianty Transformer architektury v práci [3]. Tmavé spojnice mezi buňkami reprezentují *fully-visible* maskování a světlé reprezentují *causal* maskování.

Obrázek 7: Zkoumané struktury modelu T5 (převzato z [3])

Následně byly modely porovnávány na základě několika metrik používaných v úlohách pro NLP, a to GLUE a SuperGLUE (zhodnocení obecného porozumění jazyka, [20]), BLEU (metrika hodnotící strojový překlad, [21]), ROUGE (automatické vyhodnocení kvality sumarizace, [22]) a některé další, které byly specifické pro úlohy řešené v práci [3]. Také byla brána v potaz komplexita použité architektury a to v podobě počtu vrstev a trénovatelných parametrů, ale i výpočetní náročnost a rychlost prováděných operací.

Tabulka 2 v práci [3] (strana 20) uvádí dvojici výsledků jednotlivých modelů, popsaných výše, pro *denoising* a autoregresní úlohy. Dále zde byly testovány dvě další verze architektury enkodér-dekodér, a to se sdílenými parametry enkodéru a dekodéru, a se sníženým počtem vrstev v blocích enkodéru a dekodéru.

Nejvyššího skóre přes všechny metriky pro *denoising* úlohu dosáhla plná verze architektury enkodér-dekodér transformeru. Tato architektura dosáhla i srovnatelné výpočetní náročnosti jako varianta LM (pouze dekodér), ačkoli obsahuje dvakrát více parametrů. Tedy jako základním modelem pro text-to-text framework byla zvolena architektura enkodér-dekodér transformer, která bude podrobněji popsána v následující kapitole.

### 4.1.2 Použitá varianta architektury pro model T5

Model T5 implementuje enkodér-dekodér transformer, který je založen na jeho původní architektuře navržené v práci [4] a popsané v dřívější kapitole 2. Odlišuje se od původní architektury transformerů využitím maskování v self-attention blocích, čehož je využíváno hlavně v dekodéru, kde se aplikuje na self-attention mechanismus *causal* maskování, které model nutí využívat pouze minulé výstupy. Také se využívá skalární relativní *positional encodings* (zavedené v originální práci [3]) na rozdíl od fixních používaných v původní architektuře transformerů.

Jelikož v této práci byla využívána BASE varianta T5 modelu, tak i zde budou popsány pouze její parametry. Enkodér a dekodér jsou složeny každý z 12 bloků obsahující self-attention mechanismus, FFN a v případě bloků dekodéru i enkodér-dekodér attention mechanismus. Transformerové bloky mají dimenzi $d_{model} = 768$, každý attention mechanismus je složen z 12 hlav, matice $K$ a $V$ self-attention mechanismu nabývají dimenze $d_{kv} = 64$ a FFN má skrytou dimenzi $d_{ff} = 3072$. Další varianty a jejich stručné popisy lze nalézt na stránkách 36 a 37 v originální práci [3].

## 4.2 Předtrénování a použitý model

### 4.2.1 Předtrénování T5 modelu

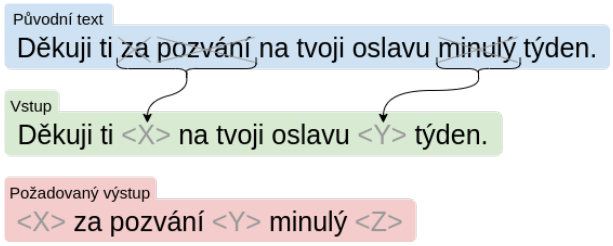
V práci [3] dále byly prozkoumávány přístupy k předtrénování s různými úlohami využívající učení bez učitele (unsupervised learning) na neoznačených datech. Volba úlohy při předtrénování má velký vliv na dosažitelné výsledky v úlohách, na kterých je model následně dotrénován, jelikož je tímto poskytován mechanismus, skrze který model získává obecné znalosti a porozumění přirozeného jazyka. Tedy dochází k využití možnosti přenosu znalostí získaných během předtrénování (transfer learning) a pro dotrénování na specifickou úlohu je možné využít menšího obnosu dat.

Následující tabulka uvádí některé prozkoumávané úlohy unsupervised learningu, kdy některé jsou kombinací obvyklých přístupů a nebo byly modifikovány za účely vyhovění text-to-text formátu.

Tabulka 3: Příklady pro použité úlohy na ukázkovém textu. Jsou zde uvedeny jejich vstupy s požadovanými výstupy ([3], upraveno). Kde <M> označuje token sdílené masky (pro všechny maskované tokeny má stejné ID), dále <X>, <Y> a <Z> jsou různé unikátní maskovací tokeny (i s různými ID), a zaměněné tokeny jsou označeny šedou barvou.

| Původní text | Děkuji ti za pozvání na tvoji oslavu minulý týden. | |
|---|---|---|
| Úloha | Vstup | Požadovaný výstup |
| Prefix LM | Děkuji ti za pozvání | na tvoji oslavu minulý týden. |
| BERT přístup [9] | Děkuji ti <M> <M> na tvoji oslavu jablko týden. | (*Původní text*) |
| Přeskupení | oslavu za tvoji na. minulý ti pozvání týden Děkuji | (*Původní text*) |
| MASS přístup [23] | Děkuji ti <M> <M> na tvoji oslavu <M> týden. | (*Původní text*) |
| Nahrazení mezer | Děkuji ti <X> na tvoji oslavu <Y> týden. | <X> za pozvání <Y> minulý <Z> |
| Vypuštění tokenů | Děkuji ti na tvoji oslavu minulý týden. | ti za pozvání |
| Náhodné mezery | Děkuji ti <X> na <Y> týden. | <X> za pozvání <Y> na tvoji oslavu minulý <Z> |

Z prováděných experimentů nad přístupy k unsupervised learningu, popsaných v tabulce 3, vycházely úlohy založené na podobném přístupu využívaném při předtrénování modelu BERT, jako ty dosahující lepších výsledků. Proto byl dále zkoumán přístup *denoising*, jehož princip je znázorněn v tabulce 3 (poslední 3 řádky) a na obrázku 8.




Obrázek 8: Schématické znázornění přetrénování na úloze *denoising* ([3], upraveno).

Na úloze *denoising* byla pozornost věnována zejména míře "poškození"(*corruption rate*) a maximální možné délce, která může být takto "poškozena"(*corruption span*). Tedy *corruption rate* udává procento zamaskovaných slov ve vstupní sekvenci. Porovnávali se *corruption rates* 10%, 15%, 25% a 50%, kdy bylo zjištěno, že užití vyššího procenta *corruption rate* mělo za následek degradaci v některých metrikách a potencionální prodloužení doby trénování, kdy je nutné předpovídat delší požadované výstupy. Na základě těchto zjištění byla využita *corruption rate* 15%, tato volba byla také ovlivněna modelem BERT, který užíval stejné procento poškození.

Dále byla uvažována *corruption span*, kdy je náhodně zamaskovaná souvislá část vstupní sekvence jedním unikátním tokenem. Tato úloha bývá parametrizována výše zmíněnou *corruption rate* a maximálním počtem takto poškozených částí. Přístup s *corruption span* dosahuje podobných výsledků jako při použití pouze *corruption rate*, ale poskytuje nepatrné urychlení během trénování, jelikož v průměru jsou produkovány kratší sekvence v požadovaném výstupu.

Pro předtrénování byla tedy využita úloha *denoising* s *corruption span*, jejíž schématické znázornění je na obrázku 8. Jako příklad se uváží uváděný původní text, "Děkuji ti za pozvání na tvoji oslavu minulý týden.", kdy se po konverzi na tokeny obdrží tyto identifikátory: $[5262, 548, 56, 18580, 27, 1211, 152, 19663, 9736, 2598, 31859, 1]$. Sekvence se skládá ze 12 tokenů, kdy při *corruption rate* 15% bude počet zamaskovaných tokenů přibližně roven $12 \cdot 0.15 \approx 1.8$ a určí-li se, že počet *corruption span* má být 2, tak lze určit průměrný počet tokenů v jedné *corruption span* odpovídající $1.8/2 \approx 0.9$.

**Dataset C4** Colossal Clean Crawled Corpus, je označení pro anglický dataset určený pro NLP, který vznikl právě za účely předtrénování modelu T5. Tento dataset vychází z již existujícího datasetu Common Crawl Corpus, který je veřejně dostupným archivem extrahovaného textu z webových stránek. Takovýto text samozřejmě není "čistý", bude obsahovat značkování, HTML tagy a vložené JavaScript skripty. Proto na základě několika heuristických úvah, jako například ukládání pouze řádků končící interpunkčními znaménky, odstranění částí obsahující "{}"závorky, odstranění ne-anglického textu, apod. (celý seznam s vysvětlením lze nalézt v původní práci [3], kapitola 2.2), byl Common Crawl Corpus filtrován, čímž dal vzniku C4 datasetu obsahující přibližně 750GB "čistého"a přirozeného anglického textu.

### 4.2.2 Implementace a použitý model

Pro model T5 byla využita jeho BASE varianta, jehož architektura byla popisována v kapitolách 2, 4.1.2 a původní práci [3]. Pro konzistenci byla použita jeho implementace ve frameworku PyTorch a obalují abstrakce prostřednictvím knihovny `transformers`, jako tomu bylo v případě dříve zmiňovaného Wav2Vec2.0 modelu v kapitole 3.2.2.

Předtrénovaný model T5 byl poskytnut opět panem Ing. Janem Lehečkou, Ph.D., který byl součástí práce [24] zabývající se automatickým sběrem, zpracováním a ukládání velkých textových dat z webových stránek, na nichž byl model předtrénován. V době kdy byl model trénován (2021), dataset obsahoval přibližně 20GB čistého textu, získaného z 8.7 milionů internetových novinových článků. Ještě před samotným předtrénováním je také nutné vytvořit tokenizer, který bude dělit vstupní sekvence na tokeny, jenž budou uloženy ve slovníku o zvolené velikosti. Tokenizer a slovník je vytvářen z trénovací sady dat pomocí SentencePiece modelu [25], který je typicky využíván, když je určená maximální velikost slovníku. Velikost slovníku byla zvolena na 32k tokenů, které byly vytvořeny ze zmiňovaných 8.7 milionů článků, dohromady obsahujících přibližně 2.9 miliard slov.

## 4.3 Dataset využitý pro dotrénování

Model T5 byl dotrénován na uměle vytvořeném datasetu z VoxPopuli [26]. VoxPopuli dataset je veřejně dostupná sbírka záznamů s jejich přepisy z evropského parlamentu mezi lety 2009 až 2020, která byla vytvořena společností Meta AI (Facebook) v roce 2021. Dataset obsahuje přibližně 62 hodin označených záznamů české promluvy.

Ačkoliv je tento dataset primárně určen pro dotrénování ASR modelů, tak bylo zapotřebí získat vhodnou reprezentaci vstupu, kterou by mohl model T5 očekávat po rozpoznaném přepisu od Wav2Vec2.0 modelu. Z tohoto důvodu byla jeho celá česká část (*train*, *validation* a *test* porce) přepsána jedním z dříve natrénovaných modelů Wav2Vec2.0 (byla využita verze v23, výsledky v tabulce 5), kdy byla získána reálná reprezentace možných výstupů z modelu Wav2Vec2.0, tedy kombinace správně a do různé míry špatně rozpoznaných přepisů. Čímž vznikly vstupní trénovací data, kdy jejich očekávané výstupy (reference) byly převzaty z původního VoxPopuli datasetu, které byly dále normalizovány, aby odpovídaly možnostem slovníku natrénovaného Wav2Vec2.0 modelu (popisováno v kapitole 5.3).

Tabulka 4: Základní statistiky české porce dat z modifikované verze Voxpopuli datasetu společně s metrikami WER a CER (více kapitola 5.2) vyhodnocující přesnost rozpoznaných přepisů modelem Wav2Vec2.0 verze v23 použité při vytváření datasetu.

| | *train* | *validation* | *test* |
|---|---|---|---|
| # příkladů [1] | 18902 | 1103 | 1123 |
| prům. délka vstupu [token] | 28.83 | 28.53 | 27.86 |
| prům. délka výstupu [token] | 27.93 | 27.61 | 26.91 |
| Wav2Vec2.0 v23 WER [%] | 15.10 | 15.00 | 15.56 |
| Wav2Vec2.0 v23 CER [%] | 6.48 | 6.39 | 6.64 |

Výsledky dosažené na validační sadě modelem Wav2Vec2.0 lze porovnat s výsledky dosaženými modelem T5 v tabulce 6.

Během analýzy chyb prováděné nad testovací sadou VoxPopuli datasetu (více v kapitole 7) se ukázalo, že dataset není plně validovaný a obsahuje někdy značné chyby. U

extrémních výchylek v analýze byly tyto chyby zapříčiněny například prázdnou nahrávkou, ale s vyplněnou transkripcí, francouzské nahrávky s českou transkripcí nebo část nahrávky v jiném jazyce než českém s plně českou transkripcí. Takovýchto nahrávek bylo objeveno alespoň 7 v testovací porci dat s indexy příkladů v datasetu [62, 171, 273, 360, 441, 834, 888].

## 4.4 Dotrénování modelu T5

Na rozdíl od modelu Wav2Vec2.0 nemusí být model T5 ve fázi dotrénování doplňován o náhodně inicializovanou klasifikační hlavu modelu, jelikož díky text-to-text úlohám, na kterých je model předtrénováván, je již inicializovaná na transformerovém dekodéru. Tato vrstva opět slouží jako projekce výstupu, kdy BASE varianta modelu produkuje $n$ vektorů (logits, kde $n$ je počet tokenů z tokenizované vstupní sekvence) a každý je dimenze o velikosti $d_{model}$, do daného počtu tříd určeného z velikosti slovníku $C_{T5}$ a doplňkových speciálních tokenů (značky počátku a konce sekvence, maskovací tokeny). Tedy slovník $C_{T5}$ má velikost 32128, z čehož 32000 záznamů jsou tokeny získané SentencePiece modelem (viz. 4.2.2) a zbylých 128 záznamů tvoří speciální a maskovací tokeny.

Při dotrénování jsou váhy modelu optimalizovány na základě minimalizace Cross-Entropy ztrátové funkce

$$\mathcal{L}_H(p, q) = -\sum_{c=1}^{C_{T5}} p(c)\log_2(q(c)) \tag{19}$$

kde $p$ je tzv. *one-hot* vektor, který nabývá velikosti slovníku $C_{T5}$ a obsahuje všude 0 kromě pozice odpovídající správné klasifikaci, která jako jediná obsahuje 1, $q$ je opět vektor velikosti $C_{T5}$ obsahují odhadované příslušnosti do daných tříd. Jelikož je využitá implementace ve frameworku PyTorch, tak klasifikační hlava modelu T5 neobsahuje na svém výstupu `softmax` funkci a využívá Cross-Entropy funkci definovanou jako

$$\mathcal{L}_H(p, q) = [l_1, ..., l_N]^T;\ \text{kde}\ l_n = -\sum_{c=1}^{C_{T5}} w_c \ln\left(\frac{\exp(q_n(c))}{\sum_{i=1}^{C_{T5}}\exp(q_n(i))}\right) p_n(c) \tag{20}$$

kde $w_c$ je daná váha třídy (vhodné pokud některá třída je majoritně zastoupena v datech, v tomto případě je pro všechny zvolena stejná váha), $l_n$ je ztráta pro $n$-tý token v sekvenci a podobně i vektory $p_n$ a $q_n$ jsou *one-hot* vektor a vektor predikcí modelu pro $n$-tý token. Také zde přibyl parametr, který definuje jakým způsobem zredukovat vektor ztrát na skalární reprezentaci "celkové"ztráty, a to zprůměrováním nebo sumou.

Originální práce [3], také prozkoumávala různé možnosti optimalizace menšího počtu parametrů modelu během dotrénování, které by byly výpočetně efektivnější než je optimalizace všech parametrů modelu. Jednou z prozkoumávaných možností bylo zafixování vah modelu, ale přidání dalších adaptivních vrstev, které jsou FFN s volitelnou skrytou dimenzí a ReLU aktivační funkcí. Druhou zkoumanou možností bylo postupné uvolňování parametrů, které mohou být v průběhu trénování optimalizovány. Ukázalo se, že v obou zkoumaných případech dochází ke značné degradaci dosahovaných výsledků, a proto se zůstalo u optimalizace všech parametrů modelu i přes větší výpočetní náročnost.

# 5 Experimenty a dosažené výsledky

Motivací této práce bylo doplnění ASR modelu Wav2Vec2.0 o jazykový model v podobě T5. Toto spojení má vést ke zvýšení přesnosti transkripce řečového signálu právě doplněním informace o psaném přirozeném jazyce. Sekce v této kapitole pojednávají o metodologii a experimentech s dotrénováním modelů Wav2Vec2.0 a T5, ale také o využívaných metrikách a realizaci trénování na hardwaru. Veškerá relevantní data a výsledky jsou součástí repozitáře, který lze nalézt zde[1]. Další kapitolou týkající se dosahovaných výsledků je kapitola 6 popisují spojení těchto dvou modelů a kapitola 7 pojednávající o analýze chyb z hlediska měřených metrik, ale také sémantické podobnosti před a po doplnění o model T5.

## 5.1 Realizace a spouštění experimentů

Trénovací skripty využívané k přípravě dat, nastavování trénovacích parametrů a samotného řízení běhu trénování a validace, vychází z ukázkových skriptů zveřejněných v repozitáři `transformers` od HuggingFace. Zatímco trénovací skript pro model Wav2Vec2.0[2] zůstal poměrně nezměněný od jeho ukázkového skriptu[3], tak trénovací skript pro model T5[4] vycházející z ukázkového skriptu[5] úlohy řešící strojový překlad, kdy bylo nutné skript příslušně modifikovat pro úlohu řešenou v této práci. Díky podstatě řešené úlohy, která je velmi blízká úloze strojového překladu, byly nutné úpravy minimální. Za účely snadného vytváření prostředí a spouštění všech experimentů byl vytvořen Jupyter Notebook[6], ve kterém probíhala kontrola nutných balíčků používaných během tréninku, inicializace spojení se službou pro záznam průběhu tréninku (bylo využito služby Weights & Biases) a výběr trénovaného modelu s verzí argumentů pro daný běh.

Nejprve bylo nutné otestovat funkčnost trénovacích skriptů, validace a záznamu průběhu trénování. Toto bylo prováděno na osobním stroji (Lenovo Legion 5 17ACH6H), který obsahuje dedikovanou mobilní verzi grafické karty Nvidia GeForce RTX 3070 (8GB VRAM, 130W TDP). V průběhu testování byly objeveny i zajímavé poznatky o implementacích daných modelů, například u modelu Wav2Vec2.0 při nízkých `batch_size` ($\leq 16$) docházelo při zpětném běhu pravděpodobně k numerické nestabilitě a byly vraceny $NaN$ hodnoty ztrátové funkce. Obdobný případ nastával i u modelu T5, ale s povolením snížené přesnosti desetinných čísel pomocí parametru `fp16`, kdy také byly vraceny $NaN$ hodnoty ze ztrátové funkce.

Samotné experimenty trénování, popisované v kapitolách 5.3 a 5.4, byly prováděny na některých z výpočetních uzlů zastřešených organizací MetaCentrum, která na omezený čas zprostředkuje jejich vypůjčení. Na první běhy byly využívány stroje *konos* obsahující grafické karty Nvidia GeForce GTX 1080 Ti (11GB VRAM, 250W TDP). Následně pro urychlení průběhu experimentů byly využívány stroje *galdor* obsahující grafické karty Nvidia A40 (48GB VRAM, 300W TDP). Experimenty vždy byly prováděny na 1 grafické kartě z důvodu dlouhých čekacích front na přidělení požadované konfigurace stroje.

[1] https://github.com/sulcm/wav2vec2-w-T5-cs

[2] https://github.com/sulcm/wav2vec2-w-T5-cs/blob/main/run_scripts/run_speech_recognition_ctc.py

[3] https://github.com/huggingface/transformers/blob/main/examples/pytorch/speech-recognition/run_speech_recognition_ctc.py

[4] https://github.com/sulcm/wav2vec2-w-T5-cs/blob/main/run_scripts/run_seq2seq.py

[5] https://github.com/huggingface/transformers/blob/main/examples/pytorch/translation/run_translation.py

[6] https://github.com/sulcm/wav2vec2-w-T5-cs/blob/main/train_nb.ipynb

## 5.2 Evaluační metriky

Během práce, při analýze chyb nebo trénování, byly využívány dvě metriky, a to chybovost na úrovni slov (Word Error Rate, WER) a chybovost na úrovni znaků (Character Error Rate, CER), popsané ve vztazích 21. Tyto metriky jsou často využívané pro vyhodnocování úloh založených na ASR, ale jsou také vhodné pro vyhodnocení úlohy "kontroly pravopisu", kdy poskytují poměrně reprezentativní informace o přesnosti prováděných úprav. Výpočty metrik byly zprostředkovány za použití knihovny `jiwer` (při trénování byla využita knihovna `evaluate` platformy HuggingFace, ale ta opět interně využívá knihovnu `jiwer`).

Metriky WER a CER se určují aplikací algoritmu Levenshteinovy vzdálenosti, jehož výsledkem je počet nutných operací, odstranění (*deletion*, $D$), vložení (*insertion*, $I$) nebo záměně (*substitution*, $S$), k tomu, aby predikovaná slova v sekvenci (WER) nebo znaky ve slově (CER) odpovídaly jejich cílovým tvarům.

Implementace těchto metrik v knihovně `jiwer` je totožná, neboť při výpočtu CER lze uvažovat, že se jedná o sekvenci slov, kde každé je délky 1 (tedy znak). Poté vztahy pro výpočet WER a CER nabývají stejného tvaru a to

$$\begin{aligned} WER &= \frac{S+D+I}{N} = \frac{S+D+I}{H+S+D} \\ CER &= \frac{S+D+I}{N} = \frac{S+D+I}{H+S+D} \end{aligned} \tag{21}$$

kde $N$ je počet slov v cílové sekvenci pro WER nebo počet znaků v cílovém sekvenci pro CER a $H$ (hits) je počet správných odhadů, kdy $H = N - S - D$.

U takto definovaných metrik WER a CER může dojít k vyhodnocení, které je větší než 1, tedy více než 100% chybovost. K tomuto případu může například dojít pokud predikovaná sekvence má značně vetší počet slov nebo znaků než její reference. Jako příklad je zde uvedený přepis s korekcí sekvence v testovací porci datasetu Common Voice s indexem 14, kde reference je "pokusím se"a ze záznamu promluvil model Wav2Vec2.0 odhadl přepis "pokus ím zde", jenž model T5 následně upravil na "pokus jsem zde". Obě tyto predikce dosáhly WER 150%.

## 5.3 Experimenty s modelem Wav2Vec2.0

V kapitole popisující dotrénování Wav2Vec2.0 modelu 3.4 bylo řečeno, že se náhodně inicializuje lineární vrstva, která vytváří projekci z výstupních vektorů v časových úsecích $t$ o dimenzi 768 do slovníku $C_{W2V2}$, ze kterého se poté vybere nejpravděpodobnější token přidělený úseku $t$. Tedy před spuštěním vlastního trénování bylo nutné vytvořit slovník $C_{W2V2}$, který bude obsahovat nejmenší nutný počet záznamů. Data prošla předzpracováním, jenž zahrnovalo odstranění nežádoucích znaků (definována ve spouštěcích skriptech) a normalizací pouze na malá písmena. Výsledkem byl slovník $C_{W2V2}$ obsahující 55 záznamů, to znamená celá česká abeceda s diakritikou a doplňkovými tokeny, které model interně využívá.

Během experimentů s dotrénováním Wav2Vec2.0 modelu byla pozornost věnována parametrům `learning_rate` (dále jen `lr`), regularizace a maskování (oddálení "přeučení", snaha o schopnost zobecňovaní), počet kroků `max_steps`, velikost sady předložené v jednom kroku `batch_size` (určeno součinem parametrů `per_device_train_batch_size` a `gradient_accumulation_steps`) a počet kroků "zahřívání"`warmup_steps`. Všechny experimenty byly prováděny se zafixovanými vahami příznakového enkodéru parametrem

`freeze_feature_encoder`. Volba některých parametrů vycházela z původní práce [1], jako například regularizace a maskování, které byly popisovány v dodatcích A a B práce [1], `max_steps` a `batch_size`. Přesné nastavení všech parametrů lze nalézt v přiložených spouštěcích skriptech pro Wav2Vec2.0 model[7].

Metodika volby a velikosti změny zvoleného parametru vycházela z určení intervalu, ve kterém by bylo možné dospět do rozumného lokálního minima ztráty na daných trénovacích datech. Jak již bylo zmíněno, vycházelo se z parametrů popisovaných v původní práci [1] a společně s parametry v repozitáři frameworku Fairseq[8], se kterými byly modely v originální práci trénovány. Poté tyto parametry byly zvoleny jako výchozí bod (střed intervalu), kde dolní a horní mez byla typicky zvolená jako $\pm$ řád parametru nebo jeho násobek/podíl daným koeficientem. Běhy trénování vycházely ze zvoleného středu intervalu, kdy se poté přecházelo na dolní a horní meze intervalu, porovnáním hodnot metrik dosažených po dotrénování z těchto třech běhů, bylo rozhodnuto do jaké části intervalu se přesune další experiment. Tedy přesněji došlo ke zvolení nového středu mezi původním středem a dolní nebo horní mezí intervalu. Tímto způsobem byly iterovány parametry jako `lr` a maskování, které je ovlivňováno parametry `mask_feature_prob`, `mask_feature_length`, `mask_time_prob` a `mask_time_length`. Některé parametry byly v průběhu experimentů postupně zvyšovány, jako `batch_size`, `max_steps` a `warmup_steps`, které bylo typicky zvoleno na 8% z parametru `max_steps`. Zbytek parametrů týkající se regularizace (různé `dropout` pravděpodobnosti v daných vrstvách modelu) a způsobu nájezdu `lr`, které lineárně narůstalo po první kroky dané `warmup_steps` a následně lineárně klesalo po zbytek trénování, zůstaly v průběhu všech experimentů nezměněny.

Experimenty byly prováděné do doby dosažení 80000 kroků při `batch_size`=256, které byly také uvedeny v práci [1]. Speciálně volba `batch_size` vycházela z požadovaného počtu minut předkládaného v jednom trénovacím kroku, kdy v originální práci [1] bylo uvedeno přibližně 26.67 minut audia v jedné dávce. Z tohoto důvodu pro trénovací sadu Common Voice, která je využívána v této práci a jejíž průměrná délka nahrávek je přibližně 4.53 sekund, bylo nutné zvolit pro dosažení obdobného množství audia v jednom kroku na 256 `batch_size`. Což v průměru odpovídalo $(4.53/60) \cdot 256 \approx 19.33$ minut audia v jednom trénovacím kroku.

[7] https://github.com/sulcm/wav2vec2-w-T5-cs/tree/main/run_scripts/configs/wav2vec2

[8] https://github.com/facebookresearch/fairseq/tree/main/examples/wav2vec/config/finetuning

Tabulka 5: Výsledky trénování Wav2Vec2.0 modelu s různými konfiguracemi. Použité parametry pro jednotlivé běhy lze nalézt ve stejnojmenných souborech v repozitáři. Váhy výsledného modelu jsou určeny z uloženého checkpointu s nejnižším WER na validační části dat.

| Verze | WER [%] | CER [%] | `lr` [1] | `batch_size` [1] | `max_steps` [1] |
|---|---|---|---|---|---|
| v3* | 12.63 | 2.92 | 8e-5 | 128 | 20000 |
| v4* | 13.42 | 3.07 | 7.5e-5 | 128 | 20000 |
| v5* | 12.90 | 2.95 | 5e-5 | 128 | 20000 |
| v6* | **11.58** | **2.66** | 5e-5 | 128 | 20000 |
| v7* | 11.73 | 2.71 | 5e-5 | 256 | 20000 |
| v8* | 11.71 | 2.79 | 5e-5 | 256 | 20000 |
| v13 | 14.80 | 3.29 | 2e-5 | 256 | 40000 |
| v16 | 14.83 | 3.37 | 2e-5 | 256 | 40000 |
| v17 | 14.13 | 3.23 | 2e-5 | 256 | 40000 |
| v20 | 14.02 | 3.16 | 1e-5 | 256 | 40000 |
| v22 | 13.85 | 3.16 | 9e-6 | 256 | 80000 |
| v23 | **13.85** | **3.16** | 1e-5 | 256 | 80000 |

Z výsledků lze vidět, že model dosahuje vyšší přesnosti při použití většího množství označených trénovacích dat, jehož verze s navýšenou trénovací sadou jsou označeny *, úprava dat je více popsaná v kapitole 3.3. Samozřejmě stejného závěru bylo dosaženo i v originální práci [1], kdy při použití celé Librispeech sady (960 hodin) bylo dosaženo na testovací sadě 1.8% WER. Ovšem důležitějším výsledkem bylo, že i při použití relativně malé sady (10 minut) bylo dosaženo 9.1% WER, což znamenalo dosažení lepších výsledků než u některých původních přístupů, u kterých bylo zapotřebí použít značně většího množství označených dat.

## 5.4 Experimenty s modelem T5

Jelikož trénovací dataset pro model T5 vychází z datasetu VoxPopuli, který byl přepsán natrénovaným Wav2Vec2.0 modelem (podrobněji v kapitole 4.3), tak obdobně i tento dataset musel být normalizován. Tato normalizace zapříčinila, že všechny predikce modelu T5 neobsahují interpunkci, velká písmena, číslice a případné další speciální znaky. V této práci, kde využívaný tokenizer a slovník modelu T5, již obsahuje záznamy pro tyto druhy znaků, by bylo vhodnější kdyby takové záznamy neobsahoval a mohlo tak dojít k efektivnějšímu rozdělení tokenů, jelikož nedojde k jejich využití.

Metodika a výběr parametrů během dotrénování T5 modelu byl velmi blízké tomu používaném při dotrénování modelu Wav2Vec2.0. Opět se vycházelo z parametrů uvedených v původní práci [3] pro `lr`, regularizace `dropout` vrstev, `batch_size`, `max_steps` a `warmup_steps`. Avšak v tomto případě nebylo vhodné využívat některých nastavení parametrů uvedených v původní práci. Například nebylo možné držet konstantní `lr` po celou dobu trénování nebo používat vyšší `batch_size` a `max_steps`, kdy poměrně rychle docházelo k přetrénování. Snaha byla toto napravit snížením `lr`, což nevedlo k nápravě, ale typicky k zanedbatelné změně v měřených metrikách (WER i CER se stále pohybovaly v blízkosti $\approx 1$, tedy nedocházelo k trénování) nebo k nestabilnímu průběhu ztrátové funkce a měřených metrik. Pravděpodobným zdrojem těchto problémů byl využitý dataset, jenž má být primárně využíván pro trénování ASR modelů, a který má menší počet příkladů než typický dataset pro NLP úlohy.

Model T5 poskytoval možnost modifikovat konfiguraci generování výstupu, kdy bylo možné využít beam-search algoritmus na místo primitivního greedy přístupu pomocí parametru `num_beams`. Využití beam-search algoritmu při dekódování bylo popisováno i v originální práci [3], kdy docházelo ke zlepšení podávaných výsledků na delších sekvencích. Proto obdobně jako v původní práci, tak i v této práci byl využit beam-search algoritmus se šířkou prohledávání nastavenou na 4. Při zvyšování šířky prohledávání během testování modelu nedocházelo k významnému zlepšení zkoumaných metrik, ale docházelo k nepatrnému prodloužení doby inference, což vedlo k provedení kompromisu a ponechání parametru `num_beams` nastaveného na 4. Další poznatek z testování přinesl parametr `do_sample`, který se typicky používá pro úlohy generování a využívá možnosti výběru jakéhokoli tokenu ze slovníku na základě vypočítaného pravděpodobnostního rozložení přes celý slovník. Motivací pro využití tohoto parametru byla možnost pomoci dosahovat vyšší sémantické podobnosti (více v kapitole o analýze chyb 7.3), kdy dojde k nesrozumitelnému přepisu části sekvence modelem Wav2Vec2.0 a model T5 má tak možnost většího výběru z různých alternativ.

Nastavení veškerých využitých parametrů a společně s generačními konfiguracemi pro všechny trénovací běhy experimentů lze nalézt v přiložených spouštěcích skriptech pro model T5[9].

Tabulka 6: Výsledky trénování T5 modelu s různými konfiguracemi. Použité parametry pro jednotlivé běhy lze nalézt ve stejnojmenných souborech v repozitáři. Váhy výsledného modelu jsou určeny z uloženého checkpointu s nejnižším WER na validační části dat.

| Verze | WER [%] | CER [%] | `lr` [1] | `batch_size` [1] | `max_steps` [1] |
|---|---|---|---|---|---|
| v1 | 9.87 | 5.75 | 3e-4 | 16 | 10000 |
| v3 | 9.61 | 5.41 | 1e-4 | 32 | 10000 |
| v4 | **9.11** | **5.17** | 3e-4 | 32 | 10000 |
| v9 | 9.39 | 5.61 | 3e-4 | 16 | 20000 |
| v12 | 9.68 | 5.66 | 6e-4 | 128 | 20000 |
| v16 | 9.40 | 5.59 | 3e-4 | 128 | 20000 |
| v17 | 9.64 | 5.92 | 1e-4 | 128 | 20000 |
| v19 | 9.34 | 5.61 | 3e-4 | 128 | 20000 |
| v20 | 9.43 | 5.58 | 3e-4 | 128 | 30000 |

Zde je poskytnuta evidence k tvrzením z předchozích odstavců, kdy si lze všimnout trendu, který zvyšováním `batch_size` a `max_steps` nevede ke zlepšení výsledků a také pravděpodobně dochází ke ztrátě schopnosti zobecňovat nebo-li přetrénování. Alespoň se to takto jeví z validační sady VoxPopuli datasetu, ale při následném testování (v sekci 6.2) nebyla ovšem tato domněnka potvrzena výsledky dosahovanými na neviděných datech. Například s verzí v20 modelu T5 bylo dosaženo srovnatelných výsledků na měřených metrikách (viz. tabulka 7). O následném testování a výsledcích dosažených po spojení, pojednávají kapitoly 6 a 7 týkající se analýzy chyb.

[9] https://github.com/sulcm/wav2vec2-w-T5-cs/tree/main/run_scripts/configs/t5

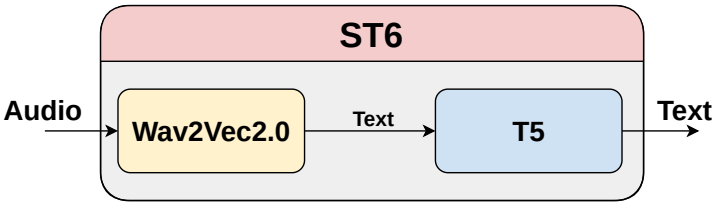
# 6 Spojení modelů Wav2Vec2.0 a T5

## 6.1 Popis funkce

Jak již může z obsahu této práce vyplývat, navržené řešení není *end-to-end* přístupem, kde je vytvářen jeden komplexní systém, který obchází nutnost mezivrstev nesoucí pouze dočasnou nebo zprostředkovanou informaci pro následné zpracování. V případu této práce dochází ke dvěma konverzím, a to řeč na text (Wav2Vec2.0, 3) následovanou konverzí text na text (T5, 4). Kombinace těchto dvou modelů bude nadále v práci označována jako ST6 (Speech-To-Text-To-Text-Transfer-Transformer).

ST6 poskytuje možnost samostatnému Wav2Vec2.0 modelu doplnit znalosti o psaném přirozeném jazyce pomocí LM v podobě T5 modelu. Což může mít za následek kompletní nebo částečnou opravu vytvořené transkripce promluvy. Samozřejmě se také přidává prostor pro zanesení více chyb, jak na již chybné transkripci, tak i na těch správných. Tento přístup poskytuje poměrně snadný způsob analýzy chyb a vlivu LM na ASR, díky výše zmíněné mezivrstvě textu mezi modely Wav2Vec2.0 a T5. Toto je samozřejmě jeden možný způsob pohledu na chyby modelu, kdy se striktně porovnávají predikované a požadované výstupy za použití zvolených metrik WER a CER (tabulka 7) a následná analýza častých nebo očekávaných chyb (sekce 7.2). Dalším možným pohledem, díky použití LM, je průzkum jejich významové (sémantické) blízkosti. Tedy nemusí nutně dojít ke korektní predikci z hlediska metrik WER a CER, ale i přesto může dojít k přiblížení v jejich významové podobě. Toto přiblížení může mít stejně velký vliv u asistenčních systémů, kde je nutné přenášet i význam zprávy od uživatele, jako by bylo dosahování lepších výsledků v metrikách WER a CER. Z tohoto důvodu je tento pohled dále prozkoumáván v sekci 7.3.

Pro účely jednodušší inicializace a inference byla vytvořena obalující třída pro ST6[10], která umožňuje výběr libovolné kombinace modelů (i samostatný Wav2Vec2.0 model). Dále poskytuje možnost volby konfigurace generování pro model T5 a zařízení, na kterém budou modely provádět inferenci.




Obrázek 9: Schématické zobrazení funkce ST6.

## 6.2 Výsledky dosažené po spojení

ST6 byl testován na neviděných porcích z datasetů využívaných pro trénování modelů Wav2Vec2.0 a T5, tedy *test* části z Common Voice 11 (3.3) a VoxPopuli (4.3) datasetů. Toto vzájemné porovnání bylo zejména důležité kvůli specifickému obsahu datasetu VoxPopuli, čímž se značně odlišuje od obecnějšího Common Voice datasetu, kdy docházelo k neproporcionálně lepším výsledkům po doplnění T5 modelu.

Dále byly zkoumány vlivy konfigurace generování modelem T5, na které již bylo poukázáno v kapitole 4.4 týkající se jeho dotrénování. Následující tabulka obsahuje po-

[10]https://github.com/sulcm/wav2vec2-w-T5-cs/blob/main/asr_w_spellchecker.py

rovnání dosahovaných výsledků samotným Wav2Vec2.0 modelem a s následně přidaným T5 modelem.

Tabulka 7: Porovnání výsledků dosažených samotným modelem Wav2Vec2.0 (verze v23, první řádka) s doplněnými různými verzemi T5 modelu a jeho lišícími se konfiguracemi generovaní.

| Model | Common Voice (*test*) | | VoxPopuli (*test*) | | T5 gen. konfigurace | |
|---|---|---|---|---|---|---|
| | WER [%] | CER [%] | WER [%] | CER [%] | `num_beams` | `do_sample` |
| W2V2v23 | 15.31 | **3.61** | 15.51 | 6.65 | - | - |
| + T5v4 | 14.00 | 6.46 | 10.66 | 6.43 | 1 | zakázáno |
| + T5v4 | 14.06 | 6.46 | 10.72 | 6.48 | 4 | povoleno |
| + T5v4 | 14.08 | 6.51 | 10.69 | 6.46 | 4 | zakázáno |
| + T5v4 | 14.00 | 6.48 | 10.74 | 6.51 | 16 | povoleno |
| + T5v20 | 13.94 | 6.48 | 10.59 | 6.28 | 1 | zakázáno |
| + T5v20 | **13.92** | 6.46 | **10.57** | **6.28** | 4 | povoleno |
| + T5v20 | 14.02 | 6.63 | 10.60 | 6.30 | 4 | zakázáno |

Ze všech testovaných kombinací vychází WER o značnou míru lépe než při použití samostatného Wav2Vec2.0 modelu. Jediné vychýlení se zdá být v metrice CER na datasetu Common Voice s použitím pouze Wav2Vec2.0 modelu. Pro ověření byl tento test opakován, ale bylo dosaženo stejných výsledků. Vysvětlení tohoto vychýlení pravděpodobně poskytuje sekce 7.2 popisující zaváděné typické chyby a specificky její část obsahující obrázek 10, na němž si lze všimnout nárůstu četnosti chyb ve třídě *other* na všech zkoumaných operacích. Do této třídy, po bližší analýze, často spadají právě chyby ve znacích nebo částech slov, což by mohlo být příčinou tohoto rozdílu v metrice CER.

Z vlivů konfigurace lze vidět, že ve většině případů, kdy je povoleno `do_sample`, dochází k statisticky nevýznamnému zlepšení v měřených metrikách. Počet `num_beams` nemá příliš velký vliv na dosahované výsledky, ale co již v tabulce není uvedeno, došlo k prodloužení doby inference s použitím vyšších hodnot `num_beams` (například 16). Proto ve všech následujících testech a analýzách je uvažována konfigurace `num_beams` 4 a povolené `do_sample`, pokud není uvedeno jinak.

# 7 Analýza chyb

V této kapitole se prozkoumávají různé pohledy na chyby zaváděné modely Wav2Vec2.0 a T5. Analýzy a testy se provádějí nad neviděnými porcemi dat z datasetů Common Voice a Voxpopuli, tedy jejich *test* části. Analýza chyb byla prováděna v jediném vytvořeném Jupyter Notebooku[11], jelikož zkoumané úlohy byly často provázány a takto bylo možné i lépe porozumět případným souvislostem.

## 7.1 Porovnání četností a fáze vzniku chyb

Sloučením modelů Wav2Vec2.0 a T5 do sekvence, se samozřejmě také slučují jejich chyby zavedené do predikce. Pro jednoduché zjištění četností možných kombinací chyb, které mohou vzniknout tímto spojením, bylo prováděno na základě porovnání metrik WER dosažených v predikci Wav2Vec2.0 a T5 modelem. Došlo k definicím množin popisujících vývoj predikovaných výstupů v jednotlivých fázích průchodu ST6 s očekávaným výstupem. Tímto vzniklo 6 možností vývoje predikce, které mohou nastat, a jejichž popisy s četnostmi lze nalézt v následující tabulce.

Tabulka 8: Četnosti predikcí výstupů v jednotlivých fázích ST6 na základě porovnání WER metriky. Všechny použité verze modelu T5 měly nastavenou stejnou konfiguraci generování (`num_beams`=4 a povolené `do_sample`).

| Wav2Vec2.0 přepis | T5 korekce | Popis chyby | Common Voice (*test*) | | VoxPopuli (*test*) | |
|---|---|---|---|---|---|---|
| | | | v23 + v4 | v23 + v20 | v23 + v4 | v23 + v20 |
| Správně | Správně | $WER_{W2V2} = 0.0 \wedge WER_{T5} = 0.0$ | 3464 | 3516 | 231 | 230 |
| | Špatně | $WER_{W2V2} = 0.0 \wedge WER_{W2V2} < WER_{T5}$ | 260 | 208 | 21 | 22 |
| Špatně | Správně | $WER_{T5} = 0.0 \wedge WER_{W2V2} > WER_{T5}$ | 638 | 658 | 178 | 185 |
| | Špatně | $WER_{W2V2,T5} > 0.0 \wedge WER_{W2V2} > WER_{T5}$ | 757 | 740 | 414 | 411 |
| | | $WER_{W2V2,T5} > 0.0 \wedge WER_{W2V2} = WER_{T5}$ | 2100 | 2115 | 192 | 206 |
| | | $WER_{W2V2,T5} > 0.0 \wedge WER_{W2V2} < WER_{T5}$ | 495 | 477 | 87 | 69 |

Tabulka poskytuje vhled na četnosti, kdy je důležité věnovat pozornost 2. a 6. řádku, kde došlo ke zvýšení WER na predikci po přidání modelu T5. Pro tento přístup je žádoucí, aby hodnoty na těchto řádcích byly minimální. Je tedy požadováno, aby model T5 chyby opravoval nebo nechával predikci od modelu Wav2Vec2.0 nezměněnou (5. řádek).

Je zřejmé, že je naopak žádoucí, aby četnosti na řádcích 1 a 3 obsahovaly většinu nebo v ideálním případě veškeré četnosti predikcí, které po průchodu ST6 neobsahují žádnou chybu. Za žádoucí lze také považovat vyšší počty na řádku 4, kde dochází k částečné opravě nebo přiblížení se k požadovanému výstupu. Což může mít značný vliv na zvýšení sémantické podobnosti, které je dále probíráno v kapitole 7.3.

## 7.2 Typické chyby

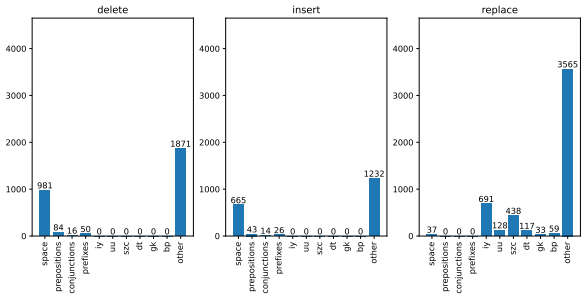
ASR modely, jako v této práci využívaný model Wav2Vec2.0, mají značně vyšší nejistotu zavedenou samotným vstupem nežli NLP modely, které pracují s prostým textem. Za limitace vycházející z podstaty vstupu (audio signál) lze považovat přidaný šum z okolí a omezení záznamového zařízení. Vlastní digitalizace zvuku může být ovlivněna frekvencí vzorkování, kvalitou součástek zařízení a přesností zvolenou pro ukládání signálu. Mezi další limitace patří změna řečníka, ale i změny v hlase stejného řečníka zapříčiněné nemocí

[11] https://github.com/sulcm/wav2vec2-w-T5-cs/blob/main/error_eval.ipynb

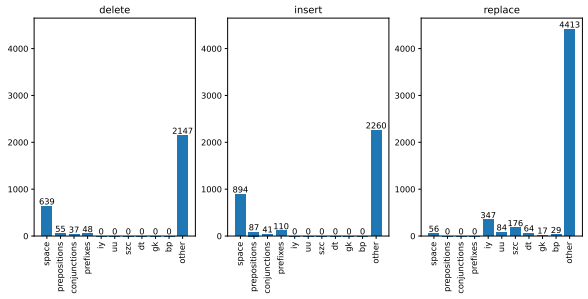
nebo změnami v emocích. Dalšími problémy se záznamem řeči a následným korektním rozpoznání bývají šumivé hlásky (sykavky, například s/z/c/š/ž/č) nebo párové hlásky znělé a neznělé (například b/p, d/t, ď/ť, g/k), jenž často bývají zaměňovány. Po hlubší analýze zaváděných chyb byly objeveny další typické problémy, které nastávaly odstraňováním nebo přidáváním významných elementů v predikci. Zejména tyto problémy nastávaly u mezer, předložek, spojek a předpon. Dále vznikaly chyby, pravděpodobně zapříčiněné nedostatečnou znalostí psaného přirozeného jazyka, kdy docházelo k záměnám i/y/í/ý nebo u/ú/ů.

Tyto nedostatky byly jednou z hlavních motivací doplnění Wav2Vec2.0 o LM v podobě T5, který by mohl opravit popisované chyby nebo v případě části nesrozumitelného přepisu odhadnout správnou nebo dostatečně blízkou formu náhrady.

Následující sloupcové grafy vyobrazují četnosti popisovaných chyb na sledovaných operacích (odstranění - *delete*, přidání - *insert* a záměny - *replace*, viz. použité metriky 5.2) pro testovací porce dat z datasetů Common Voice (obrázek 10) a VoxPopuli (obrázek 11). Pro konzistentnost byla vždy využita verze v23 modelu Wav2Vec2.0 a verze v20 modelu T5 s konfigurací generování `num_beams` 4 a povoleným `do_sample`. Samotná analýza byla prováděna za pomoci vytvořené třídy `ST6ErrorAnalysis`, jenž sloužila pro enkapsulaci načtených definicí klasifikace z JSON souboru (viz. tabulka 9), heuristicky vytvořených jednoduchých metod pro zachycení co nejvíce případů výskytu zkoumaných chyb a metody pro extrakci chyb a nejbližšího kontextu (mezi nejbližšími mezerami v sekvenci) v jakém nastala pomocí Levenshteinových operací.

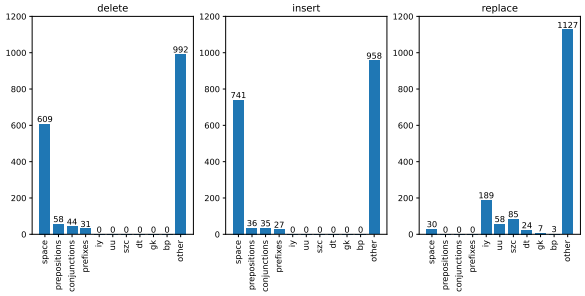



(a) Chyby v operacích zavedené modelem Wav2Vec2.0 na požadovaný výstup.

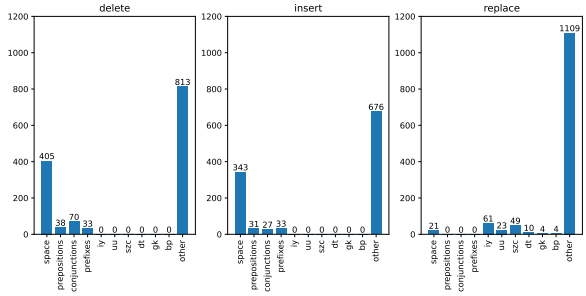



(b) Chyby v operacích na výstupu po přidání T5 modelu, chybovost ST6, v porovnání s požadovaným výstupem (důsledek přidání chyb nebo ponechání již zavedených chyb modelem Wav2Vec2.0).

Obrázek 10: Operace prováděné nad testovací sadou Common Voice datasetu (legenda v tabulce 9).




(a) Chyby v operacích zavedené modelem Wav2Vec2.0 na požadovaný výstup.




(b) Chyby v operacích na výstupu po přidání T5 modelu, chybovost ST6, v porovnání s požadovaným výstupem (důsledek přidání chyb nebo ponechání již zavedených chyb modelem Wav2Vec2.0).

Obrázek 11: Operace prováděné nad testovací sadou VoxPopuli datasetu (legenda v tabulce 9).

Tabulka 9: Legenda k obrázkům 10 a 11 uvádějící stručný popis a definici tříd z heuristických úvah založených nad obdrženými daty, na základě kterých se prováděla klasifikace. Využívaný soubor obsahující celé definice tříd lze nalézt zde[12].

| Třída | Popis a definice třídy |
|---|---|
| *space* | došlo k přidání nebo odebrání mezery |
| *prepositions* | spojitost s mezerami, kontrola typických předložek (z, k, v, ...) |
| *conjunctions* | spojitost s mezerami, kontrola typických spojek (a, i, či, ...) |
| *prefixes* | spojitost s mezerami, kontrola typických předpon (s, z, vz, ...) |
| *iy* | znalost jazyka (i, y, í, ý) |
| *uu* | znalost jazyka (u, ú, ů) |
| *szc* | sykavky (s, z, c, š, ž, č) |
| *dt* | spodoba znělosti (d, t, ď, ť) |
| *gk* | spodoba znělosti (g, k) |
| *bp* | spodoba znělosti (b, p) |
| *other* | ostatní neklasifikované chyby |

Z grafů odpovídající operacím *delete* a *insert* na obrázku 10 si lze všimnout vzniklé vazby mezi modely, kde pokud model Wav2Vec2.0 často například odstraňoval mezery, tak na to model T5 reagoval opačným způsobem. Tedy došlo k poklesu odstraňování, ale zároveň s tím došlo ke zvýšení jejich přidávání. Podobnou, ačkoliv ne tak významnou, vazbu lze pozorovat i u ostatních klasifikací odpovídající těmto operacím.

Přesuneme-li pozornost ke grafům odpovídající operaci *replace* na obrázcích 10 a 11, kde vzájemně porovnáme ty odpovídají modelu Wav2Vec2.0 s modelem T5, všimneme si značného snížení potřebných záměn u modelu T5. Což naznačuje, že provedená úvaha doplnění Wav2Vec2.0 o LM (model T5) splňuje svůj účel a korektně opravuje některé základní jazykové chyby popisované na počátku této kapitoly.

## 7.3 Sémantická podobnost

Jak již bylo v práci několikrát zmíněno, jedením ze způsobů jakým lze nahlížet na chyby je striktní, řízený například využitými metrikami WER a CER, které porovnávají predikci a referenční výstup čistě na základě chybných slov nebo znaků. Což by mohlo

[12]https://github.com/sulcm/wav2vec2-w-T5-cs/blob/main/word_classes_definitions.json

být postačující, jednalo-li by se pouze o model Wav2Vec2.0, u něhož se požaduje toto chování s minimálními odchylkami od požadovaného výstupu. Jelikož je v této práci model Wav2Vec2.0 doplněný o model T5, který jakožto jeden ze zástupců LM, má možnost do jisté míry "vymýšlet"slova hodící se do kontextu, pokud je jeho vstup z části nesrozumitelný. Takto může dojít případně i ke zhoršení měřených metrik, ale svým významovým obsahem naopak může dojít k přiblížení se k referenčnímu výstupu, které může vést ke zvýšení porozumění v navazujících systémech ovládaných hlasem nebo i u předávání informací dalšímu člověku (příklady viz. tabulka 11).

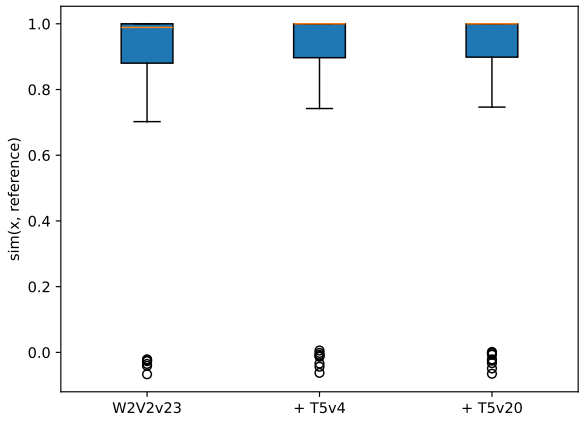
Pro porovnání sémantické blízkosti dvou textových sekvencí je zapotřebí dostat se do takového prostoru, kde je možné zvolenou metrikou určit jejich vzájemné vzdálenosti. V těchto případech se využívá možnost vytvoření vektorů embeddingu z těchto sekvencí, které je zobrazí v $n$-dimenzionálním prostoru, kde lze již poměrně jednoduše určovat vzdálenosti právě mezi dvěma vektory. Za tímto účelem v této práci byla využita schopnost modelu BERT [9], který po předtrénování poskytuje tyto vektory embeddingu obsažené v jednom z jeho speciálních tokenů [CLS], jenž se vždy nachází na počátku enkódované sekvence. Pod tímto tokenem se nachází vektor embeddingu o dimenzi 768, který obsahuje celou zakódovanou reprezentaci vstupní sekvence v prostoru vytvořeného při předtrénování modelu. Předtrénovaný model BERT byl za těmito účely poskytnut panem Ing. Janem Lehečkou, Ph.D., který se zabýval jeho předtrénováním na české verzi C4 datasetu (označovaný jako C5, Czech Colossal Clean Crawled Corpus) v práci [27].

V datové analýze pro zjištění míry podobnosti se často využívá kosinové podobnosti (22). Toto vyjádření podobnosti reprezentuje kosin úhlů, které svírají dva zkoumané vektory. Výsledek tedy vždy spadá do uzavřeného intervalu $<-1; 1>$, jehož geometrická interpretace pro 1 značí proporcionalitu dvou vektorů (vektory, které mají stejný směr, ale různou velikost), pro 0 jsou vektory vzájemně ortogonální a pro $-1$ směřují opačným směrem. V kontextu analýzy významové blízkosti dvou vektorů embeddingu textových sekvencí je žádoucí, aby se tato míra vždy blížila 1. Následující vztah vyjadřuje kosinovou podobnost:

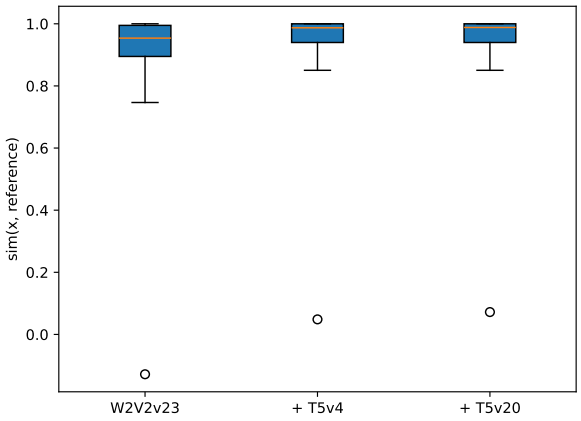
$$\mathrm{sim}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\| \, \|\mathbf{y}\|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \, \sqrt{\sum_{i=1}^{n} y_i^2}} \tag{22}$$

kde $\mathbf{x}, \mathbf{y}$ jsou porovnávané vektory a $n$ je velikost jejich dimenze. V této práci se za vektory $\mathbf{x}$ a $\mathbf{y}$ dosazují vektory embeddingu, vytvořené modelem BERT, pro predikovaný výstup modely Wav2Vec2.0 nebo T5 a jeho referenci, jejichž dimenze je $n = 768$.

Následující obrázek obsahuje vykreslené grafy *boxplot*ů zachycující základní statistické údaje ze získaných měr podobností. Tedy poskytuje informace o mediánu, dolním (0.25 percentil, $\mathrm{Q}_{0.25}$) a horním (0.75 percentil, $\mathrm{Q}_{0.75}$) kvartilu, a dolní 0.01 percentil z odlehlých hodnot vyjadřující dosažené extrémy v "nepodobnosti".




(a) Statistické porovnání dosažené kosinové podobnosti na testovací sadě Common Voice.




(b) Statistické porovnání dosažené kosinové podobnosti na testovací sadě VoxPopuli.

Obrázek 12: Statistické vlastnosti zjištěné na testovacích sadách datasetů při použití kosinové míry pro určení sémantické podobnosti.

Dosažené přiblížení v podobnosti bylo otestováno, zda ho lze považovat za statisticky významné pomocí t-testu. Porovnávání dvou nezávislých vzorků o stejné velikosti je prováděno využitím varianty dvouvýběrového t-testu (23), jehož implementace byla poskytnuta funkcí `scipy.stats.ttest_ind`, která navrací objekt `TtestResult` obsahující vypočtenou statistiku $t$ se stupni volnosti $df$ a p-hodnotu

$$\begin{aligned} t &= \frac{\mathrm{E}[\mathbf{y}] - \mathrm{E}[\mathbf{x}]}{\sqrt{\frac{\mathrm{Var}[\mathbf{y}]+\mathrm{Var}[\mathbf{x}]}{n}}} \\ df &= 2 \cdot n - 2 \end{aligned} \tag{23}$$

kde opět $\mathbf{x}, \mathbf{y}$ jsou porovnávané vektory a $n$ je velikost jejich dimenze (počet vzorků). Pro určení zda dosažené zlepšení v sémantické podobnosti je statisticky významné se typicky uvažuje výsledná p-hodnota (signifikance), kdy nabývá hodnoty $< 0.05$ (5e-2) hladiny významnosti.

Významné informace z *boxplot*ů (obrázek 12) a odpovídající p-hodnoty pro různé kombinace modelů na testovacích sadách datasetů jsou zaznamenány v následující tabulce.

Tabulka 10: Statistické hodnoty určené z vektorů obsahují vypočtené kosinové podobnosti. p-hodnota byla vždy určena porovnáním samotného Wav2Vec2.0 s doplněnými T5 modely.

| | Common Voice (*test*) | | | | | VoxPopuli (*test*) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Model | min | med | $Q_{0.25}$ | $Q_{0.75}$ | p-hodnota | min | med | $Q_{0.25}$ | $Q_{0.75}$ | p-hodnota |
| W2V2v23 | -0.07 | 0.99 | 0.88 | 1.00 | - | -0.13 | 0.95 | 0.89 | 0.99 | - |
| + T5v4 | -0.06 | 1.00 | 0.90 | 1.00 | 1.24e-2 | 0.05 | 0.99 | 0.94 | 1.00 | 4.44e-9 |
| + T5v20 | -0.06 | 1.00 | 0.90 | 1.00 | 1.28e-2 | 0.07 | 0.99 | 0.94 | 1.00 | 5.17e-10 |

Při porovnání zaznamenaných p-hodnot na hladině významnosti $\alpha = 0.05$ lze výsledky testu rozdílu sémantické blízkosti dosahované samostatným Wav2Vec2.0 a následně doplněnými modely T5 označit za statisticky významné a lze potvrdit, že dochází k zvýšení sémantické podobnosti k referenční sekvenci při využití navrhovaného přístupu ST6.

Následující tabulka 11 obsahuje několik vybraných příkladů, kdy došlo ke zvýšení sémantické podobnosti, ačkoliv dle metriky WER došlo po přidání T5 modelu k jejímu zhoršení. Pro uvedení do širšího kontextu byly vzaty všechny příklady, které jsou obsažené

v tabulce 8 na posledním řádku (sloupec s Common Voice datasetem a přidanou v20 verzí modelu T5). Kde pro 477 příkladů došlo přidáním modelu T5 ke zhoršení v měřené metrice WER, ale ve 131 případech z uváděných 477 příkladů došlo naopak ke zvýšení podobnosti dle kosinové metriky (22).

Tabulka 11: Vybrané příklady z testovací sady datasetu Common Voice, u kterých došlo ke zhoršení v metrice WER po přidání modelu T5, ale zároveň došlo i ke zvýšení sémantické podobnosti dle kosinové metriky. Uvedené příklady byly získány použitím verzí Wav2Vec2.0v23 a T5v20 modelů. Chyby a pokusy o jejich nápravu jsou tučně zvýrazněny.

| Index<br>WER<br>WER | Porovnání výstupů ST6 v jednotlivých fází s referencí (požadovaným výstupem) | |
|---|---|---|
| 1681 | Reference | po dešti se kamenná **dlažba** na ulici leskla **jako by** to bylo velké zrcadlo |
| 7.14 | W2V2 přepis | po dešti se kamenná **vlažba** na ulici leskla **jako by** to bylo velké zrcadlo |
| 14.29 | T5 korekce | po dešti se kamenná **dlažba** na ulici leskla **jakoby** to bylo velké zrcadlo |
| 2191 | Reference | **a jsou i** celkem pohodlné |
| 40.00 | W2V2 přepis | **až ceu i** celkem pohodlné |
| 60.00 | T5 korekce | **je to** celkem pohodlné |
| 2831 | Reference | do **emailů** jim píšu většinou nějakou poznámku trochu vysvětlující tyto věty ze zadání |
| 7.69 | W2V2 přepis | do **ímelu** jim píšu většinou nějakou poznámku trochu vysvětlující tyto věty ze zadání |
| 15.38 | T5 korekce | do **e mailu** jim píšu většinou nějakou poznámku trochu vysvětlující tyto věty ze zadání |
| 3233 | Reference | **jenom** nemám přes den moc času |
| 16.67 | W2V2 přepis | **iram** nemám přes den moc času |
| 50.00 | T5 korekce | **já na to** nemám přes den moc času |
| 5613 | Reference | **teď** si **nějak** nemohu vzpomenout |
| 40.00 | W2V2 přepis | **te** si **nijak** nemohu vzpomenout |
| 60.00 | T5 korekce | **na to** si **nijak** nemohu vzpomenout |

Zkoumané a uváděné příklady byly vybrány právě z takové skupiny, kde je možné nejvhodněji dokázat zmiňovaná tvrzení, týkající se popisovaných pohledů na chyby a možnosti zvýšení sémantické podobnosti, ačkoliv nedochází k plné opravě chyb modelem T5 na predikovaném textu z Wav2Vec2.0 modelu.

# 8 Závěr

Hlavním cílem této práce byl průzkum modelu Wav2Vec2.0 a návrh jeho rozšíření o navazující jazykový model T5. Výsledkem práce byly získány natrénované modely Wav2Vec2.0 a T5, které byly vyhodnoceny na základě zvolených metrik. Následně došlo k jejich porovnání vzhledem k výsledků dosahovaných samostatným modelem Wav2Vec2.0 a navrženým rozšířením o model T5, označovaným jako ST6.

První část práce se zabývala teoretickou a implementační podstatou Wav2Vec2.0 a T5 modelů s jejich společným základem založeným na architektuře transformerů. Zde byly popsány všechny významné části, ze kterých se modely skládají s jejich vztahy a důsledky. Dále došlo k popisu některých implementačních detailů týkající se použitých variant modelů. V této části byly uváděny i postupy týkající se předtrénování a dotrénování modelů s využitými datasety.

Druhá část práce obsahuje postupy experimentů s dotrénováním modelů Wav2Vec2.0 a T5 s dosahovanými výsledky v jednotlivých experimentech. Došlo k zavedení metodiky postupu, definici metrik využívaných pro vyhodnocení a porovnání. Dále bylo popsáno prostředí, kde docházelo k testování a spouštění všech experimentů. Již během postupného testování bylo evidentní, že navrhované rozšíření ASR modelu o LM přinese zlepšení v měřených metrikách. Testování navíc také poskytlo hlubší vhled na chyby zaváděné ASR a LM modely, které byly dále prozkoumávány vhodně navrženou analýzou.

V poslední části práce došlo ke spojení těchto dvou modelů a dalo se tak vzniku navrhovanému přístupu ST6. Zde dochází k vyhodnocení a porovnání samostatného Wav2Vec2.0 modelu s různými doplněnými verzemi T5 modelu. Všechny tyto experimenty byly prováděny na testovacích sadách využitých datasetů a také ve všech zaznamenaných případech došlo k významnému zlepšení v měřených metrikách s využitím navrhovaného přístupu ST6. Analýza chyb poskytla vhled a také opodstatnila dosahované výsledky, kdy byly zkoumány vzájemné vlivy mezi využívanými modely. Ukázalo se, že přidání informací o přirozeném psaném jazyce umožňuje opravy některých základních jazykových chyb zaváděných ASR modely. Dále byl zkoumán vliv na významovou podobnost po přidání T5 modelu, kdy docházelo ke statisticky významnému zvýšení v sémantické podobnosti při využití navrhovaného řešení. Tímto došlo k úspěšnému zpracování všech bodů uvedených v zadání této práce a bylo ukázáno, že navrhovaný přístup ST6 vede k zvýšení přesnosti transkripce záznamů řeči.

Během průzkumů navrhovaného řešení byl zvažován rozdílný přístup, kde dochází k přidání LM dekodéru přímo do modelu Wav2Vec2.0. Takto přidávaný dekodér byl typicky jednoduchý $n$-gramový jazykový model využívající statistické informace o četnostech sledů $n$ slov v trénovacím datasetu. Navrhované řešení s využitím modelu T5 pravděpodobně bude dosahovat lepších výsledků, jelikož poskytuje vetší porozumění kontextu a generalizaci. Avšak pokud bude brán v potaz nutný výpočetní výkon a omezenost paměťového prostoru, tak přístup s využitím $n$-gramů poskytuje vhodnou alternativu. Zmiňovaný přístup s $n$-gramovým dekodérem nebyl v této práci nijak rozvíjen z důvodů časové omezenosti, byl zde pouze uveden pro úplnost jako možná alternativa a prostor pro navázání v pracích zabývajících se obdobnou problematikou.

# Reference


[1] Alexei Baevski et al. „wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations“. In: *CoRR* abs/2006.11477 (2020). arXiv: 2006.11477. URL: https://arxiv.org/abs/2006.11477.

[2] Jan Lehečka et al. „Exploring Capabilities of Monolingual Audio Transformers using Large Datasets in Automatic Speech Recognition of Czech“. In: *Interspeech 2022*. interspeech-2022. ISCA, zář. 2022. DOI: 10.21437/interspeech.2022-10439. URL: http://dx.doi.org/10.21437/Interspeech.2022-10439.

[3] Colin Raffel et al. „Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer“. In: *CoRR* abs/1910.10683 (2019). arXiv: 1910.10683. URL: http://arxiv.org/abs/1910.10683.

[4] Ashish Vaswani. „Attention is all you need“. In: *arXiv preprint arXiv:1706.03762* (2017). URL: https://cir.nii.ac.jp/crid/1370580229800306054.

[5] Dzmitry Bahdanau, Kyunghyun Cho a Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. arXiv: 1409.0473 [cs.CL].

[6] Minh-Thang Luong, Hieu Pham a Christopher D. Manning. *Effective Approaches to Attention-based Neural Machine Translation*. 2015. arXiv: 1508.04025 [cs.CL].

[7] Jay Alammar. *The Illustrated Transformer*. Čvn. 2018. URL: https://jalammar.github.io/illustrated-transformer/.

[8] Jimmy Lei Ba, Jamie Ryan Kiros a Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: 1607.06450 [stat.ML].

[9] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[10] Dan Hendrycks a Kevin Gimpel. *Gaussian Error Linear Units (GELUs)*. 2023. arXiv: 1606.08415 [cs.LG].

[11] Abdelrahman Mohamed, Dmytro Okhonko a Luke Zettlemoyer. *Transformers with convolutional context for ASR*. 2020. arXiv: 1904.11660 [cs.CL].

[12] Herve Jégou, Matthijs Douze a Cordelia Schmid. „Product Quantization for Nearest Neighbor Search“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.1 (2011), s. 117–128. DOI: 10.1109/TPAMI.2010.57.

[13] Eric Jang, Shixiang Gu a Ben Poole. *Categorical Reparameterization with Gumbel-Softmax*. 2017. arXiv: 1611.01144 [stat.ML].

[14] Adam Paszke et al. „PyTorch: An Imperative Style, High-Performance Deep Learning Library“. In: *Advances in Neural Information Processing Systems*. Ed. H. Wallach et al. Sv. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

[15] Thomas Wolf et al. „Transformers: State-of-the-Art Natural Language Processing“. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, říj. 2020, s. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[16] R. Ardila et al. „Common Voice: A Massively-Multilingual Speech Corpus“. In: *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*. 2020, s. 4211–4215.

[17] Quentin Lhoest et al. „Datasets: A Community Library for Natural Language Processing“. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online a Punta Cana, Dominican Republic: Association for Computational Linguistics, lis. 2021, s. 175–184. arXiv: 2109.02846 [cs.CL]. URL: https://aclanthology.org/2021.emnlp-demo.21.

[18] Alex Graves et al. „Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks“. In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006, s. 369–376. ISBN: 1595933832. DOI: 10.1145/1143844.1143891. URL: https://doi.org/10.1145/1143844.1143891.

[19] Jaesong Lee a Shinji Watanabe. „Intermediate loss regularization for ctc-based speech recognition“. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, s. 6224–6228.

[20] Alex Wang et al. *GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding*. 2019. arXiv: 1804.07461 [cs.CL].

[21] Kishore Papineni et al. „Bleu: a method for automatic evaluation of machine translation“. In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 2002, s. 311–318.

[22] Chin-Yew Lin a FJ Och. „Looking for a few good metrics: ROUGE and its evaluation“. In: *Ntcir workshop*. 2004.

[23] Kaitao Song et al. *MASS: Masked Sequence to Sequence Pre-training for Language Generation*. 2019. arXiv: 1905.02450 [cs.CL].

[24] Jan Švec et al. „General framework for mining, processing and storing large amounts of electronic texts for language modeling purposes“. In: *Language resources and evaluation* 48.2 (2014), s. 227–248.

[25] Taku Kudo a John Richardson. *SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing*. 2018. arXiv: 1808.06226 [cs.CL].

[26] Changhan Wang et al. „VoxPopuli: A Large-Scale Multilingual Speech Corpus for Representation Learning, Semi-Supervised Learning and Interpretation“. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, srp. 2021, s. 993–1003. URL: https://aclanthology.org/2021.acl-long.80.

[27] Jan Lehečka a Jan Švec. „Comparison of Czech Transformers on Text Classification Tasks“. In: *Statistical Language and Speech Processing*. Ed. Luis Espinosa-Anke, Carlos Martín-Vide a Irena Spasić. Cham: Springer International Publishing, 2021, s. 27–37. ISBN: 978-3-030-89579-2. DOI: 10.1007/978-3-030-89579-2_3.