Vektorová reprezentace projektů VaVaI

Pavel Mikuláš
Matematicko-fyzikální fakulta
Univerzita Karlova
Ročníkový projekt 2019/2020

ABSTRAKT

Cílem této práce je seznámení se s daty z registru informací o výsledcích výzkumu, vývoje a inovací (dále jen RIV VaVaI), porovnání různých postupů vektorizace projektů VaVaI a následné provedení úvodních experimentů pro navazující práci zaměřenou na shlukování a měření podobnosti mezi jednotlivými projekty.

1 ÚVOD DO PROBLEMATIKY

V současné době není jednoduchý způsob, jak měřit míru podobnosti mezi různými projekty. V této práci se pokusím použít postupy strojového učení k převedení textové reprezentace projektů do vektorové reprezentace. To poté umožňuje použití numerických metod pro měření vzdálenosti mezi jednotlivými vektory a tedy i podobnosti mezi projekty. Výsledné vektory můžeme poté na základě podobnosti rozdělit do "shluků" (angl. clusters).

2 ZPŮSOBY VEKTOROVÉ REPREZENTACE

V této sekci stručně nastíním několik současně používaných metod vektorizace textu a "word embeddingových" modelů.

TF-IDF: "Term Frequency-Inverse Document Frequency" [1] je jednoduchá metoda vektorizace textu, která se snaží věty (či celé dokumenty) rozlišovat na základě relativní frekvence výskytu jednotlivých slov v dokumentu. Méně častá slova mají pak vyšší váhu, neboť podle nich můžeme dokumenty snáze rozlišit. TF-IDF reprezentace je ale velice závislá na slovníku, pro který byla vypočítána.

Word2Vec: Word2Vec [2] je prediktivní model, který vytváří vektorovou reprezentaci slov na základě kontextu, ve kterém byla použita. Nedokáže do vektorové reprezentace převádět slova, která nejsou ve slovníku modelu. Ale vektorové reprezentace slov, tzv. "word embeddingy" mohou být předtrénovány na velké množině textů (korpusu) a následně použity pro specifickou úlohu.

Doc2Vec: Doc2Vec [3] je rozšířením modelu word2vec, které zvládá reprezentovat celé dokumenty (věty nebo texty) namísto jednotlivých slov. Jako word2vec predikuje slova na základě kontextu, ve kterém se v textu vyskytují, ale také na základě dokumentu, ve kterém se slovo nachází.

Pozn.: AutoEncoder architektura je architektura neuronové sítě, která se při trénování snaží vstupní vektory promítnout do vektorového prostoru nižší dimenze, tzv. "latent space" a z těcho vektorů poté opět sestrojit vzorový vektor. Takto natrénovanou síť lze poté použít k redukci dimenze vstupních vektorů.

GAN: Generative Adversial Network [4] je model, který funguje na obráceném principu než AutoEncoder. Skládá se z generátoru a z diskriminátoru. Generátor se z vektorů v latent space snaží generovat reálná data. Diskriminátor pak dostane na vstupu jak reálná data, tak data generovaná generátorem a rozhodne, která data pochází z reálné distribuce a která jsou uměle generována.

Spojení AutoEncoder architektury s TF-IDF dosahuje pro shlukování dokumentů velmi dobrých výsledků [5]. V článku nazvaném ClusterGAN autoři předvádějí, že ještě lepších výsledků lze dosáhnout rozšířením GAN architektury [6]. Poslední objevy v oblasti zpracování textu ale ukazují, že pro podobné problémy existují ještě lepší modely založené na "Transformer" architektuře.

BERT: Bidirectional Encoder Representations from Transformers [7] je model založený na nejnovějších metodách zpracování textu. Využívá Transformer architekturu [8] [Obr. 1] a je trénován tak, aby porozuměl kontextu jazyka v obou směrech.

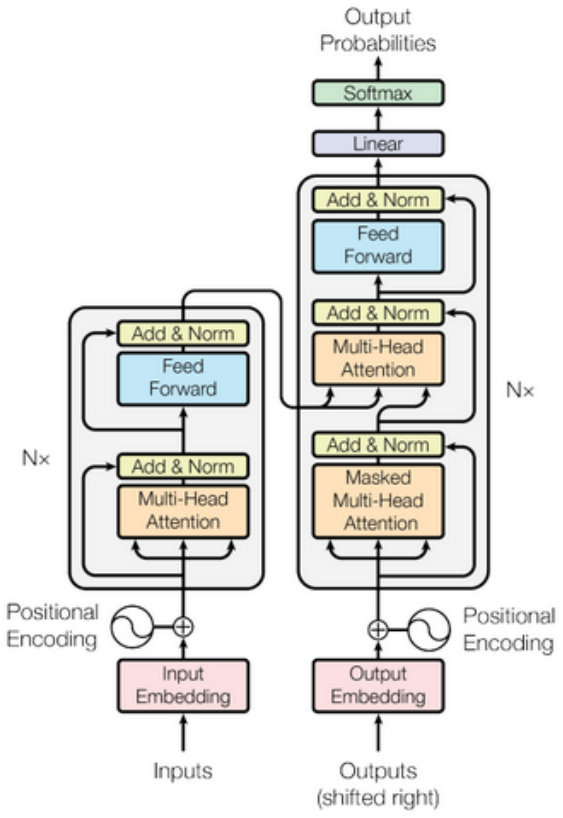

Figure 1: Transformer architektura - encoder vlevo, decoder vpravo (Zdroj: https://arxiv.org/abs/1706.03762)

2.1 TF-IDF

TF-IDF je způsob vektorizace dokumentů, jedná se o součin *term frequecny* $\text{tf}_{\text{i,j}}$ a *inverse document frequency* idf_{i}.

$$\text{tf}_{\text{i,j}} = \frac{n_{i,j}}{\sum_k n_{k,j}} \qquad \text{idf}_{\text{i}} = \log \frac{|D|}{|\{j : t_i \in d_j\}|}$$

Kde $\mathbf{n}_{i,j}$ je počet výskytů slova $\mathbf{t}_i$ v dokumentu $\mathbf{d}_j$. Pro získání relativní frekvence slova (tf) v dokumentu $\mathbf{d}_j$, jmenovatel vydělíme součtem výskytů všech slov v $\mathbf{d}_j$, tedy počtem slov v $\mathbf{d}_j$.

Pro získání idf vydělíme počet dokumentů $|\mathbf{D}|$ počtem dokumentů, které obsahují slovo $\mathbf{t}_i$. To celé následně zlogaritmujeme.

Výstupem je poté matice $\mathbf{E}$, kde na pozici $\mathbf{E}_{i,j}$ je reprezentace slova i v dokumentu j.

2.2 Word2Vec

Pro **Word2Vec** model existují 2 varianty:

- **CBOW** (Continuous Bag Of Words) se snaží predikovat slovo na základě okolních slov v kontextu nějakého rozsahu (viz Obr. 2). Model sečte vektorové reprezentace okolních slov, vynásobí maticí vah a nakonec použije softmax pro predikci jednoho slova.
- **Skip-gram** z jednoho daného slova predikuje kontext. Tedy nejprve slovo zakóduje a poté po vynásobení vahami aplikuje softmax na každou větev a predikuje výsledná slova.

Podle autora Tomáše Mikolova [2], je CBOW architektura rychlejší, ale Skip-gram architektura funguje lépe na menších datech.

Oproti TF-IDF mívají vektorové reprezentace modelu word2vec mnohem menší dimenzi.

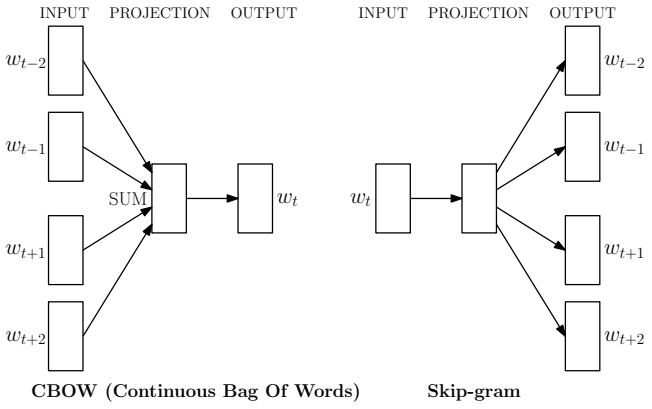

Figure 2: Word2Vec: CBOW vs Skip-Gram (Zdroj: github.com/ufal/npfl114/blob/master/slides/09/word2vec.pdf)

2.3 Doc2vec

Doc2vec rozšiřuje word2vec model přidáním nového vektoru, který zakóduje a zachová informaci o dokumentu, ve kterém se slovo vyskytlo. V původním článku [3] autoři uvádějí jak rozšíření CBOW modelu (viz Obr. 3), tak i skip-gram modelu, kde se pomocí "paragraph id" a nové matice vah predikuje kontext.

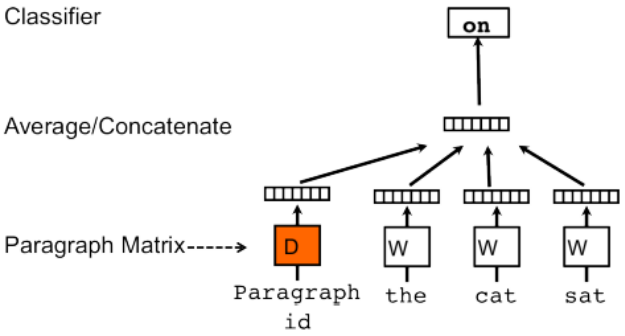

Figure 3: Doc2Vec: Rozšíření CBOW architektury o paragraph id (Zdroj: https://arxiv.org/pdf/1405.4053.pdf)

Autoři rovněž uvádějí, že model vycházející z CBOW dosahuje znatelně lepších výsledků. Doporučují ale používat obě architektury zároveň.

2.4 GAN

Generative Adversial Network je architektura neuronové sítě využívaná hlavně pro zpracování obrazu. Pokud ovšem text převedeme do vektorové reprezentace, lze GAN využít i pro zpracování textu. A to zejména ke zmenšení dimenze vektorové reprezentace pomocí natrénovaného generátoru a následného shlukování v latent space [6].

Základní premisa trénování spočívá v min-maxové hře mezi diskriminátorem a generátorem. Generátor se snaží minimalizovat pravděpodobnost toho, že diskriminátor odhalí jím vygenerovaná data. Diskriminátor se potom snaží tuto pravděpodobnost maximalizovat.

Tedy pokud označíme $\mathbf{G}(\mathbf{z})$ generátor pro distribuci $\mathbf{z} \sim P(\mathbf{z})$ generovanou generátorem a $\mathbf{D}(\mathbf{x})$ jako diskriminátor pro data $\mathbf{x}$. Diskriminátor rozhodne, zda $\mathbf{x}$ pochází z reálné distribuce $\mathbf{P}_{data}$ nebo byla vygenerována generátorem (viz Obr. 4). Hru poté lze matematicky zapsat takto:

$$\min_G \max_D \mathbb{E}_{x \sim P_{data}} [\log D(x)] + \mathbb{E}_{z \sim P(z)} [\log(1 - D(G(z)))].$$

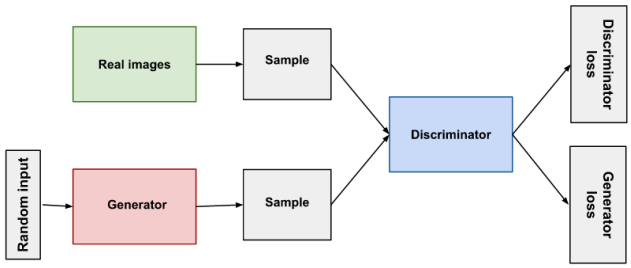

Figure 4: GAN: nástin architektury pro zpracování obrazu (Zdroj: https://developers.google.com/machine-learning/gan/gan_structure)

2.5 BERT

Nejsložitější architekturou, kterou v tomto dokumentu proberu, je **BERT**. V současné době je tato architektura state-of-the-art pro mnoho úloh zpracování textu, včetně klasifikace dokumentů [9]. Jeho hlavní výhodou je, že je přímo trénován k tomu, aby porozuměl kontextu a struktuře textu. Toho je docíleno tím, že model během tréninku například doplňuje chybějící části vět, nebo rozhoduje, zda daná věta navazuje na předchozí větu.

Model vychází z tzv. "transformer" architektury (Obr. 1). Transformer se skládá z encoderu a decoderu. Encoder "zakóduje" užitečné informace ze vstupních dat do vektorové reprezentace. Decoder poté dostane dané vektory a sérií transformací je přetvoří ve výstupní data, např. text v jiném jazyce. Vzhledem k tomu, jak byl model trénován a jak transformer architektura funguje, je tento model schopný zpracovávat celé sekvence namísto jednotlivých slov.

Samotný BERT ve skutečnosti používá pouze několik encoderů postavených za sebe. Natrénovaný model poté tedy převede vstupní data do vektorové reprezentace, ve které zachytí užitečné informace reprezentující vstupní data. Nakonec už je snadné k předtrénovanému modelu připojit další vrstvy a využít tento model pro konkrétní účely. Procesu, kdy se velký natrénovaný model drobně rozšíří a dotrénuje pro specifické použití, se říká "transfer learning".

Nevýhodou BERTa ovšem je, že zvládá zpracovávat pouze sekvence o maximální délce 512 tokenů (slov). Bylo ale ukázáno, že vhodným ořezáním lze text omezit na 512 slov bez ovlivnění výsledků modelu [10].

3 STRUKTURA DAT

V této sekci se budu stručně věnovat struktuře dat, se kterými jsem v tomto dokumentu pracoval. Z databáze projektů https://starfos.tacr.cz jsem získal data o přibližně 80000 projektech. Každý projekt je popsán následujícími atributy:

Kód projektu, Název česky, Název anglicky, Anotace česky, Anotace anglicky, Hlavní CEP obor, Vedlejší CEP obor, Další vedlejší CEP obor, Hlavní FORD obor, Vedlejší FORD obor, Další vedlejší FORD obor, Kategorie VaV, Hlavní účastníci, Další účastníci, Výčet právních forem účastníků, Výčet krajů účastníků, Výčet zemí účastníků, Podrobné informace o účastnících, Hlavní řešitelé, Další řešitelé, Klíčová slova, Výčet druhů dosažených výsledků, Poskytovatel, Program, Uznané náklady, Podpora ze SR, Ostatní veřejné zdroje fin., Neveřejné zdroje fin., Začátek řešení, Konec řešení, URL v Starfose, Relevance.

Pro účely tohoto dokumentu jsem pro vektorizaci zvolil atributy: *Název anglicky, Anotace anglicky a Klčová slova*, tedy veškeré anglické texty, které jsou v datech k dispozici. Rozhodnutí pracovat pouze na anglických textech jsem učinil z toho důvodu, abych mohl využít modely předtrénované na velkém množství dat (například v případě word2vecu).

Texty v těchto sloupcích jsem zřetězil a tedy dostal pro každý projekt reprezentaci ve formě jednoho dlouhého řetězce. Ještě předtím jsem ale z dat vyřadil projekty s chybějícím záznamem v některém ze tří mnou zvolených sloupců. Vyřazeno bylo přibližně 10000 projektů (viz Tabulka 1).

Table 1: Vyřazené projekty

Projektů bez anglického názvu	1005
Projektů bez anglické anotace	8534
Projektů bez klíčových slov	9335

4 ÚVODNÍ POKUSY S VEKTORIZACÍ

Provedl jsem několik úvodních experimentů a získal vektorové reprezentace pro téměř 70000 projektů. Před samotnou vektorizací jsem ale ještě provedl lemmatizaci. K tomu jsem použil WordNetLemmatizer z knihovny nltk [11].

Lemmatizace je proces, který převádí slova do základního tvaru, tzv. lemmatu. Provedení lematizace na textových datech značně zredukuje slovník a tedy i dimenzionalitu následné vektorové reprezentace. Což je právě pro TF-IDF velice výhodné.

4.1 TF-IDF + K-Means

Jako první experiment jsem provedl vektorizaci pouze za pomoci TF-IDF. Vektory generované metodou TF-IDF budou mít vždy stejnou dimenzi, což je velikost slovníku. Pro náš vzorek 70000 projektů dostáváme tedy vektorovou reprezentaci v podobě matice o rozměrech 70000×120000.

Takto velká dimenzionalita nám v podstatě zabraňuje efektivně provést shlukování. Přesto jsem se ale pokusil na vzorku 500 vektorů shlukování provést. Zvolil jsem pro to algoritmus K-means [12].

K-means je shlukovací algoritmus, který má počet shluků pevně daný číslem k. Každý shluk je definován tzv. "centroidem", což je zpočátku náhodně zvolený bod v prostoru. Všechna data jsou pak přiřazena do jednoho z k shluků, a to právě do toho, od jehož centroidu je bod nejblíže. Poté, co jsou všechna data přiřazena příslušným shlukům, algoritmus přepočítá centroidy. Přepočítání provede tak, že za každý centroid zvolí vektor průměrných hodnot ze všech vektorů v daném shluku.

V našem případě ovšem nevíme, jaký je optimální počet shluků. Pro volbu optimálního k, tedy počtu shluků, které "nejlépe" rozdělují data lze využít například tzv. "elbow method". Ta spočívá ve spuštění algoritmu pro všechna k v nějakém rozsahu a následném ohodnocení výsledných shluků. Ohodnocení lze provést velice jednoduše, například sečtením druhých mocnin vzdáleností bodů ve všech shlucích od jejich příslušných centroidů. Pro tuto metriku používáme anglický název "distortion". Jednotlivé výsledky tvoří v tomto případě klesající posloupnost. Místu, kde se rozdíl mezi dvěma po sobě jdoucími body posloupnosti prudce sníží říkáme rameno (angl. "elbow") a právě v tom místě zvolíme příslušnou hodnotu k (viz Obr. 5).

Z Obr. 5 můžeme vidět, že algoritmus zvolil jako optimální $k = 7$. Zároveň ale vidíme, že *distortion score* stále poměrně lineárně klesá. To nám indikuje, že optimální hodnota nejspíše leží mimo prozkoumaný interval. Jelikož je ale tento algoritmus na vektorech o velké dimenzi, jako tomu je v našem případě, časově velmi náročný, rozhodl jsem se v dalším prohledávání pro účely tohoto dokumentu nepokračovat. V navazující práci provedu pokusy i pro vyšší hodnoty k a prozkoumám i jiné možnosti shlukování, například hiearchické.

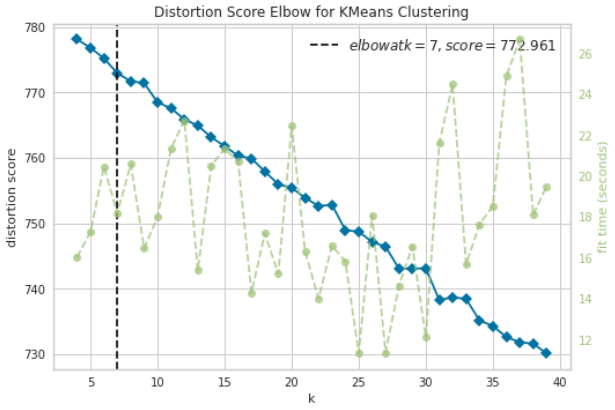

Figure 5: TF-IDF Elbow method na datech VaVaI pro $k \in (4, 40)$

4.2 Word2Vec + K-Means a t-SNE

Při dalším experimentu jsem začal stejně jako u přístupu TF-IDF. Zřetězené anglické texty jsem zlemmatizoval a odstranil stop-words(slova jako spojky, předložky, apod., které nic nevypovídají o významu věty).

Následně jsem využil předtrénovaný word-embeddingový model natrénovaný na 100 miliardách slov z knihovny gensim [13]. Tím jsem získal vektorové reprezentace jednotlivých slov.

Abych získal reprezentaci celých řetězců, spočetl jsem průměr z vektorů jednotlivých slov v daném řetězci, čímž jsem získal jeden výsledný vektor reprezentující příslušný projekt.

Jelikož v tomto případě je dimenze vygenerovaných vektorů podstatně menší, konkrétně 50, mohl jsem poměrně rychle prohledat více nastavení k pro K-means algoritmus a jako optimální hodnotu v tomto případě dostal $k = 40$ (viz Obr. 6).

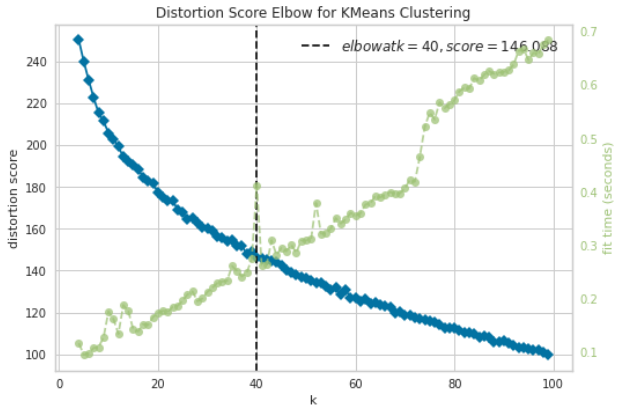

Figure 6: Word2Vec Elbow method pro $k \in (4, 100)$

Nakonec jsem provedl i finální shlukování. Vizualizace 40 shluků by ovšem byla nepřehledná, proto jsem se omezil na 6 shluků. Pro následnou vizualizaci jsem použil algoritmus t-SNE(t-stochastic neighbor embedding) popsaný Geoffreym Hintonem [14]. Jedná se o algoritmus, který je schopný zredukovat dimezi vysoce dimenzionálních dat a reprezentovat je v prostoru o dvou nebo třech dimenzích na základě dané metriky. Taková reprezentace potom umožní vizualizovat vektorové reprezentace jednotlivých projektů ve dvou dimenzích, a tedy i vizualizovat výsledné shluky.

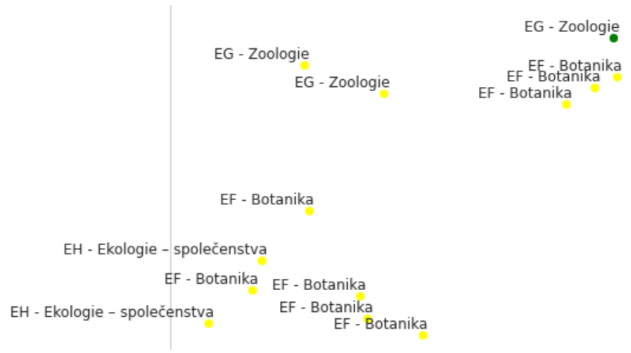

Figure 7: Shluk projektů s oborem biologie

I přesto, že jsem použil pouze omezené množství dat (pouze prvních 500 projektů) a shlukoval do 6ti namísto 40ti shluků, byl jsem schopen z vizualizace jasně rozpoznat některé skupiny projektů jako například biologie, informatika a logistika (viz např. Obr. 7).

Zdrojové kódy a kompletní vizualizace jsou dostupné v mém GitHub repozitáři https://github.com/Fassty/thesis.

5 NAVAZUJÍCÍ PRÁCE

V navazující práci bude třeba se zaměřit hlavně na podrobnou kvantitativní analýzu projektů. Porovnat přístup vektorizace projektů napříč všemi daty oproti zaměření se na jednotlivé obory.

Poté poznatky z úvodních experimentů a analýzy dat využít pro natrénování komplexnějších modelů (jako AutoEncoder, ELMo nebo BERT), které jsem v tomto dokumentu zmínil. Tyto modely poté porovnat s jednodušími způsoby vektorizace a vybrat vhodný způsob shlukování a vizualizace shluků.

REFERENCE

[1] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, no. 1, p. 11–21, 1972.
[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.
[3] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," 2014.
[4] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.
[5] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," vol. 70 of *Proceedings of Machine Learning Research*, (International Convention Centre, Sydney, Australia), pp. 3861–3870, PMLR, 06–11 Aug 2017.
[6] S. Mukherjee, H. Asnani, E. Lin, and S. Kannan, "Clustergan : Latent space clustering in generative adversarial networks," 2018.
[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018.
[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.
[9] A. Adhikari, A. Ram, R. Tang, and J. Lin, "Docbert: Bert for document classification," 2019.
[10] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune bert for text classification?," 2019.

[11] E. Loper and S. Bird, "Nltk: The natural language toolkit," in *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*, 2002.

[12] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[13] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, (Valletta, Malta), pp. 45–50, ELRA, May 2010. http://is.muni.cz/publication/884893/en.

[14] L. van der Maaten and G. Hinton, "Visualizing high-dimensional data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.