Exploring Tokenization Techniques for Protein Language Models

Creston Brooks* **and Indu Panigrahi***
Princeton University, Princeton, NJ, USA
{cabrooks, indup}@princeton.edu

Abstract

The sequence of amino acids that comprise a protein determines its structure and, consequently, its function. A model which learns effective representations for sequences of amino acids can be used for important downstream tasks, such as determining if a protein variant is pathogenic. Protein language models (PLMs), such as ESM2 (Lin et al., 2023), have proven to be effective at learning such representations. An important step in training language models is the choice of vocabulary which defines a model's embedding table and output layer. For models which are trained on natural language, the choice of vocabulary can substantially impact model performance (Bostrom and Durrett, 2020). Most existing work on PLMs defines the model's vocabulary by default to be the set of amino acids (resulting in a character-level model). By using an architecture designed for ancient inscription restoration (Assael et al., 2022), we seek to combine character-level information with information captured by higher-level sequence motifs. We experiment with different choices of tokenizers to capture this higher-level information and compare model performance to the character-level baseline across several evaluation tasks. The code for our trainings and experiments are available at https://github.com/brooksca3/EmBunkBed.

1 Introduction

Given the success of transformer-based models of capturing representations of natural language (Radford et al., 2018; Devlin et al., 2018), protein language models (PLMs) have been explored as a means of learning effective representations of amino acid sequences and applying them to downstream tasks (Cheng et al., 2023; Jumper et al., 2021; Lin et al., 2023). An important step in training language models is the choice of vocabulary which defines a model's embedding table and output layer. For models which are trained on natural language, the choice of vocabulary can substantially impact model performance (Bostrom and Durrett, 2020). Language models trained on natural language typically utilize vocabularies of subwords derived from algorithms such as Byte-Pair Encoding (Sennrich et al., 2015), Unigram (Kudo, 2018), and WordPiece (Schuster and Nakajima, 2012) (Figure 1). However, it is less clear how one might tokenize amino acid sequences apart from the character level.

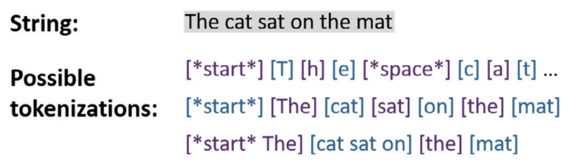
String:
The cat sat on the mat
Possible tokenizations:
[*start*] [T] [h] [e] [*space*] [c] [a] [t] ...
[*start*] [The] [cat] [sat] [on] [the] [mat]
[*start* The] [cat sat on] [the] [mat]

Figure 1: **Examples of tokenizing a sentence.** This example shows several possible tokenizations for a natural language sentence. The first tokenization is a character-level tokenization where each letter forms a token, and the second tokenization is a word-level tokenization where each word forms a token. The third tokenization shows a higher-level tokenization where some words are combined. Though not shown in these examples, suffixes and prefixes can also form tokens. Figure from Figure 1B in Ofer et al. (2021).

While most prior work on PLMs have trained character-level models, with each amino acid forming its own token, some work has applied tokenization methods which are commonly used for natural language to protein sequences. For example, Asgari et al. (2019) uses Byte-Pair Encoding, and Wang et al. (2019) uses SentencePiece to create tokens of groups of contiguous amino acids.

Although some domains, such as protein sequences, rely heavily on bidirectional context use for inference, the use of Masked Language Models (MLMs) for pretraining across most domains is increasingly rare, with the development of large auto-regressive language models exhibiting state-of-the-art performance across baselines. Beyond

PLMs, rarer still is the use of character-level language models: in most contexts, operating at a sub-word level allows for shorter input sequences and quicker generalized learning.

Remarkably, one domain which also relies heavily on bidirectional context for inference and operates on the character level is the restoration of ancient texts and inscriptions (Assael et al., 2019, 2022). Manuscripts and inscriptions contain gaps caused by physical damage – it is the work of epigraphers and philologists to fill these spans of missing characters to attempt to restore the original text (Graziosi et al., 2023). Moreover, research in ancient inscription restoration has found that combining character-level with sub-word level information can be beneficial for character-level MLM pretraining. (Assael et al., 2022).

In this work, we apply the MLM architecture proposed by Assael et al. (2022), which was developed for ancient inscription restoration, to sequences of amino acids. This architecture takes a sub-word tokenizer, auxiliary to the primary tokenizer which is character-level, and combines the embedding of a given character with the embedding of the larger sub-word of which it is a part. Likewise, we train multiple models, trying different choices of auxiliary tokenizers, and we evaluate these models on two tasks, single-token infilling and variant-effect prediction (VEP). We compare the performances of the resulting models with that of the character-only baseline.

We find that combining character-level and word-level embeddings improves the model's understanding of protein sequences compared to the character-only baseline. Furthermore, our experiments show some evidence that naively using fixed-length words as tokens can be competitive with tokenizations that represent complex, variable-length words. Finally, we identify avenues for extending our experimentation to more settings and larger models.

2 Related Work

2.1 Protein Language Models

Proteins are sequences of amino acids of which there are 20 in total. The identity and ordering of amino acids in a sequence determines the structure and function of the represented protein. Furthermore, certain groups of amino acids tend to co-occur with others (Ofer et al., 2021; Shen et al., 2007). Thus, interpreting protein sequences requires an understanding of the amino acids and their positioning, similar to how we read natural language sentences.

Protein sequences can be up to thousands of amino acids long, and there exist billions of proteins whose structures are unknown (Jumper et al., 2021). While it would take an unreasonable amount of time for domain experts to manually convert every sequence to a structure, leveraging deep learning to perform these conversions is one way to automate this process.

One of the most well-known works in applying deep learning to protein sequences is AlphaFold2 (Jumper et al., 2021) which predicts the structure of the protein from its sequence. At its base, AlphaFold2 is an MLM, randomly masking out and predicting input tokens; however, it also makes use of multiple sequence alignments and pairwise features during training (Jumper et al., 2021). Other past work in protein language models includes ESM2 (Lin et al., 2023) and ProteinBERT (Brandes et al., 2022). These works use character-level vocabularies and do not compare the effects of using different tokenizations or auxiliary embedding architectures.

2.2 Tokenizations

Tokenization is the process of segmenting input text into a sequence of tokens, which is a predefined discrete set of vocabulary items (Figure 1). This process occurs at the beginning of most modern NLP pipelines, as it is integral to tasks such as masked language modeling and next token prediction. Without a discrete set of vocabulary items, predicting the next token (in the auto-regressive case) or a missing token (in the MLM case) would be infeasible. Tokens can be characters (individual letters), sub-words (clusters of letters which can join to form whole words), or even whole words.

While in natural language, it is simple enough to delineate characters, sub-words, and words, based on conventions of spacing and punctuation, there are no well-defined separations in protein sequences. Nonetheless, amino acids do have some co-occurrences that form motifs, motivating the potential for super-character delineations. (Ben-Hur and Brutlag, 2006; Ofer et al., 2021).

Some prior work in protein language modeling has tried tokenizing protein sequences by using applying tokenization algorithms commonly used for natural language, such as SentencePiece (Wang

Sequence: TYHMCQFHCR

Character-level: [start] [T] [Y] [H] [M] [C] [Q] [F] [H] [C] [R]
n-gram: [start] [TYH] [MCQ] [FHC] [R##]
WordPiece: [start] [TYHM] [CQ] [FHCR]

Figure 2: **Tokenization Types.** We compare three types of tokenization: character-level, n-gram, and WordPiece with an example shown for each. n-gram and WordPiece are both word-level tokenizations that we combine with character-level embeddings. The n-gram is shown for $n = 3$. Sample sequence is from the UniProt dataset (Consortium, 2022).

et al., 2019) and Byte-Pair Encoding (Asgari et al., 2019). However, prior work in PLMs has not used architectures which operate at multiple token granularities (i.e., character level and super-character level).

2.3 Ithaca Architecture

Proposed by Assael et al. (2022), Ithaca is an MLM trained on a corpus of Ancient Greek inscriptions for the purpose of gap filling, as well as temporal and geographical attribution. Ithaca is a character-level BERT model with one major architectural change – in addition to its character-level vocabulary, Ithaca keeps a separate, auxiliary word-level vocabulary of about $35,000$ tokens. Consequently, the model learns two separate embedding tables, one for character tokens and another for word tokens. When the model takes in text, the embedding of a given character is combined with the embedding of the larger word of which it is a part; for example, for the input "dog", the character embeddings of "d", "o", and "g" are each combined with the word embedding for "dog", using a feed-forward layer. This embeds word-level understanding to the character-level units.

3 Methods

3.1 Tokenizers

We implement a character-level tokenizer where each amino acid is a separate token. This is the base tokenizer for each of the Ithaca-like models we train. Additionally, using the training data (described later in Section 4), we use the HuggingFace's WordPiece trainer[1] to attain WordPiece tokenization with vocabulary sizes of $1,000$ tokens and $10,000$ tokens. Lastly, we also create "n-gram" tokenizers, which split sequences into groups of n consecutive amino acids (Figure 2). We perform experiments for $n = 2$ and $n = 3$, building vocabularies by taking every unique sequence of two and three consecutive amino acids, respectively.[2] Notably, the n-gram tokenizers are not robust to insertions or deletions (indels); any indel would cause the sequence to be tokenized completely differently (Figure 3).

Original Sequence: TYHMCQF
Mutated Sequence: THMCQF

Original n-gram: [start] [TYH] [MCQ] [F##]
Mutated n-gram: [start] [THM] [CQF]

Figure 3: **Example of Deletion affecting an n-gram tokenization.** When the Y is deleted from the original protein sequence, the entire sequence after that position shifts left by one. As a result, a 3-gram tokenization would change the grouping of amino acids at and after the deleted position which results in a completely different tokenization. Sequence sample is from the UniProt dataset (Consortium, 2022).

As we have several variations on several tokenization methods, in total we train five models; four of these are Ithaca-like models, each with a different auxiliary tokenizer (WordPiece 1K, WordPiece 10K, 2-gram, 3-gram), and the final model is the charater-only BERT which serves as our baseline.

3.2 Three-Phase Training

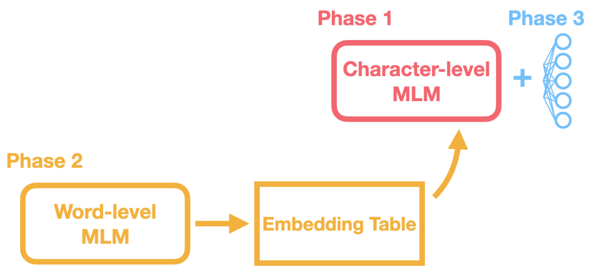

Figure 4: **Three-Phase Training.** We train the models in three phases. During Phase 1, we train a character-level MLM. During Phase 2, we train a word-level MLM; this phase differs depending on whether we are training with an n-gram or with a WordPiece tokenizer. During Phase 3, we append a fully-connected layer that combines the character-level embeddings from Phase 1 and the word-level embeddings from the embedding table in Phase 2.

[1] https://huggingface.co/docs/tokenizers/en/api/trainers

[2] The 2-gram and 3-gram tokenizers have vocabulary sizes of just under $1,000$ and $10,000$ tokens, respectively.

Model	Top 1	Top 2	Top 3	Top 4	Top 5
Char-Only	17.29	28.05	37.09	44.80	51.56
Char+WP1K	17.39	**28.28**	37.21	**44.97**	51.67
Char+WP10K	17.39	28.26	**37.22**	44.96	**51.69**
Char+2-gram	17.36	28.13	37.11	44.83	51.63
Char+3-gram	**17.42**	28.21	37.17	44.85	51.57

Table 1: **Top K Percent Accuracies on Character Infilling Task.** Performance of five models on a character infilling task, where a character is randomly masked in a test example 500,000 times. We evaluate how accurately each model predicts the masked character, with accuracies presented as percentages for predictions ranking from the top 1 to top 5 choices.

Following Assael et al. (2022), we train character-level MLMs with separate word-level embedding tables and a combination layer to combine the character-level and word-level embeddings. However, instead of performing a single training from scratch, we separate the training process into three different phases (Figure 4). Performing separate pretrainings for such an architecture with different embedding tables has proved crucial in prior work (Devaul, 2024).

In the first phase, we train a standard character-level MLM without any combination layer or additional embedding tables. Likewise, in the second phase, we train a standard word-level MLM without any character-level components. Finally, in the third phase, we take the embedding table from the word-level model and use it as a new embedding table in the character-level model. Additionally, we append a fully-connected layer to combine the character-level and word-level embeddings. We then continue training as the model learns to integrate word-level information to perform its character-level MLM objective.

Importantly, each phase allows the model to specialize in learning distinct representations. Phase 1 builds the fundamental character-level dynamics of the model. Phase 2 learns auxiliary embeddings in a setting where they are the primary means of tokenization, allowing these embeddings to be developed independently. With all other parameters having been pretrained, Phase 3 concentrates on learning the combination layer.

4 Implementation Details

For training, we use a subset of the UniProt dataset which contains millions of protein sequences (Consortium, 2022). Specifically, we train on $623,000$ sequences and reserve $16,000$ sequences as our test set. We truncate the length of each sequence to be 512 amino acids to fit our context window. It is worth noting that, as we are proposing a change to the model architecture, we train from scratch to faithfully assess the effect of the change; however, state-of-the-art PLMs, such as ESM2, have a very large number of parameters and train on many GPUs for many days. Due to the constraints of this project, we conduct a lower-scale training.

We begin by training a character-level BERT from scratch for about 22 epochs. We conduct this training on two A100 GPUs for 48 hours. The model has 12 hidden layers each with 12 attention heads and a context window of 512. We use a hidden size of 768 and an embedding dimension of 60 (due to the small vocabulary size). We then train four more BERT models from scratch, one for each of the auxiliary tokenizers we are using. These training procedures are identical to the character-level training, with the exception of the embedding size, which we set to 768 instead of 60, as each of these models have a larger vocabulary size.

Finally, we train four Ithaca-like models for an additional 24 hours.[3] We initialize them using the character-level BERT model weights (trained in Phase 1) and add in the respective auxiliary-tokenizer embedding tables (trained in Phase 2) as well as a randomly-initialized combination layer, as described in 3.2. Importantly, for the character-level weights, we take the checkpoint of the model after training for 24 hours, so that after another 24 hours of Phase 3 training, the Ithaca-like models have roughly seen the same amount of data as the character-level model which trained for 48 hours.

5 Evaluation

To compare the resulting models, we test on two tasks: masked-token infilling and variant effect pre-

[3]To train these models, we make use of the Desformers repository: `https://github.com/ddevaul/desformers`

Tokenization	All mutations	Insertions only	Deletions only
Char-Only	0.45	0.50	0.45
Char+WP1K	0.46	**0.55**	0.47
Char+WP10K	0.45	0.54	0.44
Char+2-gram	0.45	0.49	**0.48**
Char+3-gram	**0.48**	0.54	**0.48**

Table 2: **AUROC on VEP**. For each tokenization, we show the AUROC across all mutations ($2,500$ sequences), insertions only (937 sequences), and deletions only ($2,248$ sequences).

diction. Similar to the BERT pretraining task (Devlin et al., 2019), masked-token infilling consists of masking a single token in a given sequence and assessing the model's predictions for the missing amino acid.

Additionally, we test our models on variant effect prediction (VEP) which requires the model to predict whether a protein sequence is malignant or benign (Cheng et al., 2023). Following Brandes et al. (2023), we use our trained MLMs to predict pathogenicity without any further training or finetuning by calculating a score known as Pseudo-Log-Likelihood Ratio (PLLR). This score is computed by using an MLM to compute a pseudo-log-likelihood of a sequence and the variant of that given sequence. If the variant sequence has a much lower pseudo-log-likelihood than the original unmutated sequence, this is a zero-shot indicator of malignancy (Brandes et al., 2023).

For the VEP task, we use the ClinVar dataset which contains information about variants of protein sequences and whether those variants are malignant or benign (Landrum et al., 2014). Using this same dataset and different variants of PLLR, Brandes et al. (2023) achieve AUROC scores of around 0.86 for the VEP task using ESM1b.

6 Results

On masked-token prediction ($n = 500,000$), each of the Ithaca-like models has a slight improvement over the character-only baseline (Table 1). Specifically, the WordPiece models outperform the others in the Top 2 through Top 5 accuracies, while the 3-gram model outperforms other models in the Top 1 accuracy. Similarly, for the VEP task, the Ithaca-like models outperform the character-only model across all three subdivisions of mutations (Table 2). These trends suggest that incorporating word-level information within the character-level tokens positively impacts how the model understands sequences.

Among the Ithaca-like models, there does not seem to be a clear winner in the masked-token prediction task. Though the WordPiece models perform well across the Top 2 through Top 5 accuracies regardless of the vocabulary size, the 3-gram model performs the best in Top 1 accuracy, which is arguably the most important accuracy.

The 3-gram model seems relatively competitive in the VEP task as it achieves the highest AUROC for sequences that contain all types of mutations (0.48) and sequences that only contain deletions (0.48) (Table 2). Additionally, for sequences that only contain insertions, the 3-gram tokenizer is still competitive (0.54) with the highest AUROC which was achieved by the 1K WordPiece (0.55). However, the AUROC values overall are low; an AUROC of 0.50 indicates random chance, and current state-of-the-art on the VEP task achieves an AUROC of 0.86 (Brandes et al., 2023). The low AUROC values do make sense, given the scope of our trainings, as VEP is a difficult task which state-of-the-art models struggle with. Nonetheless, the fact that the character-only model achieves the lowest AUROC within most experiments does still point to an advantage in the Ithaca architecture.

The performance of the 3-gram tokenizer across both tasks weakly suggests that naively using fixed-length words as auxiliary tokens is competitive with state-of-the-art tokenization techniques like WordPiece, at least at the scope of training conducted in this paper. Though limited by the size of the models that we train, these results suggest that the type of word-level tokenizer used in tandem with the character-level pretraining may not have a clear impact on model performance. Nonetheless, the results do motivate further experimentation with a wider sweep of auxiliary tokenizers, evaluation metrics, and most importantly, larger models trained for more time on more data.

7 Conclusion

In conclusion, we adapt an MLM architecture, developed for ancient inscription restoration, to protein sequences and provide a unified comparison of a baseline character-level model with models equipped with various auxiliary tokenizers. Based on these experiments, we find that combining character-level embeddings with word-level information marginally improves the model's understanding of protein sequences. Furthermore, we find that among auxiliary tokenizers, there is no clear advantage between using fixed-length n-grams tokens or tokens derived from WordPiece. Additionally, there there is no clear trend in how the vocabulary size of the auxiliary tokenizer affects performance. We note that the scope of our trainings are likely too small to discern trends which might emerge from larger trainings.

We hope that this work spurs future work in evaluating how best to apply NLP techniques to computational biology. In particular, potential improvements and avenues for future work include testing more values of n for the n-gram tokenization and test with larger models. Though we test $n = 2$ and $n = 3$, proteins can be thousands of amino acids long, so an immediate next step would be to test larger values of n for a more complete comparison. Additionally, given our compute constraints and since this paper is mainly intended as a proof of concept, we used a relatively small number of parameters in our models. As a result, it would be interesting to perform the same comparison with larger models.

Author Contributions

Both I.P. and C.B. developed the project goal and structure and wrote code. C.B. ran and debugged the jobs on the Della research cluster. Both I.P. and C.B. contributed to the writing.

Acknowledgements

We would like to acknowledge Desi DeVaul, the creator of Desformers, whose senior thesis on combining character-level & sub-word-level embeddings for ancient Greek models was pivotal for the completion of this project. We would also like to acknowledge Prof. Olga Troyanskaya, Tamjeed Azad, and our classmates in COS 557 for a semester of interesting and collaborative discussions.

References

Ehsaneddin Asgari, Alice C McHardy, and Mohammad R K Mofrad. 2019. Probabilistic variable-length segmentation of protein sequences for discriminative motif discovery (DiMotif) and sequence embedding (ProtVecX). *Scientific Reports*, 9(1):3577.

Yannis Assael, Thea Sommerschield, and Jonathan Prag. 2019. Restoring ancient text using deep learning: a case study on greek epigraphy. *arXiv preprint arXiv:1910.06262*.

Yannis Assael, Thea Sommerschield, Brendan Shillingford, Mahyar Bordbar, John Pavlopoulos, Marita Chatzipanagiotou, Ion Androutsopoulos, Jonathan Prag, and Nando de Freitas. 2022. Restoring and attributing ancient texts using deep neural networks. *Nature*, 603(7900):280–283.

Asa Ben-Hur and Douglas Brutlag. 2006. *Sequence Motifs: Highly Predictive Features of Protein Function*, pages 625–645. Springer Berlin Heidelberg, Berlin, Heidelberg.

Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. *arXiv preprint arXiv:2004.03720*.

Nadav Brandes, Grant Goldman, Charlotte H Wang, Chun Jimmie Ye, and Vasilis Ntranos. 2023. Genome-wide prediction of disease variant effects with a deep protein language model. *Nature Genetics*, 55(9):1512–1522.

Nadav Brandes, Dan Ofer, Yam Peleg, Nadav Rappoport, and Michal Linial. 2022. ProteinBERT: a universal deep-learning model of protein sequence and function. *Bioinformatics*, 38(8):2102–2110.

Jun Cheng, Guido Novati, Joshua Pan, Clare Bycroft, Akvilė Žemgulytė, Taylor Applebaum, Alexander Pritzel, Lai Hong Wong, Michal Zielinski, Tobias Sargeant, Rosalia G. Schneider, Andrew W. Senior, John Jumper, Demis Hassabis, Pushmeet Kohli, and Žiga Avsec. 2023. Accurate proteome-wide missense variant effect prediction with alphamissense. *Science*, 381(6664):eadg7492.

The UniProt Consortium. 2022. UniProt: the Universal Protein Knowledgebase in 2023. *Nucleic Acids Research*, 51(D1):D523–D531.

Desmond Devaul. 2024. Combining tokenization methods towards improved machine learning models for ancient greek.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of* *the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Barbara Graziosi, Johannes Haubold, Charlie Cowen-Breen, and Creston Brooks. 2023. Machine learning and the future of philology: A case study. *TAPA*, 153(1):253–284.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Andrew J Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. 2021. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*.

Melissa J Landrum, Jennifer M Lee, George R Riley, Wonhee Jang, Wendy S Rubinstein, Deanna M Church, and Donna R Maglott. 2014. ClinVar: public archive of relationships among sequence variation and human phenotype. *Nucleic Acids Res.*, 42(Database issue):D980–5.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, et al. 2023. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130.

Dan Ofer, Nadav Brandes, and Michal Linial. 2021. The language of proteins: Nlp, machine learning & protein sequences. *Computational and Structural Biotechnology Journal*, 19:1750–1758.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5149–5152. IEEE.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Juwen Shen, Jian Zhang, Xiaomin Luo, Weiliang Zhu, Kunqian Yu, Kaixian Chen, Yixue Li, and Hualiang Jiang. 2007. Predicting protein-protein interactions based only on sequences information. *Proc. Natl. Acad. Sci. U. S. A.*, 104(11):4337–4341.

Yanbin Wang, Zhu-Hong You, Shan Yang, Xiao Li, Tong-Hai Jiang, and Xi Zhou. 2019. A high efficient biological language model for predicting Protein-Protein interactions. *Cells*, 8(2):122.