CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Multilingual Language Models for the Evaluation and Selection of auto-generated Abstract Wikipedia Articles

Master's thesis in Computer science and engineering

Jiahui Le

Ruxin Zhou

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2022

Master's thesis 2022

Multilingual Language Models for the Evaluation and Selection of auto-generated Abstract Wikipedia Articles

Jiahui Le

Ruxin Zhou

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2022

Multilingual Language Models for the Evaluation and Selection of auto-generated Abstract Wikipedia Articles
Jiahui Le
Ruxin Zhou

Supervisor: Aarne Ranta, Department of Computer Science and Engineering
Examiner: Aarne Ranta, Department of Computer Science and Engineering

Master's Thesis 2022
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Description of the picture on the cover page (if applicable)

Typeset in LaTeX
Gothenburg, Sweden 2022

Multilingual Language Models for the Evaluation and Selection of auto-generated Abstract Wikipedia Articles
Jiahui Le
Ruxin Zhou
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

To enrich Wikipedia to more topics with less cost, Abstract Wikipedia project, an initiative from the Wikimedia foundation, is considered to be created . The general architecture of Natural Language Generation part of the project to automatically generate articles from wiki-data has been basically built. However, the same input wiki-data may be transformed to several sentences with different sentence structures. This thesis built multilingual data sets and utilized Natural Language Processing techniques (e.g. n-gram model and RoBERTa model) to evaluate the quality of these sentences. The report concludes, that a suitable language model is capable of evaluating and selecting auto-generated Abstract Wikipedia articles and has the potential to improve Abstract Wikipedia project. The model performance slightly varies according to the model architecture and the data set.

Keywords: n-gram, RoBERTa, Language Model, Natural Language Processing, Abstract Wikipedia project.

Acknowledgements

We want to express our deep and sincere gratitude to our supervisor Aarne Ranta for the project opportunity along with valuable advice and support that he gives us to proceed and accomplish this project in the past few months. We also would like to thank Denny Vrandečić for providing a presentation on the entire Abstract Wikipedia project and thank Emilio Jorge for offering help for the server.

Our thanks also go to our friends for their support for providing input examples and manual evaluation criteria for our extrinsic model evaluation.

Finally, we would like to express our gratitude towards our parents and our friends for their encouragement which supports us in completing this project.

Jiahui Le & Ruxin Zhou, Gothenburg, June 2022

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background

Wikipedia is a multilingual online encyclopedia, generated and maintained by a community of volunteers by using a wiki-based editing system. It covers a variety of topics, each demonstrated by multiple languages, and provides basic and advanced knowledge of these topics.

However, if Wikipedia is enlarged to cover most of topics (e.g. 20,000,000 topics) with most of languages (e.g. 300 languages), the cost of Wikipedia will be tremendous if the article generation is only implemented by Wikipedians (people contribute to Wikipedia): at least $20,000,000 \times 300 = 6,000,000,000$ Wikipedia articles need to be created. Thus, to enrich Wikipedia to more topics with less cost, Abstract Wikipedia project [2] is considered to be created: by using Grammatical Framework (GF) [3], the concise wiki-data with a fixed format can be transformed automatically into a wiki-style article. Therefore, the cost can be reduced to $20,000,000 + 300 \approx 20,000,000$, much smaller than before. Abstract Wikipedia project is a novel project which can enrich the content of Wikipedia articles with various languages in a more doable way by the communities. It is an initiative from the Wikimedia foundation. The general architecture of Natural Language Generation (NLG) part of the system to automatically generate articles from wiki-data has been basically built but there is still some work to do.

For example, with the same set of wiki-data, the current NLG system can generate different sentences with different ways of expression and grammatical structures but all of them convey the same information. Thus, it is essential to assess which is the most suitable one. Considering that the aim of the project is to automatically generate Wikipedia articles and reduce the wiki-dictionary cost, the best expression is the one that is most similar to that provided by the original Wikipedia community. Therefore, by evaluating and selecting a more suitable sentence structure, the final result of Abstract Wikipedia project can in turn be improved.

In recent years, Natural Language Processing (NLP) has shown remarkable improvement and enabled impressive breakthroughs. NLP is a subfield of linguistics and machine learning (ML), focusing on the interactions between computers and human languages. It is widely used to program computers to process and implement analysis on a large amount of natural language data and to teach the computer to ‘understand’, ‘categorize’ and ‘organize’ the data. Thus we propose to utilize some state-of-the-art NLP techniques to train language models. And then we will use the trained language model to evaluate and select the best sentence among automatically-generated sentences with the same meaning. Furthermore, by inserting the best language model after the NLG process into the current Abstract Wikipedia System, we can improve the quality of the auto-generated articles and thus improve the whole system.

Therefore, the aim of this study is to investigate the ability of language models of the evaluation and selection of auto-generated Abstract Wikipedia articles.

1.2 Related Work

1.2.1 Multilingual Natural Language Generation

NLP is the set of methods that make machines understand texts and respond to them as humans do [4]. NLP can be split into two parts. One is Natural Language Understanding (NLU) and the other is Natural Language Generation. NLU analyzes the meaning of texts from the semantic and syntactic points of view [5]. NLG solves the problem through establishing software systems which are able to create readable, consistent and human-like texts in English and other human languages [6]. Generally, NLG is able to deal with a variety of tasks: translating a sentence or a paragraph from one language to another, question-answering, etc. The input of NLG can be a data set, an image or a natural language prompt and the output can be a sequence of readable, consistent and human-like texts [6]. In other words, NLG writes while NLU reads.

With the development of multilingual NLG techniques, nowadays one of the most well-known implementation platforms is Grammatical Framework (GF). GF is a programming language for multilingual grammar applications [3]. It is a development platform for natural language grammars and machine translation. One of the aims of GF is to reduce the expense of development. Thus, module system that supports division of labour, functional programming that enables powerful abstractions and information extraction which changes resources from other formats to GF are designed in GF [3]. A GF grammar consists of two parts: abstract syntax and concrete syntax. The former relates to trees which capture semantically-relevant structure and the latter relates to linear language strings [7]. According to this characteristic, Kaljurand et al. [7] presented a semantic wiki system with underlying controlled multilingual grammar implemented by GF. One of the remarkable characteristics of GF is that it provides a reusable grammar library which covers 30 natural languages. Due to the feature that the library can be accessed independent of language, GF is able to parse and generate texts simultaneously in multiple languages while using a language-independent representation of meaning [8].

However, there is a big problem of NLG. Reiter et al. [9] stated that according to their experiments, it was tough to gain accurate knowledge for NLG systems.

Different factors might lead to this problem, within which, the basic one was the desire that the writing task can be automatically finished. Authors came up with four factors: 1) complex circumstances of NLG tasks, 2) not done by human, 3) the understanding problem, 4) multiple solutions allowed [9]. Go into details on the fourth point: during a translation task, the same input sentence can be transformed to multiple plausible translated sentences as the output, but it is hard to assess which one is the best. This is also an essential problem that affects the quality of the current Abstract Wikipedia project.

Thus, since Abstract Wikipedia project is based on multilingual NLG, it also has the aforementioned problem: although the automatically-generated texts can convey the semantic contents, it may be difficult to read and understand due to the sentence structure and phraseology.

1.2.2 Multilingual Data Sets

There are a wide range of multilingual data sets all over the world. Hu et al. [10] came up with a massively multilingual benchmark: XTREME. It is able to evaluate the cross-lingual generalization across 40 languages and 9 tasks. The 9 tasks can be divided into 4 categories related to classification, structure prediction, question answering and sentence retrieval. Authors also supply an online platform and leaderboard for evaluating multilingual models. XTREME estimates the cross-lingual generalization ability of the model more accurately and supplies a broader scope and more fine-grained analysis tools.

Ladhak et al. [11] introduced a large-scale and multilingual data set with 18 languages called WikiLingua. It works for the cross-lingual abstractive summarization system. It is established based on WikiHow, and there is an online resource on how-to guides on different topics, written and reviewed by human users. Thus, there are jointly written how-to guides with gold-standard summary alignments across 18 languages based on English. According to authors, at that time, WikiLingua was the biggest cross-lingual abstract summarization data set with parallel articles and summaries.

Wiki-40B, introduced by Guo et al. [12], is a multilingual data set with more than 40 languages. It reconstructed the articles with structural markers including `_START_ARTICLE_`, `_START_SECTION_`, `_START_PARAGRAPH_` and `_NEWLINE_`. The final data set consists of approximately 40 billion characters which come from 19.5 million Wikipedia webpages. It is easy to apply this data set to other tasks, but the problem is that it misses some sentences belonging to the original Wikipedia articles. For example, sentences in the `list` are ignored. Sometimes, the format of sentences is not exactly what they should be.

1.2.3 Multilingual Language Model

Nowadays, Hinton et.al [13] introduced a method called 'first pre-train and then fine-tune' to train new models. The goal is to take advantage of a large number of unlabeled texts and build a general language-understanding model before it is fine-tuned for different NLP tasks. There are two prevalent models. One is Masked Language Modeling (MLM) and the other is Causal Language Modeling (CLM). In this project, RoBERTa belongs to MLM and n-gram belongs to CLM.

Basically, MLM uses 'mask' markers to randomly mask a certain percentage of tokens in a given sentence. The goal is to predict each originally-masked word according to its context. Intrinsically, it is a bidirectional model because the masked word can be learnt from other words both from left and right. A typical model is Bidirectional Encoder Representations from Transformers (BERT). The advantage of MLM is that it can know the position information of the full sentence and thus can alleviate the position bias [14]. MLM can be applied on text categorization, question answering System, and auto correct and auto prediction.

The principle of CLM is to predict the masked token in a given sentence. The model does the similar thing as MLM, but CLM only considers words that occur in one direction (left or right). Typical models are Generative Pre-trained Transformer (GPT) family of language models. It helps people understand the causal structure of the data generating process to find potential confounding factors [15]. Language Translator and Chatbot are the two typical applications of CLM.

1.3 Research Goal

The first goal is to establish the multilingual data sets with 46 languages: including widely-used languages (e.g. English, Chinese, French, etc.) and small languages that other data sets do not include (e.g. Waray, Ilocano, Cantonese, etc.). A huge number of Wikipedia articles will be crawled from Wikipedia webpages. Each language will have its own data set created according to the language characteristics hidden in Wikipedia articles (e.g. html formats, punctuation, tokenization, etc.).

The secondary goal of this thesis is to train the multilingual language model and use the best model to evaluate the probability of several sets of sentences. Sentences in each set have the same semantic meaning but different grammatical structures. The probability will be transformed to a pre-defined score or perplexity. The training samples are sentences from Wikipedia. The standard of the best sentence is the original Wikipedia version. In Abstract Wikipedia project, different sentences generated by the NLG task with the same meaning will be evaluated by aforementioned models and the corresponding outputs are the scores showing how probably the original Wikipedia could generate it. The higher the score is, the more similar to Wikipedia style, and the better the quality of the input sentence is. Then the best language model among state-of-the-art NLP techniques should be selected. Models will be evaluated from two aspects. First is to use numerical metrics to compare these models. Second is to manually compare several sentences and choose the best one.

The final goal is to combine the best language model with the NLG part of Abstract Wikipedia project. Then the total process should be: given a fixed format of wiki-data, we can first use the NLG part of our system to generate several wiki-style sentences that convey the information, and then use the NLP part to implement data analysis and choose only one sentence that is smoother and more Wikipedia-like. Thus, after several sets of sentences with the same meaning being generated, our models can select the best sentence in each set and make up a complete article with a detailed report. Therefore, the quality of the current Abstract Wikipedia project can be improved.

To accomplish the goals, the following research questions have to be answered:

- **What language models perform best depending on the wiki-style sentences?**
- **How does the language model perform differently when trained with different languages?**

1.4 Motivation

The project arises with Abstract Wikipedia project. The motivation of Abstract Wikipedia project is to enlarge the content of Wikipedia articles with less cost. And to improve the performance of NLG part of Abstract Wikipedia project, a suitable language model is needed to evaluate the quality of automatically-generated sentences, which motivates us to propose and construct research on this project.

1.5 Scope and Limitation

The project has a focus on improving the performance and accuracy of auto-generated Abstract Wikipedia articles by applying language models: 1) n-gram, 2) RoBERTa. The first limitation is that although there are many kinds of language models, only the two models we mentioned before will be explored. Thus, we may miss some model structures with better performances. Second, considering that the manual evaluation is also included in the model evaluation stage, different people may have different criteria and it may lead to additional bias and restrict the performance of our model. The third limitation is generated from the nature of the multilingual model. During the data collection step, some small languages, such as Sicilian and Aragon, only have few original Wikipedia articles. Therefore, the amount of data of these small languages are minor. Although the same model structure is applied on the 46-language data sets in the same way, the model-training on small-language data sets is insufficient. Thus, small-language language models still have a lot of room to improve.

1.6 Target Group

The project aims to provide insights on multilingual data sets and multilingual language models for academic readers, such as computer engineers and computer scientists. It proposes an uncommon application of language models, which is to evaluate the quality of sentences generated from an NLG project. We hope it would provide some enlightenment if academic readers would like to continue studies on multilingual language models.

1.7 Outline

The rest of this thesis report comprises the following chapters:

Chapter 2 - Theory: In this section, the theory on n-gram model, RoBERTa model and model evaluation will be explained.

Chapter 3 - Methods: This section will demonstrate research methods and implementation in detail.

Chapter 4 - Results: The data collected during the experiments will be presented without further analysis.

Chapter 5 - Conclusion: This section first will discuss the results and give an introduction to what could be done to further continue on the topic of language models of Abstract Wikipedia project. Finally, it will conclude the research.

2
Theory

In the following sections, the theoretical knowledge of two language models: n-gram model and RoBERTa model are presented in detail.

2.1 n-gram

2.1.1 Basic Theory

An n-gram model is a probabilistic language model. It assigns the probability to the next item given the previous context, and thus can be used to estimate the probability of a complete sentence, or an article with several sequences.

Considering a word sequence $\{w_1, w_2, w_3, \ldots, w_n\}$, the probability that the sentence can be generated, $P(X_1 = w_1, X_2 = w_2, X_3 = w_3, \ldots, X_n = w_n)$ (abbreviated as $\boldsymbol{P(w_1, w_2, w_3, ..., w_n)}$), is computed based on chain rule of probability, Markov assumption and maximum likelihood estimation.

Chain rule of probability helps to decompose the joint probability of a sentence into the product of some conditional probability of a word given all of its previous words, thus

$$\begin{aligned} P(w_1, w_2, w_3, ..., w_n) &= P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_{1:2}) \cdots P(w_n|w_{1:n-1}) \\ &= \prod_{K=1}^{n} P(w_K|w_{1:K-1}). \end{aligned} \tag{2.1}$$

Markov assumption helps to simplify the computation. The assumption presents that to predict the future, only the present state does matter. Based on this assumption, in order to predict the next word in the sentence, it is enough to look into the past several words instead of its entire context. Therefore, the conditional probability in Eq. 2.1 can be reformulated as follows:

$$\begin{aligned} &\text{bigram probability(considering past one word)} : P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-1}) \\ &\text{trigram probability(considering past two words)} : P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-2:n-1}) \\ &\cdots \\ &\text{N-gram probability(considering past N-1 words)} : P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-N+1:n-1}) \end{aligned} \tag{2.2}$$

Maximum likelihood estimation (MLE) helps to compute the above-mentioned probability: compute the count of N-grams $C(w_{n-N+1:n-1}w_n)$, and normalize the counts to the range from 0 to 1 by dividing the sum of all of the N-grams that share the same first (N-1) words, namely, the count of (N-1)-grams for the word w_{n-1}, represented as $C(w_{n-N+1:n-1})$. Therefore, Eq. 2.2 can be reformulated as follows:

$$\begin{aligned}
&\text{bigram probability} : P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})} \\
&\text{trigram probability} : P(w_n|w_{n-2:n-1}) = \frac{C(w_{n-2:n-1}w_n)}{C(w_{n-2:n-1})} \\
&\dots \\
&\text{N-gram probability} : P(w_n|w_{n-N+1:n-1}) = \frac{C(w_{n-N+1:n-1}w_n)}{C(w_{n-N+1:n-1})}
\end{aligned} \tag{2.3}$$

Thus, the formulation of the joint probability of a sentence with n-gram theory:

$$P(w_1, w_2, w_3, ..., w_n) = \prod_{K=1}^{n} \frac{C(w_{K-N+1:K-1}w_K)}{C(w_{K-N+1:K-1})}. \tag{2.4}$$

In this project, instead of using raw probability presented in Eq. 2.4, **log probability** is used, which is often applied in scientific research. The motivation of using log probability is shown as follows:

- **Accuracy**: Considering that the probabilities are smaller than or equal to 1, with the number of next-word probabilities that need to be multiplied increasing, the final product, namely the probability of the sentence, becomes smaller and smaller. When a computer represents very small numbers, the numerical stability may decrease. Using log probability can, to some extent, avoid too small numbers and thus increase the computation accuracy.
- **Speed**: Multiplication is more expensive than addition. Using log probability can convert multiplication (a product of multiple numbers) to addition (a sum of multiple numbers) and thus can improve the computation efficiency.

Log probability is formulated as follows:

$$P(w_1, w_2, w_3, ..., w_n) = \exp(\log \prod_{K=1}^{n} \frac{C(w_{K-N+1:K-1}w_K)}{C(w_{K-N+1:K-1})}) = \exp(\sum_{K=1}^{n} \log \frac{C(w_{K-N+1:K-1}w_K)}{C(w_{K-N+1:K-1})}) \tag{2.5}$$

2.1.2 Evaluation of Sentence Fluency

Divide the sentence into a sequence of words with length n, $\{w_1, w_2, w_3, \dots, w_n\}$. Next, use special tokens (e.g. `<s>` and `</s>`) to mark the boundary of each sentence. Then the probability of this sentence can be calculated by Eq. 2.4. The loss, perplexity (ppl) and fluency score of this sentence are thus defined as follows:

$$\begin{aligned}
\text{Sentence loss} &= -\log P(w_1, w_2, w_3, ..., w_n) = -\sum_{K=1}^{n} \log \frac{C(w_{K-N+1:K-1} w_K)}{C(w_{K-N+1:K-1})} \\
\text{Perplexity} &= P(w_1, w_2, w_3, ..., w_n)^{-\frac{1}{n}} = \sqrt[n]{\prod_{K=1}^{n} \frac{C(w_{K-N+1:K-1})}{C(w_{K-N+1:K-1} w_K)}} \\
\text{Fluency score} &= P(w_1, w_2, w_3, ..., w_n)^{\frac{1}{n}} = \sqrt[n]{\prod_{K=1}^{n} \frac{C(w_{K-N+1:K-1} w_K)}{C(w_{K-N+1:K-1})}}
\end{aligned} \tag{2.6}$$

The lower the perplexity is (alternatively, the higher the fluency score is), the more probably that the sentence could be generated.

2.1.3 Treatment of Zeros

Considering that the probability of an entire word sequence (or an entire test set) is a product of multiple less-than-one conditional probabilities, if one of them, say $P(w_j|w_{j-N+1})$ is zero, the entire probability will be zero. The zero is meaningless not only because it leads to infinite perplexity, but the information from the rest of the sentence (or the test set) is lost as well.

The zeros are originated from the limited training corpus and are directly caused by the following two reasons: the first reason is unknown words, and the second one is known words but unseen contexts. Suitable actions should be taken to deal with these issues.

Unknown words: Considering that the training corpus is limited, some words in the test set may not occur in the training set. These words are thus defined as out of vocabulary (OOV) words, or unknown words. To deal with OOV words, a pseudo-word `<UNK>` is introduced. In the training set, if a word appears fewer than p times, where p is a small number, it will be replaced with a `<UNK>` mark. Then treat the `<UNK>` mark as a regular word and train the language model with the processed training set. The probabilities of OOV words can be trained.

Known words but unseen contexts: It is possible that the word in the test set occurs in the training set but the contexts in the two sets are different. In general, for an N-gram model, the word w_K only appears after context $\{w_{K_1}, w_{K_2}, ...w_{K_{N-1}}\}$ in the training set, but it appears after context $\{w_{P_1}, w_{P_2}, ...w_{P_{N-1}}\}$ in the test set. Thus, in the test set, the N-gram probability $P(w_K|w_{P_1:P_{N-1}}) = \dfrac{C(w_{P_1:P_{N-1}} w_K)}{C(w_{P_1:P_{N-1}})} = 0$, leading to a zero-probability of this sentence and the entire test set.

Smoothing or Discounting is a good way to deal with this issue. It is to shave off a bit of probability mass from some more frequent events and give it to the events that have not happened before.

Laplace smoothing is one of the simplest methods, which is to assume every event, no matter it is seen or unseen, occurred once more than it did in the training data. Thus, if the total number of word tokens in the training corpus is V, namely the vocabulary size V, Eq. 2.3 is reformulated as follows:

$$\text{N-gram probability} : P_{Laplace}(w_n|w_{n-N+1:n-1}) = \frac{C(w_{n-N+1:n-1}w_n) + 1}{C(w_{n-N+1:n-1}) + V} \tag{2.7}$$

However, Laplace smoothing moves too much probability mass from seen to unseen events. One alternative is **Add-k smoothing**, which moves a little bit less than Laplace smoothing. k is always a small number ranging from 0 to 1. Therefore, if the total number of word tokens in the training corpus is V, namely the vocabulary size V, Eq. 2.3 is reformulated as follows:

$$\text{N-gram probability} : P_{Add-k}(w_n|w_{n-N+1:n-1}) = \frac{C(w_{n-N+1:n-1}w_n) + k}{C(w_{n-N+1:n-1}) + kV} \tag{2.8}$$

2.2 RoBERTa

RoBERTa, which stands for **R**obustly **O**ptimized **BERT a**pproach, is a pretrained language model with a similar model architecture, but a different pretraining recipe compared to **BERT** (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers). It proves to perform more outstandingly than the originally-implemented BERT in many downstream tasks [16].

2.2.1 Model Architecture

The model architecture, as shown in Fig. 2.1, is a multi-layer bidirectional Transformer architecture [1], which is ubiquitously used in NLP. The Transformer follows the overall encoder-decoder structure using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder. For a detailed review of the model architecture, please refer to [1].

2.2.2 Pretraining Procedure

The recipe for pretraining RoBERTa is based on that for pretraining BERT but is modified from some aspects.

2.2.2.1 Training Objectives

BERT is pretrained by two unsupervised tasks. One is Masked Language Modeling and the other one is Next Sentence Prediction [17]. On the basis of that, RoBERTa makes some modifications [16].

Masked Language Modeling (MLM):
To train each token from both left-to-right and right-to-left directions, MLM is implemented. The basic idea is to randomly mask a certain percentage of the input

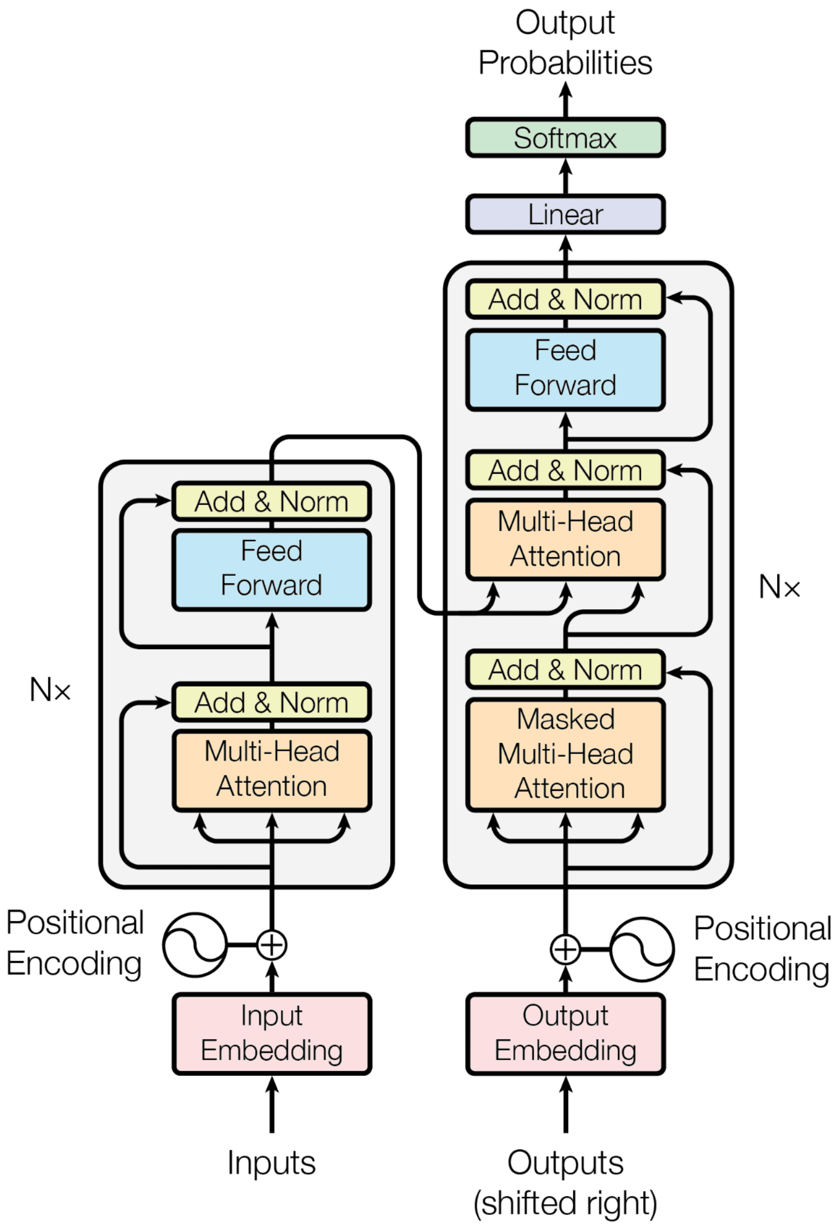


Figure 2.1: The Transformer - model architecture [1].

tokens and then predict the masked tokens according to other given tokens. BERT selects 15% of the input tokens at random, replaces each with a `[MASK]` token and predicts them with cross entropy loss. To avoid mismatching between pretraining and finetuning, another random element is inserted. If a token is selected for prediction, there is 80% probability of it being replaced by `[MASK]`, 10% probability of it being replaced with a random token, and 10% probability of it being left unchanged.

The original implementation is a *static* masking: masking only once at the data preprocessing stage. However, in RoBERTa, a *dynamic* masking is utilized: masking every time a sequence of tokens is fed to the model. Dynamic masking proves to be slightly better than static masking [16].

Next Sentence Prediction (NSP):
To understand sentence relationships and improve some specific downstream tasks (e.g. Question Answering), NSP is implemented. BERT selects two sentences s_A and s_B from the text corpus at random. There is 50% probability of the two sentences following each other and they will be concatenated as a single input sequence labeled with `IsNext`. Likewise, there is another 50% possibility that the two sentences are separate in the text corpus and they will be concatenated as an input labeled with `NotNext`. The input format of the positive example is as follows:

$\texttt{Input} = \texttt{[CLS]}\ A_1, A_2, \ldots, A_{l_A},\ \texttt{[SEP]}\ B_1, B_2, \ldots, B_{l_B},\ \texttt{[EOS]}.$
$\texttt{Label} = \texttt{IsNext}$

NSP objective is a binary classification loss to predict whether two sentences are contiguous in the original text corpus.

However, RoBERTa removes the NSP objective. After comparing the following four training formats: `segment-pair + NSP` (BERT version), `sentence-pair + NSP`, `full-sentences` and `doc-sentences`. It is demonstrated that the fourth one performs best, and the third one is in second place [16]. Considering that the format of `doc-sentences` leads to different batch sizes, `full-sentences` format is implemented in RoBERTa: sentences sampled contiguously from one document are grouped together as inputs. The length of each input sequence is 512 tokens. If the end of one document is reached, a `[SEP]` token will be added as a separator and then sentences will be sampled contiguously from a new document.

2.2.2.2 Tokenizer

Both BERT and RoBERTa utilize the subword tokenization algorithm which is a hybrid between word-level and character-level tokenization. First, space tokenization is implemented to split sentences into words. Then, a dictionary is built, with each word as key and the frequency of it occurring in the training data as value. Next, each word in the dictionary is represented as a sequence of characters and all the unique characters form the base vocabulary. Then a merge rule is learnt. According to the rule, a merge is operated, and the newly-formed symbol will be added to the vocabulary. The above-mentioned operation is iteratively implemented until the size of the final vocabulary (i.e. the size of base vocabulary *plus* the number of merge operations) reaches the desired value.

The differences between BERT and RoBERTa are the format of base vocabulary and the merge rule.

BERT uses **WordPiece** tokenizer [18]. The base vocabulary is made up with characters. And the merge rule is that only if the symbol pair maximizes the likelihood of the training data, can it be merged and added to the vocabulary.

RoBERTa uses **Byte-level Byte-Pair Encoding (BPE)** tokenizer [19]. It uses bytes as the base vocabulary. Compared to character-level base vocabulary, it has two advantages. First, the size of base vocabulary is forced to be 256. RoBERTa thus can realize subword tokenization with a modest vocabulary size of 50K. The second advantage is that it can tokenize any input text without introducing the unknown symbol, `<UNK>`. Besides, the merge rule is to count each possible symbol pair and replace each occurrence of the most frequent symbol pair `('A', 'B')` with a new symbol `'AB'`.

2.2.2.3 Batch Size

Ott et al. [20] shows that in Neural Machine Translation, when the learning rate is appropriately increased, training with large mini-batches can result in the improvement in both optimization speed and end-task performance. RoBERTa [16] demonstrates that for the MLM objective, training with large mini-batches can decrease the perplexity and improve the end-task accuracy. Thus, in RoBERTa pretraining, the size of mini-batches is increased to 8,000.

2.2.3 Evaluation of Sentence Fluency

In Causal Language Modeling (CLM), such as n-gram and GPT-2, the probability that the sequence of words can form a sentence is calculated based on chain rule of probability as shown in Eq. 2.1. In these unidirectional language models, the context of each token is equivalent to all of its previous words. However, in the bidirectional language model, the context of each token includes not only the previous words but also the words after it. Thus, the chain rule is not precisely applicable, but we can simply adopt the following approximation:

$$\begin{aligned} P(w_1, w_2, w_3, ..., w_n) &\approx \prod_{K=1}^{n} P(w_K|\text{context}_K) \\ &= \prod_{K=1}^{n} P(w_K|w_1, w_2, \ldots, w_{K-1}, w_{K+1}, \ldots, w_n) \end{aligned} \tag{2.9}$$

The model only provides prediction scores for each token in the vocabulary (i.e. $f_{1,C_K}, f_{2,C_K}, \ldots, f_{V_n,C_K}$ with V_n as the vocabulary size and C_K as the context of word w_K: $\{w_1, w_2, \ldots, w_{K-1}, w_{K+1}, \ldots, w_n\}$). To calculate the probability of each token in the vocabulary given a fixed context, `SoftMax` is applied:

$$P(w_K|w_1, w_2, \ldots, w_{K-1}, w_{K+1}, \ldots, w_n) = \frac{e^{f_{w_K, C_K}}}{\sum\limits_{j=1}^{V_n} e^{f_{j, C_K}}} \tag{2.10}$$

Thus, similar to Eq. 2.6, loss, perplexity and fluency score of the sentence are thus defined as follows:

$$\begin{aligned}
\text{Sentence loss} &= -\log P(w_1, w_2, w_3, ..., w_n) = -\sum_{K=1}^{n} \log \frac{e^{f_{w_K, C_K}}}{\sum\limits_{j=1}^{V_n} e^{f_{j, C_K}}} \\
\text{Perplexity} &= P(w_1, w_2, w_3, ..., w_n)^{-\frac{1}{n}} = \sqrt[n]{\prod_{K=1}^{n} \frac{\sum\limits_{j=1}^{V_n} e^{f_{j, C_K}}}{e^{f_{w_K, C_K}}}} \\
\text{Fluency score} &= P(w_1, w_2, w_3, ..., w_n)^{\frac{1}{n}} = \sqrt[n]{\prod_{K=1}^{n} \frac{e^{f_{w_K, C_K}}}{\sum\limits_{j=1}^{V_n} e^{f_{j, C_K}}}}
\end{aligned} \tag{2.11}$$

The lower the perplexity is (alternatively, the higher the fluency score is), the more probably that the sentence could be generated.

2.3 Model Evaluation

Considering that there are many sets of hyperparameters, it is essential to find a way to choose the best model with the most suitable hyperparameters (e.g. the value of N and Laplace for add-k smoothing in n-gram model, the value of training epochs and learning rate in RoBERTa model). In general, there are two ways to do the model evaluation: extrinsic evaluation and intrinsic evaluation.

Extrinsic evaluation is an end-to-end evaluation on a real task [21]. In this project, it is to embed each model into Abstract Wikipedia project and evaluate the quality of several sets of sentences, sentences in each set with the same information but different sentence structures. Then compare the performance of each language model in Abstract Wikipedia project. This is the only way to directly compare the performance of language models. However, it requires high computational resources and might be very slow [22]. In this project, it also requires a large amount of human resource to provide example sets of sentences and the corresponding manual evaluation results as shown in Section 4.3.

Intrinsic evaluation is an evaluation on a specific, intermediate task [21]. The computation is faster and more convenient than the extrinsic one. The intrinsic evaluation is reasonable as long as the evaluation criterion is positively correlated with the performance on the final task. In this project, it is to train the model on a training set and then evaluate the performance of each model on a validation set with unseen data. The model that assigns a higher probability, namely a lower perplexity and a high fluency score, on the validation set, fits the validation set better and thus is a better model.

3 Methods

The following chapter describes the research method and some detailed implementation of this project. The implementation part can be further divided into two parts. The first part is to collect and create data sets. There will be a total of 46 different data sets to collect and explore, each for one language. The second part is to utilize the data, train different language models and choose the best model.

3.1 Research Methods

3.1.1 Literature Study

Literature Study can help to have a grasp of the overall advantages and applications of language models and also to achieve a better understanding of several state-of-the-art language models. Thus, according to the project goal, unsuitable models can be filtered at the very beginning and the candidates of the most suitable language model can be chosen for detailed research.

In addition, from literature study, one can not only gain knowledge on how to build an n-gram language model and a RoBERTa language model, but can also learn how to apply these language models on a new data set. More details about different language models are presented in Chapter 2.

3.1.2 Design Science

After the study of literature, the Design Science of training multilingual language models for the evaluation and selection of auto-generated Abstract Wikipedia articles to improve the performance of the whole Abstract Wikipedia project states the workflow as follows [23]:

1. Collect Data
2. Analyze Data
3. Apply or design a language model
4. Validate and choose the best model
5. Deploy

Regarding data collection, the raw data was collected from Wikipedia. The final data set is made up of 46 individual data sets, each corresponding to one language.

After obtaining the multilingual data set, exploratory data analysis is conducted. Detailed results are presented in Chapter 4. The requirement of conducting data analysis is originated from the tight relationship between the data and the model. It is essential to understand the features of our data before the seeking of an appropriate language model.

After the exploratory data analysis, two or more candidates of the most suitable language models will be applied on the training set. At the same time, a robust validation should be applied to evaluate the model performance. If the validation result is not reasonable, one can take some action to do modification, such as attempting a different model architecture, changing model hyperparameters or adding more modules to the current model structure. During this process, the language model can be optimized until the best model is achieved.

Finally, the best language model can be encapsulated and deployed into the real-life application.

3.2 Implementation

The following sections describe how data is collected.

3.2.1 Data Collection

The raw data is available online. It can be crawled from Wikipedia. The data then should be tokenized to sentences. Finally, it should be split into two subsets called training set and validation set.

3.2.1.1 Crawling

The principle is to crawl the contents in HTML elements `<p>` on Wikipedia webpages. The main method we used for crawling is to apply `lxml` library with the help of `XML` Path Language (`XPath`) [24]. Take English language as an example. The process of crawling English Wikipedia articles is shown as follows:

Step 1. Remove useless sections from Wikipedia webpages
First, the whole contents on Wikipedia webpage are crawled to the local. Then, considering that the contents in some sections, such as 'Reference', 'External Links' and 'See also' only consist of words, phrases and url links, which are not relevant to language models, these sections will be removed.

Step 2. Remove reference citations
The reference citations from Wikipedia webpages are usually composed of supscript tags. There are different kinds of supscript tags. The following list shows some typical cases. To solve the problem, tags in the path '`//sup//*`' should be removed.

1. [1] or [a]
2. [1] : i

3. [citation needed] or [better source needed]

Step 3. Remove HTML formats
There are various HTML formats on one Wikipedia webpage. For instance, on the webpage about country such as United States, there is a coordinator (Coordinates: 40°N 100°W) of this country on the top right of the webpage. Since these formats are useless, they should be eliminated.

Step 4. Replace HTML entities
Basically, all HTML files may reserve these HTML entities. An HTML entity relates to a string that begins with an ampersand ('&') and ends with a semicolon (';'). These characters are usually meaningless. Thus, we replace an HTML entity such as   and with an empty string or a string only with a space to keep the format the same as how it is shown on the webpage.

Step 5. Restructure HTML elements
Basically, lists should be kept following certain rules. To make tokenization simple, the HTML element <p> with the token '\n' will be added after each item in one list. In addition, for the HTML element called 'blockquote', usually there is no space between the last character in the block and the first character in the new line. Therefore, an HTML element <p> with a space will be added after each item in one list.

Step 6. Reserve useful contents
Only contents from HTML elements <p>, / and / are kept to the next step.

Based on the above-mentioned process, some individual modifications are made for each of the 46 languages. Some typical cases are shown in Table 3.1.

Language	Problem	Solution
Spanish	an HTML entity called &ZeroSpaceWidth	replace it with a string only with space.
Japanese	quotes (blockquote)	an HTML element <p> with the only token '\n' is added after the blockquote.
Belarusian	quotes (blockquote)	an HTML element <p> with the only token '\n' is added before and after the blockquote.
Scottish	superscript (e.g. :274)	replace the words in the <sup> tag with None.

Table 3.1: Typical problems encountered in other languages during the crawling process and the corresponding solutions.

3.2.1.2 Tokenization

The main principle is to tokenize the data (texts) to sentences by the full stop '.' and save the sentence if the last character of it is full stop '.', right bracket ')', single quote '’' or double quote '"'. Take English as an example. The process of tokenization is as follows:

Step 1: Data preprocess

In this step, data preprocessing is handled by `Reg expression` from Python library [25]. Generally, this step handles with some strange formats. For instance, if there is no action on the originally-connected brackets and quote markers, these two elements cannot be kept in one sentence after tokenization. `Reg expression` is used to solve this kind of problems. The basic method is to add a token '`\n`' between the full stop '`.`' and brackets or quotes (e.g. '`)`' and '`"`'). After that, originally-connected brackets or quotes can be kept in one sentence. Other languages and different kinds of brackets and quotes are dealt with in a similar way.

Based on previously-mentioned methods, some individual modifications are also applied to other languages. Table 3.2 presents some typical examples.

Language	Problem	Example	Solution
Norwegian	'Wrong line feed' problem when a dot after a date number	14. mars 1879–18. april 1955.	find this format of strings replace the dot with `@*`
Bosanski	'Wrong line feed' problem when a dot after a year number	Troilus and Cressida; 1601. - 1602.	find this format of strings replace the dot with `@#*`
Polish	'Two formats of ellipsis': ... and …	Tak jest …	find this format of strings replace … with `@&*`
Ukrainian	'Two-continuous dots' problem when a sentence ends with an abbreviation	H. e..	find this format of strings replace the first dot with `@%^`

Table 3.2: Typical problems encountered in other languages during the tokenization process and the corresponding solutions.

Step 2: Tokenization

The basic method is to train a new tokenizer based on Natural Language Toolkit (`nltk`) from Python library [26], where tokenization is only determined by a full stop.

However, abbreviation is the main problem. There may be one or more full stops (or dots) in one sentence because of the existence of abbreviations. If special methods are not used to dispose it, texts will be wrongly tokenized. To solve this problem, a `.lex` file is created for each language. The .lex file is used to save all abbreviations and then a new tokenizer is trained based on it to ignore the related full stops. However, Chinese family languages (i.e. simplified Chinese (zh-cn), traditional Chinese (zh-mo), classical Chinese (zh-classical) and Cantonese (zh-yue)), Korean language and Japanese language are processed in different ways. Since Chinese family languages and Japanese language use '。' instead of '`.`' to mark the end of the sentence, abbreviation is not a big issue. These languages are directly tokenized by '。'. Different from all the languages mentioned above, a direct approach is applied in Korean language, which is to find all abbreviations by Reg expression in raw data and replace the dots with other characters. These characters should have no influence on the whole data and will be replaced back to the original tokens when written to the file.

Table 3.3 presents three pairs of example texts. Texts in the left column are collected after raw tokenization, and texts in the right column will be finally used for the language model.

Data set after raw tokenized	Our data set
In 1921, U. S. President Warren G. Harding received ...	In 1921, U.S. President Warren G. Harding received ...
(The host nation's team is ... as the 24th slot.)	(The host nation's team is ... as the 24th slot.)
Albert Einstein (14. mars 1879–18. april 1955) var ein ...	Albert Einstein (14. mars 1879–18. april 1955) var ein ...

Table 3.3: Comparison between the texts after raw tokenization and the final texts used for the language model.

3.2.1.3 Dataset split

After data preprocessing, for each language, sentences in the data set are randomly shuffled and then split into two subsets. A training set with the size of 0.9 (90 %), while 0.1 (10 %) is assigned to the validation set.

3.2.2 Language Model

3.2.2.1 n-gram

The process of the training and validation of the n-gram model is as follows:

Step 1. Data preprocess
Data preprocessing is implemented based on the previously-built training and validation set. For n-gram ($n > 1$) model, each sentence is augmented with $(n - 1)$ `<s>` at the beginning of the sentence as the `SOS` token, marking that a new sentence starts. Each sentence is augmented with a `</s>` at the end of the sentence as the `EOS` token, marking that the sentence is finished. Then the sentence will be decomposed into a sequence of words. In most cases, decomposition is implemented according to space. And punctuation will be considered as independent words.

Chinese family languages and Japanese language are different because they have no space in one sentence. Thus, two approaches are applied on these languages for tokenization. The first approach is to treat each character as a word and split the sentence character by character. The second one is to use a language-specific text segmentation library: 'jieba' [27] for Chinese family languages and 'MeCab' [28] for Japanese language. Mon, Burmese, Thai and Hakka Chinese languages are different from all the other languages. Considering that their characters are special and are easily mistaken for punctuation, the sentence is split only according to the space.

The final step is to deal with OOV words. All words that appear fewer than once in the training set will be replaced by the unknown word token, `<UNK>`.

Step 2. Train the model
The training step mainly includes three parts: compute the count of the N-gram $C(w_{n-N+1:n-1}w_n)$ and (N-1)-gram $C(w_{n-N+1:n-1})$, implement smoothing to the counts to deal with unseen events, and normalize the final counts to a probability ranging from 0 to 1. Finally, the smoothed probability of each N-gram is achieved.

Step 3. Evaluate the model
The model validation is implemented on the validation set with unseen data. First, in the validation set, the word that does not appear in the training tokens will be replaced by a `<UNK>` token. Instead of only calculating one sentence, the model evaluation calculates several sequences of words $\{w_{s1,1}, w_{s1,2}, \ldots, w_{s1,l1}\}$, $\{w_{s2,1}, w_{s2,2}, \ldots, w_{s2,l2}\}$, $\ldots$, $\{w_{sN,1}, w_{sN,2}, \ldots, w_{sN,lN}\}$ from all sentences in the validation set. The loss for each sentence can thus be calculated according to Eq. 2.6. After normalization, perplexity (also, score) can thus be calculated. For each language, different values of n and different values of Laplace will lead to different models. According to validation results, the model with the lowest perplexity (alternatively, the highest score) means that the model predicts the validation set most correctly and thus is the best model. In addition, for Chinese family languages and Japanese language, the text segmentation method can be changed and thus will lead to different models as well.

3.2.2.2 RoBERTa

Instead of training RoBERTa from scratch, finetuning a pretrained RoBERTa model is applied in this project. The model finetuning and evaluation of a single sentence are based on `Transformers` [29] and `PyTorch` [30] from Python library.

Step 1. Data preprocess
A pretrained RoBERTa tokenizer is loaded using `AutoTokenizer`. Then the `tokenizer` is called on all our training data. Here, `padding` and `truncation` is set as `True` to activate both padding and truncation. Thus, each input sequence can be kept with the same length which is controlled by the parameter `max_length`. Considering that our data set is composed of individual sentences, the step of concatenating all texts and grouping them into small chunks of a fixed block size can be skipped. Next, for the MLM objective, random and dynamic masking needs to be done every time we feed the sequence to the model. Thus, `DataCollatorForLanguageModeling` is applied with the probability 0.15 to randomly mask tokens in the input. Finally, a smaller subset of the full training set

is created by randomly selecting sequences with `train_size` to achieve a balance between training time and model performance.

Step 2. Finetune a pretrained model
First, load a pretrained RoBERTa model with a language modeling head on top by `RobertaForMaskedLM`. Then, define the training hyperparameters in `TrainingArguments`. Next, pass the training arguments, the pretrained model, the training set and validation set and the data collator to `Trainer`. Finally, call `train()` to finetune the model.

Step 3. Evaluate a single sentence
To evaluate the fluency of a single sentence, the pretrained tokenizer used in the model training is loaded again and applied on the sentence. It returns PyTorch `torch.Tensor` objects with `input_ids` to be fed to the model and `attention_masks` specifying which tokens should be attended to by the model. Then we mask the first token in the sentence as `<mask>` token, use the model to predict the originally-masked token, and generate prediction scores for all the tokens in the vocabulary. After `SoftMax` as shown in Eq. 2.10, the probability that each token in the vocabulary can occur at the first position of this sentence is generated. The one that corresponds to the original token in our sentence is selected as $P(w_1|w_2, w_3, \ldots, w_n)$. The masking, prediction and `SoftMax` transformation are iteratively implemented on each token of the sentence. Finally, sentence loss, perplexity and score can be calculated according to Eq. 2.11.

3.2.2.3 Google Ngram Viewer

Google Ngram Viewer [31] is an online platform that provides the proportions of a word or a phrase (length n) from all the n-grams contained in a corpus of books over the selected years.

To evaluate the fluency of a single sentence, first split the sentence into a sequence of words and punctuation. According to the value of n, generate a list of n-grams in sequence. Then make a request to Google Ngram Viewer with a dictionary of `params`, indicating the word or the phrase to search for, the range of selected years, the corpus of books and some other advanced searching options. The probability of each n-gram can thus be achieved. Finally, according to Eq. 2.6, loss and perplexity (also, score) of the sentence can be calculated. If the word or phrase cannot be matched in the corpus, the probability of it will be set as `'NF'` (i.e. not found), loss and perplexity of the whole sentence will be set as `'INF'` (i.e. infinity), and its score will be set as 0.

3.2.3 Combine with Abstract Wikipedia project

To insert the best language model into Abstract Wikipedia project, all the related codes are encapsulated into a Python module. Then to evaluate the quality of each sentence generated from the NLG part of Abstract Wikipedia project, an input file is needed. This file should contain several sets of sentences, sentences in each set with the same meaning but with different structures. After calling the main function in the Python module, the quality of each sentence will be evaluated by the best language model. The output includes two parts. The first part is a report, where sentences in each set are ranked with scores from high to low. The second output is an entire article. The sentence with the highest score (i.e. the lowest perplexity) in each set is chosen and formed into the final article.

3.2.4 Parameters and Hardware

The hyperparameters that can be modified and the hardware used in the project are listed as follows.

	Definition and Values
n	n-gram model. Size of the contiguous word sequence taken into account for predicting the next word, an integer ranging from 1 to 5.
Laplace	n-gram model. Value of k in Add-k smoothing, chosen from {0.0001, 0.0005, 0.001, 0.01, 0.1, 1}, with the value of 1 as Laplace smoothing.
learning_rate	RoBERTa model. The initial learning rate for Adam optimizer, chosen from {1e-07, 2e-07, 5e-07, 1e-06, 2e-06, 5e-06, 2e-05}.
num_train_epochs	RoBERTa model. Number of training epochs to finetune the pretrained model, chosen from {40, 60}.

Table 3.4: Hyperparameters used in language models.

	Parameters
CPU	Intel Core i9 8-Core 2.4 GHz Intel Xeon 2.20GHz Interl Core i5 4-Core 2 GHz
Memory	DDR 4 16 GB DDR 4 26 GB DDR 4X 16GB
GPU	Tesla V100 Tesla A100

Table 3.5: Parameters of hardware.

4

Results

4.1 Data set

Table 4.1 and Table 4.2 present the statistics of data sets organized by languages. The number of sentences and characters are counted for the training set and validation set for each language. For Japanese language and each Chinese family language, there are two vocabulary sizes depending on different text segmentation methods. 'manual' and 'jieba' methods are applied to Chinese family languages while 'manual' and 'MeCab' methods are applied to Japanese language.

Language	**Language Code**	**# Articles**	**# Vocab**	**# Sentences**		**# Characters**	
			train	train	validation	train	validation
English	en	204783	544745	4013160	445906	483188002	53667384
Chinese Simplified	zh-cn	103408	103387/409396	1224408	136045	146046773	16188855
Chinese Traditional	zh-mo	101601	105473/523709	1222470	135830	144947290	16113160
Aragonés	an	8715	9656	6885	765	863088	94605
Belarusian	be	10057	93940	133178	14797	22203423	2496294
Bulgarian	bg	11328	104567	172200	19133	32719349	3614237
Bosanski	bs	4593	62081	61336	6815	7053489:	7839452
Danish	da	6295	64050	95911	10656	10167150	1130685
German	de	60345	485563	1474709	163856	183396694	20363343
Greek	el	5204	80486	104904	11656	24681425	2735840
Spanish	es	38387	228528	771157	85684	106464088	11796715
Finnish	fi	11932	138171	198932	22103	20384538	2264370
French	fr	61564	279673	1178769	130974	158707084	17572173
Irish	ga	4648	23608	28853	3205	3014718	338438
Hakka Chinese	hak	3046	6718	6107	678	691335	77108
Indonesian	id	12167	66791	137231	15247	16275867	1807877
Ilocano	ilo	5086	20304	27534	3059	3267669	354977

Table 4.1: Statistics of the data set grouped by 46 languages. (I)

Language	Language Code	# Articles	# Vocab	# Sentences		# Characters	
			train	train	validation	train	validation
Icelandic	is	4146	40519	50355	5595	5163864	577448
Italian	it	61408	270505	933442	103715	132175169	14696742
Japanese	ja	19056	44011/143507	579465	64385	78823723	8784408
Javanese	jv	4045	27806	33229	3692	3031573	341155
Korean	ko	11528	148710	151907	16878	19234380	2133142
Latin	la	4065	25382	22698	2522	2373568	265328
Mongolian	mn	5171	53787	69387	7709	13856255	1551796
Mon	mnw	1633	69352	48498	5388	28175807	3119773
Malaysian	ms	8715	35388	55578	6175	6645568	736737
Burmese	my	3720	23150	19902	2211	9393003	1048015
Dutch	nl	30504	144593	371814	41312	38215916	4241830
Norwegian	nn	4265	33603	43878	4875	4246098	476739
Polish	pl	61910	319765	752092	83565	83207570	9200464
Portuguese	pt	26866	145458	393540	43726	51491910	5720861
Romanian	ro	9054	77370	104033	11559	13789134	1531430
Russian	ru	100484	685216	2230993	247888	439997426	48892970
Sicilian	scn	3328	14711	13767	1529	1366130	142250
Scottish	sco	3904	24657	28945	3216	2979228	333394
Slovak	sk	5140	58332	63407	7045	6442855	729046
Serbian	sr	10826	115853	139109	15456	25563470	2849432
Swedish	sv	29141	278083	248385	27598	24121695	2665731
Thai	th	3937	27060	46305	5145	19457138	2217840
Filipino	tl	4593	37070	61526	6836	6402553	701149
Turkish	tr	11702	101126	149517	16613	16442861	1840975
Ukrainian	uk	50727	340376	831406	92378	147356703	16300135
Vietnamese	vi	20452	53242	193330	21481	28080002	3125010
Waray	war	11052	14180	23942	2660	1884320	209790
Chinese Classical	zh-classical	3639	6247/35942	44576	4952	2997558	332952
Cantonese	zh-yue	5744	9596/30702	26909	2989	3157775	356319

Table 4.2: Statistics of the data set grouped by 46 languages. (II)
For Japanese language and each Chinese family language, there are two vocabulary sizes depending on different text segmentation methods. Chinese family language: left one is 'manual' and right one is 'jieba'. Japanese family: left one is 'manual' and right one is 'MeCab'.

4.2 Language Model

4.2.1 n-gram

4.2.1.1 Overall

By optimizing on the validation set with unseen data, the best model (i.e. with the highest score and the lowest perplexity) for each language data set has been achieved, along with the corresponding hyperparameter set (including the value of n and Laplace). The results are numerically shown in Table 4.3 and Table 4.4, and are visualized in Fig. 4.1.

Language Code	best perplexity	best score	best n	best Laplace
en	383.367723	0.002608	2	0.0005
zh-cn(manual)	142.304634	0.007027	2	0.0010
zh-cn(jieba)	900.610551	0.001110	2	0.0005
zh-mo(manual)	141.236512	0.007080	2	0.0010
zh-mo(jieba)	742.495532	0.001347	2	0.0005
an	93.180579	0.010732	2	0.0010
be	538.690925	0.001856	2	0.0005
bg	430.249635	0.002324	2	0.0005
bs	562.230404	0.001779	2	0.0010
da	480.599000	0.002081	2	0.0010
de	734.599713	0.001361	2	0.0005
el	454.061340	0.002202	2	0.0005
es	339.988222	0.002941	2	0.0005
fi	912.428032	0.001096	2	0.0005
fr	232.842145	0.004295	2	0.0001
ga	226.915473	0.004407	2	0.0010
hak	95.959386	0.010421	2	0.0010
id	512.774400	0.001950	2	0.0010
ilo	90.757173	0.011018	2	0.0005

Table 4.3: Hyperparameters, score and perplexity of the best n-gram model for each language. For more details, please refer to Appendix. (I)

Language Code	best perplexity	best score	best n	best Laplace
is	388.563807	0.002574	2	0.0010
it	411.386867	0.002431	2	0.0005
ja(manual)	43.081505	0.023212	3	0.0005
ja(MeCab)	176.979142	0.005650	2	0.0005
jv	391.855224	0.002552	2	0.0010
ko	1033.171112	0.000968	2	0.0005
la	254.534439	0.003929	2	0.0010
mn	707.819729	0.001413	2	0.0010
mnw	74.664497	0.013393	2	0.0001
ms	438.712607	0.002279	2	0.0010
my	54.256080	0.018431	2	0.0005
nl	398.842048	0.002507	2	0.0005
nn	325.709432	0.003070	2	0.0010
pl	769.374944	0.001300	2	0.0005
pt	393.155324	0.002544	2	0.0005
ro	449.179759	0.002226	2	0.0005
ru	776.734816	0.001287	2	0.0001
scn	151.218510	0.006613	2	0.0010
sco	288.358690	0.003468	2	0.0010
sk	509.952819	0.001961	2	0.0010
sr	661.573699	0.001512	2	0.0005
sv	250.323337	0.003995	2	0.0001
th	40.606102	0.024627	2	0.0001
tl	141.206348	0.007082	2	0.0005
tr	814.135458	0.001228	2	0.0010
uk	646.050860	0.001548	2	0.0005
vi	159.397096	0.006274	2	0.0010
war	7.414110	0.134878	3	0.0001
zh-classical(manual)	234.079726	0.004272	2	0.0100
zh-classical(jieba)	238.545846	0.004192	2	0.0010
zh-yue(manual)	153.607483	0.006510	2	0.0100
zh-yue(jieba)	345.504906	0.002894	2	0.0010

Table 4.4: Hyperparameters, score and perplexity of the best n-gram model for each language. For more details, please refer to Appendix. (II)

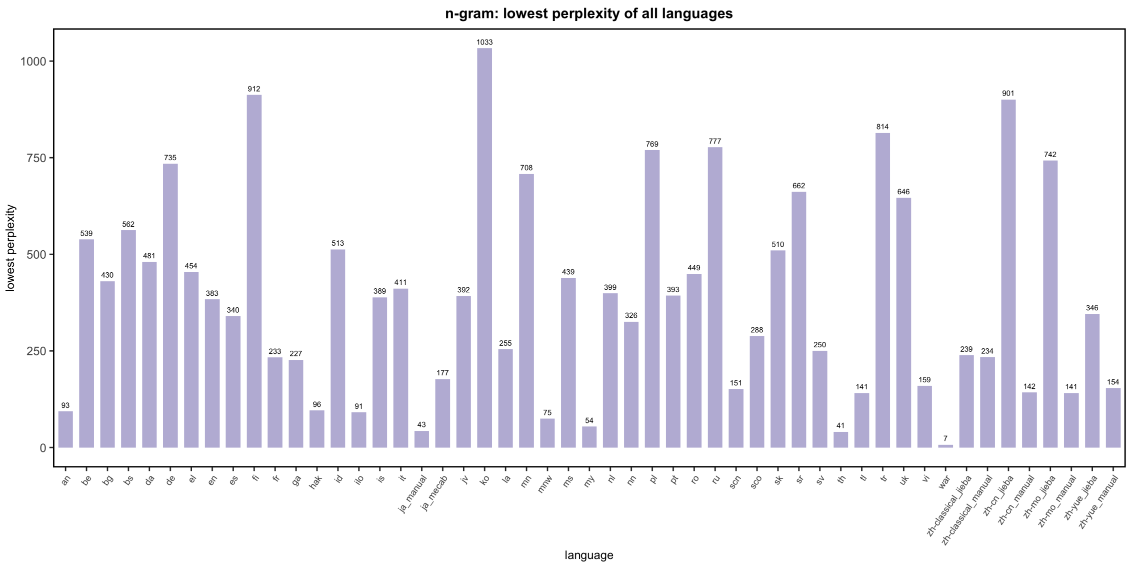
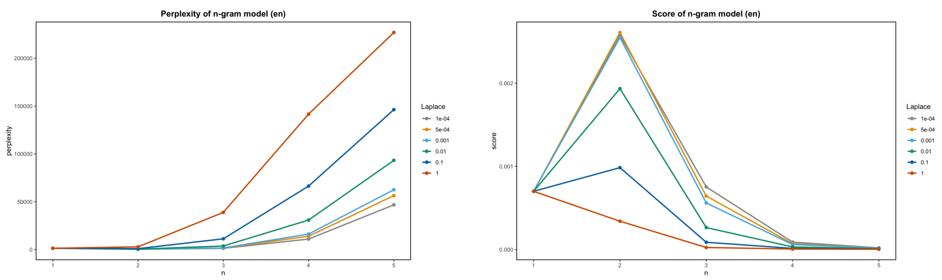


Figure 4.1: Perplexity on the validation set of best n-gram model for each of the 46 languages.

Among the 46 languages, the results of n-gram models of English language, simplified Chinese language, Swedish Language and Japanese language will be shown in detail in the following sections. For the other 42 languages, the detailed results will be presented in Appendix.

4.2.1.2 n-gram for English Language

Table 4.5 and Table 4.6 present model perplexity and model score for different English n-gram models with various values of n and Laplace. Model perplexity and model score are also visualized in Fig. 4.2. Table 4.7 presents the time used for training one English n-gram model with different values of n and Laplace.



(a) Model perplexity **(b)** Model score

Figure 4.2: Model evaluation of English n-gram models on the validation set with unseen data.

Laplace \ n	1	2	3	4	5
0.0001	1426.663686	388.256257	1328.349927	11042.220677	46802.777893
0.0005	1426.664133	383.367723	1553.808416	14012.023682	56466.116169
0.0010	1426.664692	392.552337	1782.857139	16264.727077	62546.633866
0.0100	1426.674819	517.050929	3774.524901	30892.051189	93295.515516
0.1000	1426.782386	1015.426327	11313.754803	66424.648208	146287.284856
1.0000	1428.366392	2930.059708	38928.199359	141601.984866	226907.481921

Table 4.5: Model perplexity of English n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.000701	0.002576	0.000753	0.000091	0.000021
0.0005	0.000701	0.002608	0.000644	0.000071	0.000018
0.0010	0.000701	0.002547	0.000561	0.000061	0.000016
0.0100	0.000701	0.001934	0.000265	0.000032	0.000011
0.1000	0.000701	0.000985	0.000088	0.000015	0.000007
1.0000	0.000700	0.000341	0.000026	0.000007	0.000004

Table 4.6: Model scores of English n-gram models with different n and Laplace values. The higher the score, the better the model.

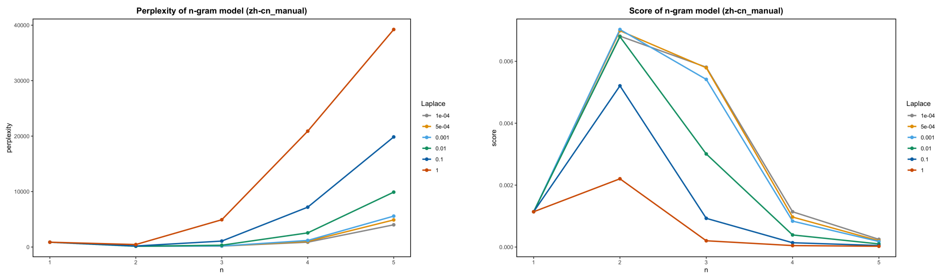
Laplace \ n	1	2	3	4	5
0.0001	413.256694	429.185674	542.682286	739.702677	1230.009797
0.0005	387.321656	433.954818	551.509103	767.614388	1129.381437
0.0010	386.910492	423.289475	531.619424	754.383746	1083.057772
0.0100	397.812255	426.653275	532.868561	746.839408	1138.520161
0.1000	390.678012	437.597651	545.417196	760.290933	1155.943183
1.0000	386.909770	439.600267	551.606176	759.186090	1160.696678

Table 4.7: Training time of English n-gram models with different n and Laplace values.

4.2.1.3 n-gram for Simplified Chinese Language

According to the two text segmentation methods to split the sentence into a sequence of words and punctuation, simplified Chinese n-gram model has two versions: zh-cn_jieba n-gram model and zh-cn_manual n-gram model.

manual text segmentation: Numerical results about training time, model perplexity and model score of simplified Chinese n-gram models with different values of n and Laplace are shown in Table 4.10, Table 4.8 and Table 4.9. Also, these model evaluation results are visualized in Fig. 4.3.



(a) Model perplexity **(b)** Model score

Figure 4.3: Model evaluation of simplified Chinese n-gram models with 'manual' text segmentation method on the validation set with unseen data.

Laplace \ n	1	2	3	4	5
0.0001	876.559474	146.783082	172.151756	876.864172	4009.971329
0.0005	876.559658	143.109786	172.695683	1031.710516	4890.282555
0.0010	876.559889	142.304634	184.678910	1194.229419	5576.480826
0.0100	876.564060	147.189739	332.559675	2556.349995	9901.651343
0.1000	876.607275	192.030093	1077.496331	7192.659206	19852.476919
1.0000	877.160111	453.489024	4926.130272	20896.384487	39224.232309

Table 4.8: Model perplexity of simplified Chinese n-gram models (with 'manual' text segmentation method) with different n and Laplace values. The lower the perplexity, the better the model.

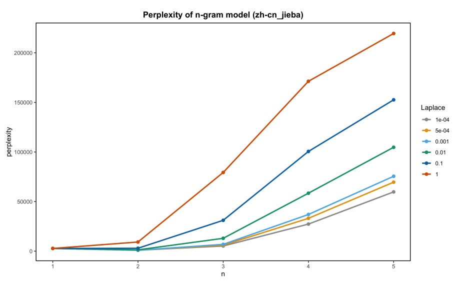
n / Laplace	1	2	3	4	5
0.0001	0.001141	0.006813	0.005809	0.001140	0.000249
0.0005	0.001141	0.006988	0.005791	0.000969	0.000204
0.0010	0.001141	0.007027	0.005415	0.000837	0.000179
0.0100	0.001141	0.006794	0.003007	0.000391	0.000101
0.1000	0.001141	0.005208	0.000928	0.000139	0.000050
1.0000	0.001140	0.002205	0.000203	0.000048	0.000025

Table 4.9: Model scores of simplified Chinese n-gram models (with 'manual' text segmentation method) with different n and Laplace values. The higher the score, the better the model.

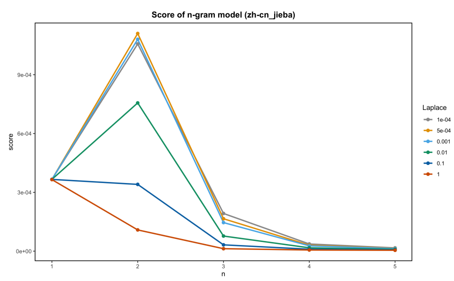
n / Laplace	1	2	3	4	5
0.0001	191.007011	214.515894	260.854823	287.021106	318.566071
0.0005	197.787346	218.298279	260.743847	289.231004	307.179598
0.0010	192.882314	217.571397	261.149233	288.842395	306.757899
0.0100	194.741033	221.278993	256.818773	285.067583	309.274771
0.1000	192.504309	217.566016	257.165019	290.258735	306.424002
1.0000	200.164733	215.833805	262.042322	295.471856	317.688070

Table 4.10: Training time of simplified Chinese n-gram models (with 'manual' text segmentation method) with different n and Laplace values.

jieba text segmentation: Numerical results about training time, model perplexity and model score of simplified Chinese n-gram models with different values of n and Laplace are shown in Table 4.13, Table 4.11 and Table 4.12. Also, these model evaluation results are visualized in Fig. 4.4.



(a) Model perplexity



(b) Model score

Figure 4.4: Model evaluation of simplified Chinese n-gram models with 'jieba' text segmentation method on the validation set with unseen data.

Laplace \ n	1	2	3	4	5
0.0001	2731.217352	945.906752	5200.445766	27296.194149	59743.552735
0.0005	2731.219538	900.610551	6046.467469	33094.538516	69558.326171
0.0010	2731.222272	925.273590	6861.196097	36979.146381	75500.907338
0.0100	2731.271756	1322.461913	12965.197109	58444.500809	104734.866703
0.1000	2731.794126	2935.700304	31099.973309	100535.266890	152576.764451
1.0000	2739.223446	9228.274185	79346.197590	171269.874530	219466.961533

Table 4.11: Model perplexity of simplified Chinese n-gram models (with 'jieba' text segmentation method) with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.000366	0.001057	0.000192	0.000037	0.000017
0.0005	0.000366	0.001110	0.000165	0.000030	0.000014
0.0010	0.000366	0.001081	0.000146	0.000027	0.000013
0.0100	0.000366	0.000756	0.000077	0.000017	0.000010
0.1000	0.000366	0.000341	0.000032	0.000010	0.000007
1.0000	0.000365	0.000108	0.000013	0.000006	0.000005

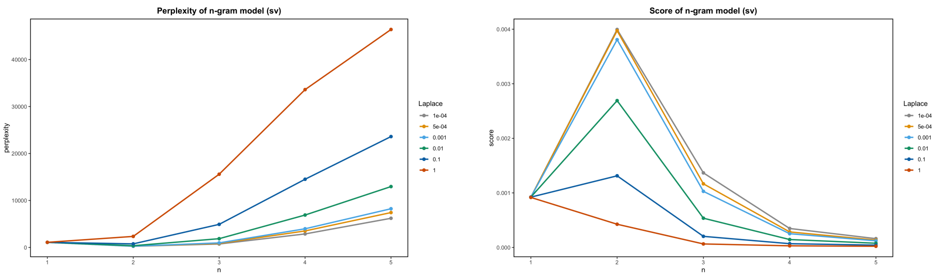
Table 4.12: Model scores of simplified Chinese n-gram models (with 'jieba' text segmentation method) with different n and Laplace values. The higher the score, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	265.799895	283.482426	307.426348	396.464734	344.899579
0.0005	264.807778	283.433598	301.571704	322.990175	343.389842
0.0010	261.600384	396.217262	302.058740	328.375117	345.709002
0.0100	266.302442	286.475218	306.234378	332.210694	343.619654
0.1000	263.821375	284.717204	304.550881	326.384089	343.436067
1.0000	267.220387	287.918683	308.148953	333.914883	345.051493

Table 4.13: Training time of simplified Chinese n-gram models (with 'jieba' text segmentation method) with different n and Laplace values.

4.2.1.4 n-gram for Swedish Language

Table 4.14 and Table 4.15 present model perplexity and model score for different Swedish n-gram models with various values of n and Laplace. Model perplexity and model score are also visualized in Fig. 4.5. Table 4.16 presents the time used for training one Swedish n-gram model with different values of n and Laplace.



(a) Model perplexity **(b)** Model score

Figure 4.5: Model evaluation of Swedish n-gram models on the validation set with unseen data.

Laplace \ n	1	2	3	4	5
0.0001	1083.908679	250.323337	731.676817	2875.762767	6209.409013
0.0005	1083.910660	251.737280	857.552543	3511.963146	7426.370924
0.0010	1083.913138	262.568489	972.610436	3971.851440	8239.080660
0.0100	1083.957910	371.608910	1864.672556	6898.608047	12960.427566
0.1000	1084.423092	761.666379	4915.106564	14518.856625	23605.363077
1.0000	1090.459424	2351.142234	15590.621454	33603.499572	46405.511038

Table 4.14: Model perplexity of Swedish n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.000923	0.003995	0.001367	0.000348	0.000161
0.0005	0.000923	0.003972	0.001166	0.000285	0.000135
0.0010	0.000923	0.003809	0.001028	0.000252	0.000121
0.0100	0.000923	0.002691	0.000536	0.000145	0.000077
0.1000	0.000922	0.001313	0.000203	0.000069	0.000042
1.0000	0.000917	0.000425	0.000064	0.000030	0.000022

Table 4.15: Model scores of Swedish n-gram models with different n and Laplace values. The higher the score, the better the model.

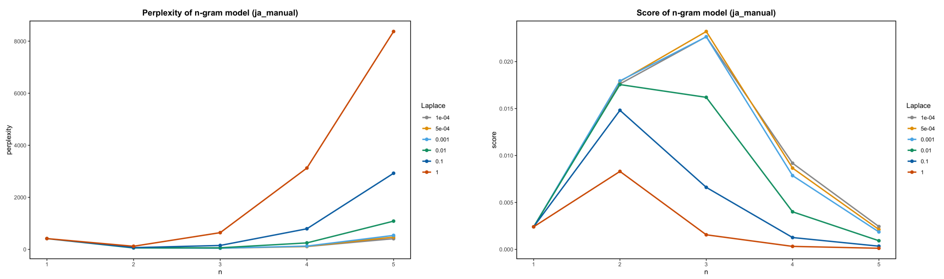
Laplace \ n	1	2	3	4	5
0.0001	49.209463	53.697508	62.660029	68.125721	69.027323
0.0005	48.215874	54.199657	61.444960	66.407604	68.501671
0.0010	49.089781	53.937266	62.433149	66.307748	70.495080
0.0100	49.907539	53.839525	62.525241	65.060424	68.413205
0.1000	49.323816	53.871102	62.327154	65.943273	69.882035
1.0000	48.061592	53.056513	61.374292	65.443740	68.511220

Table 4.16: Training time of Swedish n-gram models with different n and Laplace values.

4.2.1.5 n-gram for Japanese Language

According to the two text segmentation methods to split the sentence into a sequence of words and punctuation, Japanese n-gram model has two versions: ja_manual n-gram model and ja_MeCab n-gram model.

manual text segmentation: Numerical results about training time, model perplexity and model score of Japanese n-gram models with different values of n and Laplace are shown in Table 4.19, Table 4.17 and Table 4.18. Also, these model evaluation results are visualized in Fig. 4.6.



(a) Model perplexity **(b)** Model score

Figure 4.6: Model evaluation of Japanese n-gram models with 'manual' text segmentation method on the validation set with unseen data.

Laplace \ n	1	2	3	4	5
0.0001	415.742347	56.734840	44.139807	108.925635	410.929474
0.0005	415.742402	55.882810	43.081505	115.463553	472.193548
0.0010	415.742471	55.704431	44.161761	127.244471	537.824618
0.0100	415.743721	56.958344	61.736336	249.658225	1085.992645
0.1000	415.756773	67.494626	151.303628	792.125316	2925.536604
1.0000	415.931969	120.377211	643.307052	3125.918466	8372.597461

Table 4.17: Model perplexity of Japanese n-gram models (with 'manual' text segmentation method) with different n and Laplace values. The lower the perplexity, the better the model.

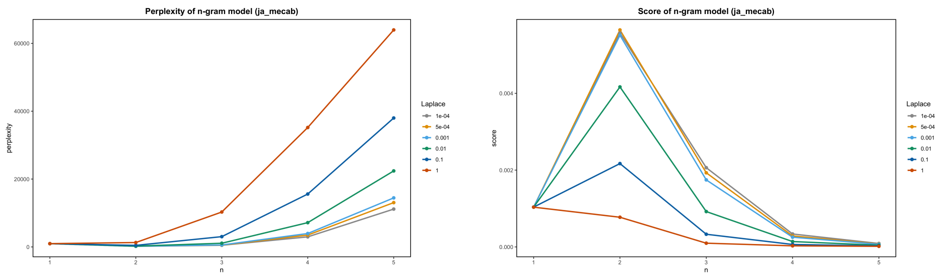
Laplace \ n	1	2	3	4	5
0.0001	0.002405	0.017626	0.022655	0.009181	0.002434
0.0005	0.002405	0.017895	0.023212	0.008661	0.002118
0.0010	0.002405	0.017952	0.022644	0.007859	0.001859
0.0100	0.002405	0.017557	0.016198	0.004005	0.000921
0.1000	0.002405	0.014816	0.006609	0.001262	0.000342
1.0000	0.002404	0.008307	0.001554	0.000320	0.000119

Table 4.18: Model scores of Japanese n-gram models (with 'manual' text segmentation method) with different n and Laplace values. The higher the score, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	128.660767	136.774154	152.654631	175.350526	190.941009
0.0005	133.235604	138.310743	158.023371	178.747799	189.122955
0.0010	127.785426	133.722441	151.934388	172.768345	184.307173
0.0100	124.828156	131.338162	150.085088	169.825827	181.969982
0.1000	129.082832	134.303772	153.360059	173.203857	184.686787
1.0000	127.823501	134.646246	152.285621	172.683118	185.223379

Table 4.19: Training time of Japanese n-gram models (with 'manual' text segmentation method) with different n and Laplace values.

MeCab text segmentation: Numerical results about training time, model perplexity and model score of Japanese n-grams with different values of n and Laplace are shown in Table 4.22, Table 4.20 and Table 4.21. Also, these model evaluation results are visualized in Fig. 4.7.



(a) Model perplexity **(b)** Model score

Figure 4.7: Model evaluation of Japanese n-gram models with 'MeCab' text segmentation method on the validation set with unseen data.

Laplace \ n	1	2	3	4	5
0.0001	964.460049	179.463917	484.615955	2968.875177	11149.875220
0.0005	964.460400	176.979142	518.021595	3490.874556	13062.002445
0.0010	964.460839	181.427672	572.942369	3950.319106	14450.265241
0.0100	964.468811	240.020800	1085.714356	7154.265932	22400.309486
0.1000	964.554821	460.758069	3028.167705	15587.936618	37994.977401
1.0000	965.924130	1293.727750	10292.075658	35193.218401	63951.669768

Table 4.20: Model perplexity of Japanese n-gram models (with 'MeCab' text segmentation method) with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.001037	0.005572	0.002063	0.000337	0.000090
0.0005	0.001037	0.005650	0.001930	0.000286	0.000077
0.0010	0.001037	0.005512	0.001745	0.000253	0.000069
0.0100	0.001037	0.004166	0.000921	0.000140	0.000045
0.1000	0.001037	0.002170	0.000330	0.000064	0.000026
1.0000	0.001035	0.000773	0.000097	0.000028	0.000016

Table 4.21: Model scores of Japanese n-gram models (with 'MeCab' text segmentation method) with different n and Laplace values. The higher the score, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	60.213790	64.755524	79.062338	84.369617	89.114033
0.0005	59.644753	66.827022	78.049657	85.362782	89.803616
0.0010	60.676196	67.074853	78.333647	86.098758	92.184395
0.0100	59.867570	67.427060	78.053961	85.821009	90.829220
0.1000	59.136524	67.349071	78.444734	86.064742	90.840067
1.0000	59.094302	67.182006	78.270227	86.240954	90.811902

Table 4.22: Training time of Japanese n-gram models (with 'MeCab' text segmentation method) with different n and Laplace values.

4.2.1.6 n-gram for other 42 languages

For the numerical results of model perplexity and model score for the other 42 languages, please refer to Appendix.

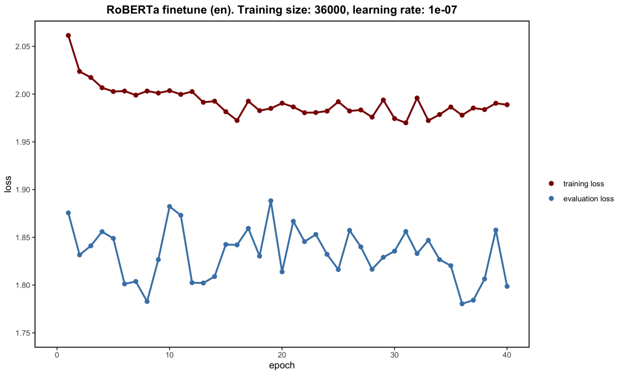
4.2.2 RoBERTa

4.2.2.1 Overall

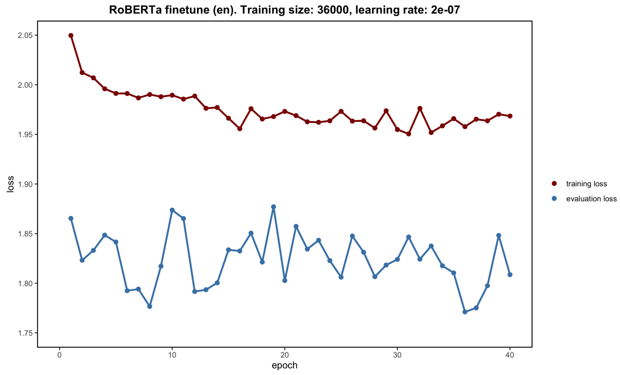
Finetuning a pretrained RoBERTa model with Wikipedia articles has been implemented on English, simplified Chinese, Swedish and Japanese data sets. And the results are shown in the following sections.

4.2.2.2 RoBERTa for English Language

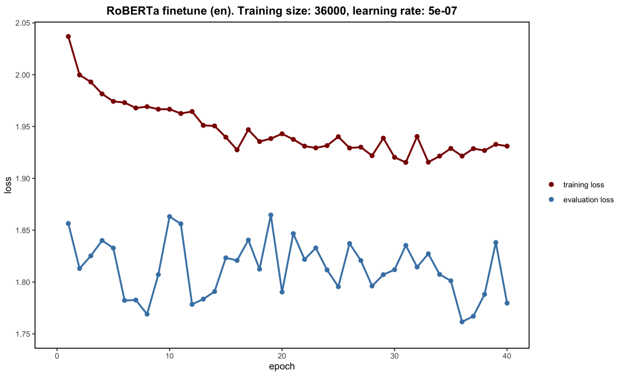
The pretrained RoBERTa model for English is `'roberta-base'` [32]. We use 36,000 training examples from English Wikipedia articles to finetune the model for 40 epochs. The learning rate is selected from {1e-07, 2e-07, 5e-07, 1e-06, 2e-06, 5e-06}. Fig. 4.8 shows training loss and evaluation loss during the finetuning process with different learning rates.



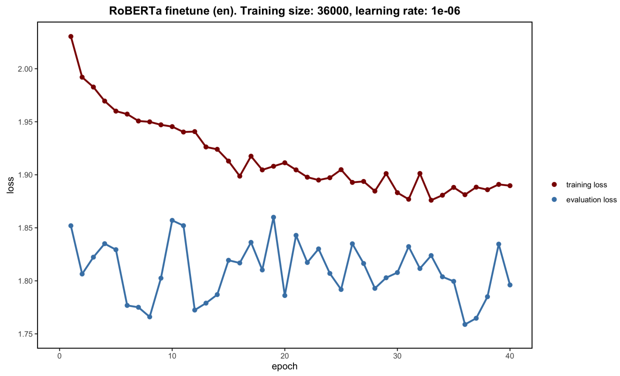
(a)



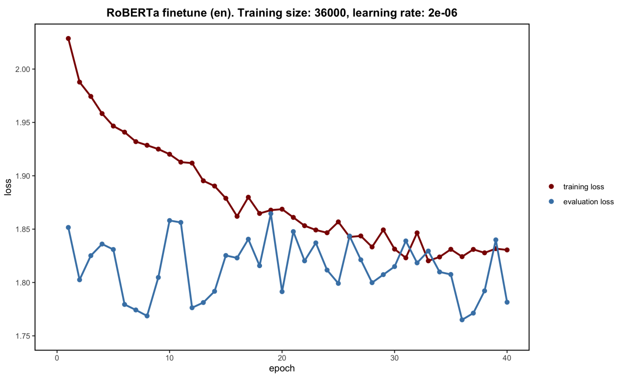
(b)



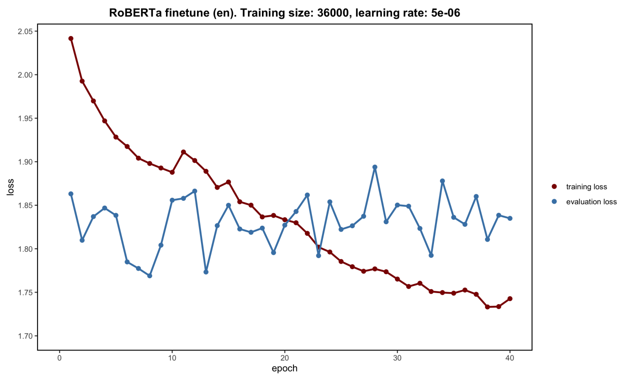
(c)



(d)



(e)

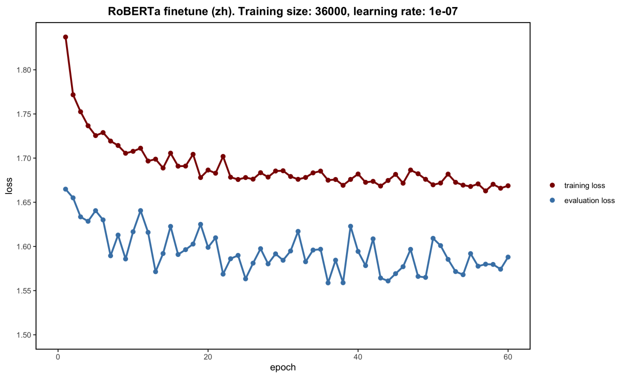


(f)

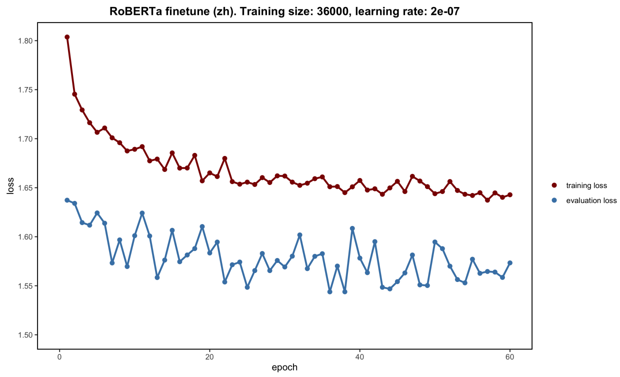
Figure 4.8: Training loss and evaluation loss of RoBERTa finetuning model for English language with 40 epochs. Training size is 36,000 and learning rates are 1e-07, 2e-07, 5e-07, 1e-06, 2e-06 and 5e-06.

4.2.2.3 RoBERTa for Simplified Chinese Language

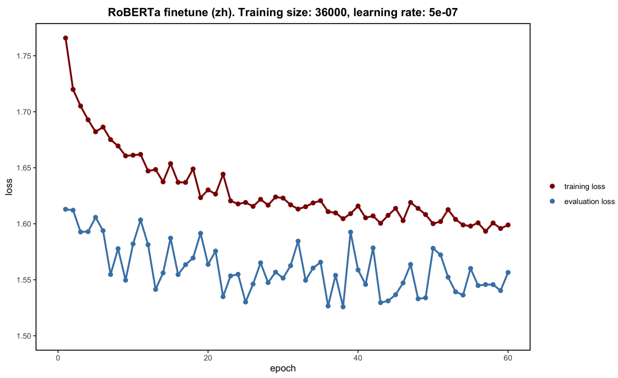
The pretrained RoBERTa model for simplified Chinese is `'hfl/chinese-roberta-wwm-ext'` [33]. We use 36,000 training examples from simplified Chinese Wikipedia articles to finetune the model for 60 epochs. The learning rate is selected from {1e-07, 2e-07, 5e-07, 2e-06}. Fig. 4.9 shows training loss and evaluation loss during the finetuning process with different learning rates.



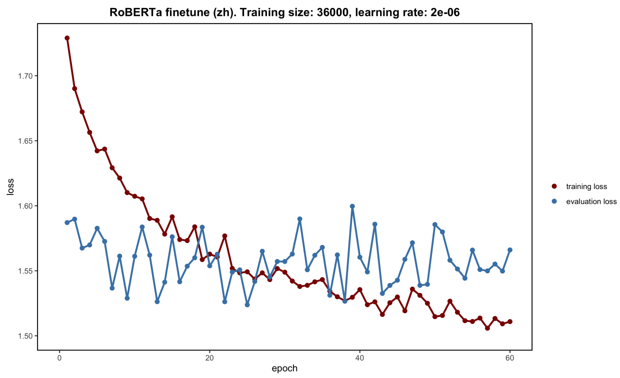
(a)



(b)



(c)

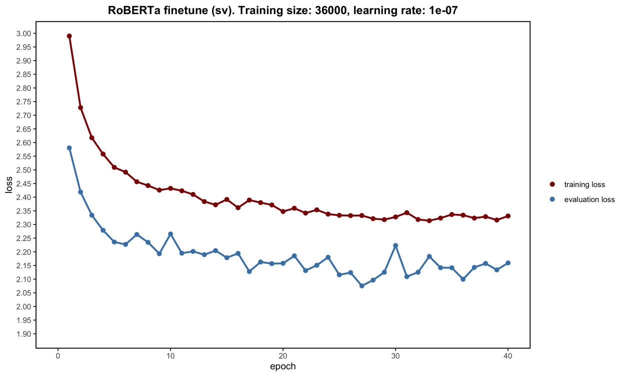


(d)

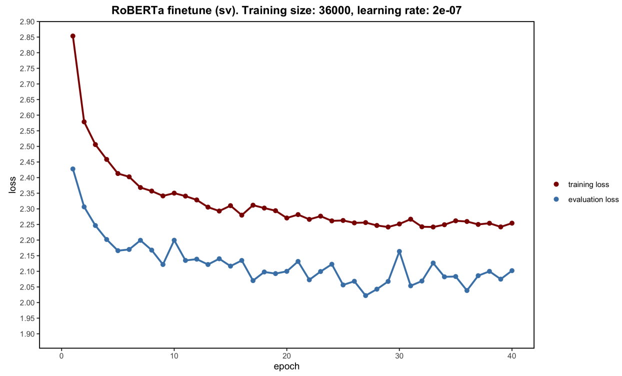
Figure 4.9: Training loss and evaluation loss of RoBERTa finetuning model for simplified Chinese language with 60 epochs. Training size is 36,000 and learning rates are 1e-07, 2e-07, 5e-07 and 2e-06.

4.2.2.4 RoBERTa for Swedish Language

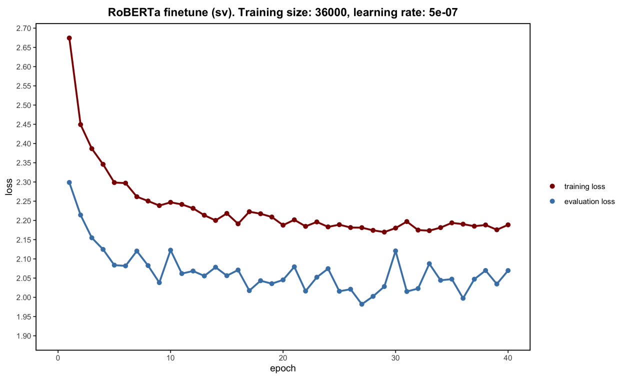
The pretrained RoBERTa model for Swedish is `'birgermoell/roberta-swedish'` [34]. We use 36,000 training examples from Swedish Wikipedia articles to finetune the model for 40 epochs. The learning rate is selected from {1e-07, 2e-07, 5e-07, 2e-06}. Fig. 4.10 shows training loss and evaluation loss during the finetuning process with different learning rates.



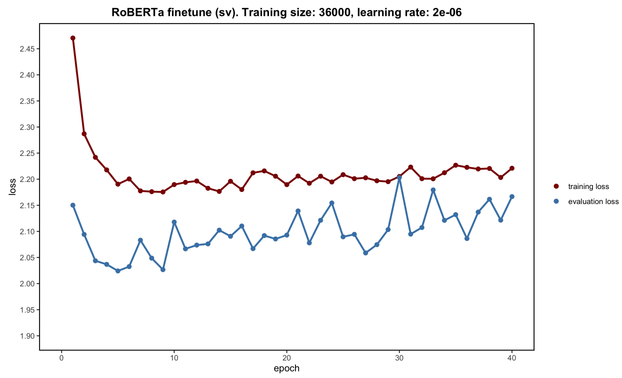
(a)



(b)



(c)

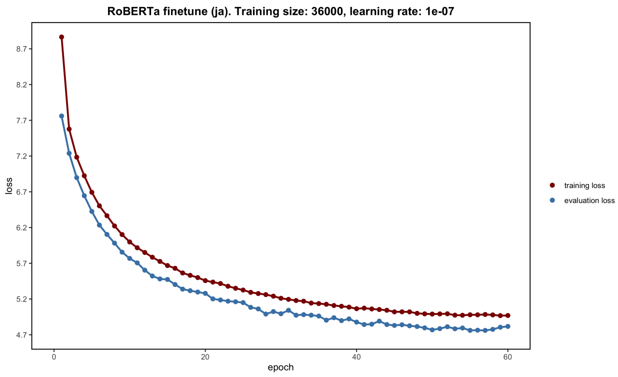


(d)

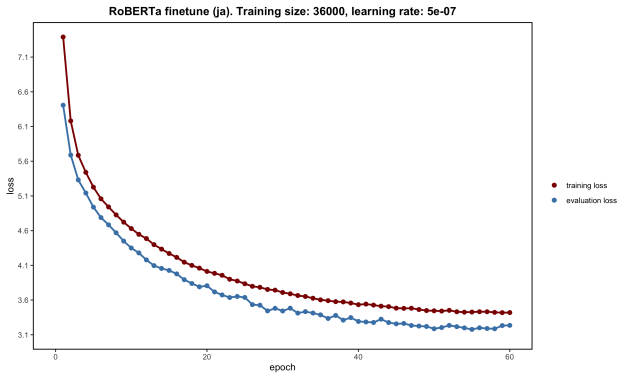
Figure 4.10: Training loss and evaluation loss of RoBERTa finetuning model for Swedish language. Training size is 36,000 and learning rates are 1e-07, 2e-07, 5e-07 and 2e-06.

4.2.2.5 RoBERTa for Japanese Language

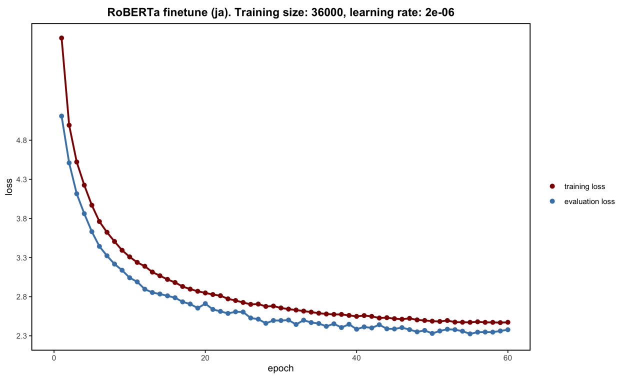
The pretrained RoBERTa model for Japanese is `'nlp-waseda/roberta-base-japanese'` [35]. We use 36,000 training examples from Japanese Wikipedia articles to finetune the model for 60 epochs. The learning rate is selected from {1e-07, 5e-07, 2e-06, 2e-05}. Fig. 4.11 shows training loss and evaluation loss during the finetuning process with different learning rates.



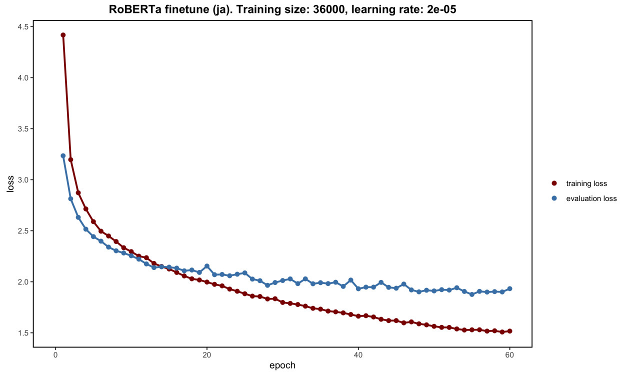
(a)



(b)



(c)



(d)

Figure 4.11: Training loss and evaluation loss of RoBERTa finetuning model for Japanese language. Training size is 36,000 and learning rates are 1e-07, 5e-07, 2e-06 and 2e-05.

4.3 Applied in Abstract Wikipedia project

4.3.1 Overall

Finally, combine the language model with NLG part of Abstract Wikipedia project. The input is a text document with several sets of sentences. Sentences in each set are the candidates convey the same information but have different sentence structures. Some of them may have grammatical mistakes. The outputs are two text documents named as 'report' and 'article', respectively. In the report, sentences in each set are ranking according to the score from high to low. The probability of each token in the sentence can also be included in the report. In the article, the candidate sentence with the highest score in each set will be extracted and forms into a paragraph or an article. The example of an input file is shown in Listing 1. The example of the corresponding output report and the output article are shown in Listing 2 and Listing 3.

```
3
It is famous that Marie Curie discovered Radium.
Marie Curie is best known for discovering Radium.
Marie Curie is best known at discovering Radium.
3
Marie Curie took her daughters on visits to Poland.
She took her daughters on visits to Poland.
Her daughters were took to Poland on visits by her.
2
In 1906 Pierre Curie died in a Paris street accident.
Pierre Curie died because a Paris street accident in 1906.
```

Listing 1: An example input file with several sets of sentences, sentences in each set with the same meaning but different structures.

```
File: test.txt
Model: n-gram model
Num of Sentences: 3
1
[1  -  1]: Marie Curie is best known for discovering Radium.
score = 0.003513, loss = 56.512032, perplexity = 284.633734
[1  -  2]: Marie Curie is best known at discovering Radium.
score = 0.002026, loss = 62.015347, perplexity = 493.505848
[1  -  3]: It is famous that Marie Curie discovered Radium.
score = 0.001048, loss = 68.609279, perplexity = 954.252115
2
[2  -  1]: She took her daughters on visits to Poland.
score = 0.006532, loss = 50.310231, perplexity = 153.089561
[2  -  2]: Marie Curie took her daughters on visits to Poland.
score = 0.004311, loss = 59.913088, perplexity = 231.978421
[2  -  3]: Her daughters were took to Poland on visits by her.
score = 0.001406, loss = 78.802267, perplexity = 711.130214
3
[3  -  1]: In 1906 Pierre Curie died in a Paris street accident.
score = 0.005912, loss = 61.568697, perplexity = 169.139657
[3  -  2]: Pierre Curie died because a Paris street accident in
 ↪  1906.
score = 0.002735, loss = 70.821164, perplexity = 365.681829
```

Listing 2: One of the output files corresponding to the previous input file: a report with sentences in each set ranking with scores from high to low.

```
Marie Curie is best known for discovering Radium. She took her
 ↪  daughters on visits to Poland. In 1906 Pierre Curie died in a
 ↪  Paris street accident.
```

Listing 3: One of the output files corresponding to the previous input file: an article formed into by the sentence with the highest score and the lowest perplexity in each set.

The above-mentioned language models (i.e. n-gram model, Google Ngram Viewer, pretrained RoBERTa model, and RoBERTa finetuning model) for different languages (i.e. English, simplified Chinese, Swedish and Japanese) are evaluated and the results are shown in the following sections.

4.3.2 English

Table 4.23, Table 4.24 and Table 4.25 present sentence loss, perplexity and score (after normalization) of each sentence evaluated by different models with their best hyperparameters. The ranking of candidates in each sentence set is also provided. The sentence ranking 1st will be selected and formed into the final article.

n-gram model is trained with n = 2 and Laplace = 0.0005; Google Ngram model is searched on Google Ngram Viewer platform with n = 2, contents in 'eng_2019' corpus starting from 1949 to 2019; RoBERTa base model corresponds to the pretrained model `'roberta-base'`; RoBERTa finetuning model is finetuned from the above-mentioned pretrained RoBERTa model with learning_rate = 5e-06, training_size = 36,000 for 40 epochs.

		It is famous that Marie Curie discovered Radium.	**Marie Curie is best known for discovering Radium.**	**Marie Curie is best known at discovering Radium.**
Manual ranking		2nd	1st	3rd
n-gram	loss	68.609279	56.512032	62.015347
	ppl	954.252115	284.633734	493.505848
	score	0.001048	0.003513	0.002026
	ranking	3rd	1st	2nd
Google Ngram	loss	131.660881	INF	INF
	ppl	14042656.654888	INF	INF
	score	0.000000	0.000000	0.000000
	ranking	1st	-	-
RoBERTa base	loss	17.803611	17.159906	34.543549
	ppl	5.045567	4.758781	23.111322
	score	0.198194	0.210138	0.043269
	ranking	2nd	1st	3rd
RoBERTa finetune	loss	22.214543	16.043951	35.261480
	ppl	7.534586	4.299682	24.670028
	score	0.132721	0.232575	0.040535
	ranking	2nd	1st	3rd

Table 4.23: Evaluation of three candidates from the first set with the same meaning but different sentence structures by different English language models. The sentence ranking 1st is the best one.

		Marie Curie took her daughters on visits to Poland.	She took her daughters on visits to Poland.	Her daughters were took to Poland on visits by her.
Manual ranking		2nd	1st	3rd
n-gram	loss	59.913088	50.310231	78.802267
	ppl	231.978421	153.089561	711.130214
	score	0.004311	0.006532	0.001406
	ranking	2nd	1st	3rd
Google Ngram	loss	136.841885	112.092330	146.426886
	ppl	4011411.997498	1216564.302507	2286856.201565
	score	0.000000	0.000001	0.000000
	ranking	3rd	1st	2nd
RoBERTa base	loss	15.780193	14.019427	37.246408
	ppl	4.197810	4.747955	29.548621
	score	0.238219	0.210617	0.033843
	ranking	1st	2nd	3rd
RoBERTa finetune	loss	13.764801	13.334520	33.991279
	ppl	3.495042	4.400039	21.979633
	score	0.286120	0.227271	0.045497
	ranking	1st	2nd	3rd

Table 4.24: Evaluation of three candidates from the second set with the same meaning but different sentence structures by different English language models. The sentence ranking 1st is the best one.

		In 1906 Pierre Curie died in a Paris street accident.	Pierre Curie died because a Paris street accident in 1906.
Manual ranking		1st	2nd
n-gram	loss	61.568697	70.821164
	ppl	169.139657	365.681829
	score	0.005912	0.002735
	ranking	1st	2nd
Google Ngram	loss	157.027969	157.635068
	ppl	6601430.575107	7014617.656136
	score	0.000000	0.000000
	ranking	1st	2nd
RoBERTa base	loss	22.810793	46.727163
	ppl	6.691910	49.103499
	score	0.149434	0.020365
	ranking	1st	2nd
RoBERTa finetune	loss	23.302984	41.373760
	ppl	6.972092	31.431587
	score	0.143429	0.031815
	ranking	1st	2nd

Table 4.25: Evaluation of two candidates from the third set with the same meaning but different sentence structures by different English language models. The sentence ranking 1st is the best one.

4.3.3 Simplified Chinese

Table 4.26, Table 4.27 and Table 4.28 present sentence loss, perplexity and score (after normalization) of each sentence evaluated by different models with their best hyperparameters. The ranking of candidates in each sentence set is also provided. The sentence ranking 1st will be selected and formed into the final article. There are two more models than English section because according to the text segmentation method (i.e. 'manual' and 'jieba'), n-gram model, as well as Google Ngram model, may generate different results.

n-gram model with 'manual' text segmentation method is trained with n = 2 and Laplace = 0.001; n-gram model with 'jieba' text segmentation method is trained with n = 2 and Laplace = 0.0005; Google Ngram model is searched on Google Ngram Viewer platform with n = 2, contents in 'chi_sim_2019' corpus starting from 1949 to 2019 with two different text segmentation methods; RoBERTa base model corresponds to the pretrained model `'hfl/chinese-roberta-wwm-ext'`; RoBERTa finetuning model is finetuned from the above-mentioned pretrained RoBERTa model with learning_rate = 2e-06, training_size = 36,000 for 60 epochs.

		1934年，玛丽病逝于法国疗养院，享年66岁。	1934年，66岁的玛丽因为生病在法国疗养院去世。
Manual ranking		1st	2nd
n-gram (manual)	loss	87.285338	99.569014
	ppl	78.591762	92.375728
	score	0.012724	0.010825
	ranking	1st	2nd
n-gram (jieba)	loss	81.075005	109.299566
	ppl	327.363698	1460.651322
	score	0.003055	0.000685
	ranking	1st	2nd
Google Ngram (manual)	loss	INF	INF
	ppl	INF	INF
	score	0.000000	0.000000
	ranking	-	-
Google Ngram (jieba)	loss	INF	INF
	ppl	INF	INF
	score	0.000000	0.000000
	ranking	-	-
RoBERTa base	loss	64.848522	65.192983
	ppl	30.358609	22.296457
	score	0.032940	0.044850
	ranking	2nd	1st
RoBERTa finetune	loss	31.762017	38.340632
	ppl	5.321127	6.207414
	score	0.187930	0.161098
	ranking	1st	2nd

Table 4.26: Evaluation of two candidates from the first set with the same meaning but different sentence structures by different simplified Chinese language models. The sentence ranking 1st is the best one.

		她教女儿波兰文，多次带她们去波兰。	她教女儿波兰文，多次带女儿去波兰。	居里的女儿和妈妈学波兰文，经常和妈妈去波兰。
Manual ranking		1st	2nd	3rd
n-gram (manual)	loss	92.219067	93.660379	132.878853
	ppl	167.885381	181.881322	322.899603
	score	0.005956	0.005498	0.003097
	ranking	1st	2nd	3rd
n-gram (jieba)	loss	100.292606	101.596436	138.089174
	ppl	4262.952080	4752.231970	9956.142323
	score	0.000235	0.000210	0.000100
	ranking	1st	2nd	3rd
Google Ngram (manual)	loss	INF	INF	INF
	ppl	INF	INF	INF
	score	0.000000	0.000000	0.000000
	ranking	-	-	-
Google Ngram (jieba)	loss	INF	INF	INF
	ppl	INF	INF	INF
	score	0.000000	0.000000	0.000000
	ranking	-	-	-
RoBERTa base	loss	73.489355	62.290350	83.630952
	ppl	75.407237	39.022495	44.764119
	score	0.013261	0.025626	0.022339
	ranking	3rd	1st	2nd
RoBERTa finetune	loss	37.507127	33.538051	37.079328
	ppl	9.082065	7.190973	5.394738
	score	0.110107	0.139063	0.185366
	ranking	3rd	2nd	1st

Table 4.27: Evaluation of three candidates from the second set with the same meaning but different sentence structures by different simplified Chinese language models. The sentence ranking 1st is the best one.

		玛丽亚的父亲是无神论者。	玛丽亚的父亲不相信神明的存在。
Manual ranking		1st	2nd
n-gram (manual)	loss	48.304659	65.271499
	ppl	41.089104	59.114055
	score	0.024337	0.016916
	ranking	1st	2nd
n-gram (jieba)	loss	35.082732	51.884333
	ppl	150.177634	179.187599
	score	0.006659	0.005581
	ranking	1st	2nd
Google Ngram (manual)	loss	147.850230	192.424337
	ppl	687577.619409	931541.023595
	score	0.000001	0.000001
	ranking	1st	2nd
Google Ngram (jieba)	loss	76.810635	INF
	ppl	4695555.519230	INF
	score	0.000000	0.000000
	ranking	1st	2nd
RoBERTa base	loss	82.484620	64.063388
	ppl	966.535777	71.585499
	score	0.001035	0.013969
	ranking	2nd	1st
RoBERTa finetune	loss	15.394852	20.859242
	ppl	3.607101	4.017324
	score	0.277231	0.248922
	ranking	1st	2nd

Table 4.28: Evaluation of two candidates from the third set with the same meaning but different sentence structures by different simplified Chinese language models. The sentence ranking 1st is the best one.

4.3.4 Swedish

Table 4.29, Table 4.30 and Table 4.31 present sentence loss, perplexity and score (after normalization) of each sentence evaluated by different models with their best hyperparameters. The ranking of candidates in each sentence set is also provided. The sentence ranking 1st will be selected and formed into the final article. There is one less model than English section because Google Ngram Viewer does not provide Swedish corpora [31].

n-gram model is trained with n = 2 and Laplace = 0.0001; RoBERTa base model corresponds to the pretrained model `‘birgermoell/roberta-swedish’`; RoBERTa finetuning model is finetuned from the above-mentioned pretrained RoBERTa model with learning_rate = 5e-07, training_size = 36,000 for 40 epochs.

		Marie Curie är mest känd för att hon upptäckte Radium.	**Marie Curie blev berömd efter hon hade upptäckt Radium.**	**Marie Curie är bäst känd för upptäcka Radium.**
Manual ranking		1st	2nd	3rd
n-gram	loss	67.056621	74.767268	68.753006
	ppl	267.214504	895.179588	968.066356
	score	0.003742	0.001117	0.001033
	ranking	1st	2nd	3rd
RoBERTa base	loss	19.142519	39.409143	39.255937
	ppl	5.698648	35.968709	78.395833
	score	0.175480	0.027802	0.012756
	ranking	1st	2nd	3rd
RoBERTa finetune	loss	20.946149	28.954442	38.957004
	ppl	6.714003	13.904632	75.834701
	score	0.148942	0.071918	0.013187
	ranking	1st	2nd	3rd

Table 4.29: Evaluation of three candidates from the first set with the same meaning but different sentence structures by different Swedish language models. The sentence ranking 1st is the best one.

		Marie Curie tog med sina döttrar när hon besökte Polen.	Marie Curie reste till Polen med sin dotter.	Marie Curie tog sin döttrar till besök till Polen.
Manual ranking		1st	2nd	3rd
n-gram	loss	75.429583	54.217873	77.059032
	ppl	536.892895	226.283196	1102.534063
	score	0.001863	0.004419	0.000907
	ranking	2nd	1st	3rd
RoBERTa base	loss	23.251199	18.146692	57.257677
	ppl	10.227906	9.663312	579.383233
	score	0.097772	0.103484	0.001726
	ranking	2nd	1st	3rd
RoBERTa finetune	loss	23.725008	16.417966	52.206184
	ppl	10.724178	7.785366	330.526606
	score	0.093247	0.128446	0.003025
	ranking	2nd	1st	3rd

Table 4.30: Evaluation of three candidates from the second set with the same meaning but different sentence structures by different Swedish language models. The sentence ranking 1st is the best one.

		År 1906 dog Pierre Curie I en olycka i Paris.	Pierre Curie dog i en olycka i Paris år 1906.	I 1906 Pierre Curie dog i en Paris gata olycka.
Manual ranking		1st	2nd	3rd
n-gram	loss	84.634493	65.294691	109.524231
	ppl	1156.177278	230.724484	9200.556260
	score	0.000865	0.004334	0.000109
	ranking	2nd	1st	3rd
RoBERTa base	loss	35.696190	17.474551	67.116705
	ppl	35.503063	4.896867	821.942536
	score	0.028167	0.204212	0.001217
	ranking	2nd	1st	3rd
RoBERTa finetune	loss	35.475657	19.042420	66.765370
	ppl	34.728673	5.647026	793.566201
	score	0.028795	0.177084	0.001260
	ranking	2nd	1st	3rd

Table 4.31: Evaluation of three candidates from the third set with the same meaning but different sentence structures by different Swedish language models. The sentence ranking 1st is the best one.

4.3.5 Japanese

Table 4.32, Table 4.33 and Table 4.34 present sentence loss, perplexity and score (after normalization) of each sentence evaluated by different models. The ranking of candidates in each sentence set is also provided. The sentence ranking 1st will be selected and formed into the final article. The models are slightly different from those in English section because according to the text segmentation method (i.e. 'manual' and 'MeCab'), n-gram model may generate different results. Besides, Google Ngram Viewer does not provide Japanese corpora [31].

n-gram model with 'manual' text segmentation method is trained with n = 3 and Laplace = 0.0005; n-gram model with 'MeCab' text segmentation method is trained with n = 2 and Laplace = 0.0005; RoBERTa base model corresponds to the pretrained model `'nlp-waseda/roberta-base-japanese'`; RoBERTa finetuning model is finetuned from the above-mentioned pretrained RoBERTa model with learning_rate = 2e-05, training_size = 36,000 for 60 epochs.

		キュリー夫人は娘をポーランドに連れて行った。	娘はキュリー夫人にポーランドに連れられて行った。
Manual ranking		1st	2nd
n-gram (manual)	loss	72.299517	89.629561
	ppl	23.183881	36.059936
	score	0.043133	0.027732
	ranking	1st	2nd
n-gram (MeCab)	loss	60.197401	73.236920
	ppl	102.572444	187.018687
	score	0.009749	0.005347
	ranking	1st	2nd
RoBERTa base	loss	187.664115	205.680822
	ppl	271289.287624	179669.058406
	score	0.000004	0.000006
	ranking	2nd	1st
RoBERTa finetune	loss	29.159436	42.907953
	ppl	6.986379	12.478376
	score	0.143136	0.080139
	ranking	1st	2nd

Table 4.32: Evaluation of two candidates from the first set with the same meaning but different sentence structures by different Japanese language models. The sentence ranking 1st is the best one.

		キュリー夫人はラジウムを発見して有名だ。	ラジウムを発見したのでキュリー夫人は有名になった。
Manual ranking		1st	2nd
n-gram (manual)	loss	63.591689	76.562735
	ppl	20.659508	19.005351
	score	0.048404	0.052617
	ranking	2nd	1st
n-gram (MeCab)	loss	53.773062	76.151322
	ppl	88.330766	116.682623
	score	0.011321	0.008570
	ranking	1st	2nd
RoBERTa base	loss	193.659349	224.678426
	ppl	404587.488986	136650.293255
	score	0.000002	0.000007
	ranking	2nd	1st
RoBERTa finetune	loss	43.830879	42.808124
	ppl	18.579496	9.516805
	score	0.053823	0.105077
	ranking	2nd	1st

Table 4.33: Evaluation of two candidates from the second set with the same meaning but different sentence structures by different Japanese language models. The sentence ranking 1st is the best one.

		1906年、ピエール・キュリーはパリの街頭事故で死んだ。	パリの街頭事故でピエール・キュリーは1906年に死んだ。
Manual ranking		1st	2nd
	loss	88.641112	109.246122
n-gram	ppl	30.243266	66.804732
(manual)	score	0.033065	0.014969
	ranking	1st	2nd
	loss	79.572547	84.316976
n-gram	ppl	144.500675	194.379224
(MeCab)	score	0.006920	0.005145
	ranking	1st	2nd
	loss	240.953199	246.163606
RoBERTa	ppl	170699.493154	221500.539779
base	score	0.000006	0.000005
	ranking	1st	2nd
	loss	36.362916	42.521083
RoBERTa	ppl	6.160425	8.381728
finetune	score	0.162326	0.119307
	ranking	1st	2nd

Table 4.34: Evaluation of two candidates from the third set with the same meaning but different sentence structures by different Japanese language models. The sentence ranking 1st is the best one.

5 Conclusion

5.1 Discussion

5.1.1 Exploratory Data Analysis

The entire data set contains approximately 1 billion characters and 7 million sentences from 1,173,914 Wikipedia articles. For each language, sentences in the data set are randomly shuffled and then split into two subsets (i.e. 90% for the training set and 10% for the validation set). Table 4.1 and Table 4.2 present the statistics of data set. According to the two different text segmentation methods applied to Chinese family languages and Japanese language, two sets of statistics exist in the training set. Thus, the total number of vocabulary size is roughly 5 million by using 'manual' method while roughly 6 million by using 'jieba' and 'MeCab' respectively for those languages. Also, it is demonstrated that the language with the maximum number of Wikipedia articles is English, while the smallest data set is generated from Mon Wikipedia articles. The size of English data set is about 125 times larger than that of Mon data set.

5.1.2 n-gram

According to Table 4.3 and Table 4.4, with different data sets, the choices of n are different. However, in most cases, the bigram model (i.e. $n = 2$) always shows the lowest perplexity. When n is larger than 2, with the increasing of n, computation complexity and model training time keep increasing, but model performance decreases.

Add-k smoothing is useful to avoid zero probability and has a better effect than Laplace smoothing. With different data sets, the choices of the Laplace parameter vary.

For Chinese family languages and Japanese language, the n-gram model with the 'manual' text segementation method always performs better than that with an advanced method (i.e. 'jieba' and 'MeCab'). The reason is that according to different text segmentation methods, the vocabulary sizes are different. The advanced method always results in a larger vocabulary size and the matching is more difficult, leading to a higher perplexity on the validation set. Thus, only across language models with the same vocabulary, should model perplexity be compared and should intrinsic evaluation be meaningful.

5.1.3 RoBERTa

Fig. 4.8, Fig. 4.9, Fig. 4.10 and Fig. 4.11 illustrate that when epoch increases, training loss goes down and converges after some epochs. However, the change of validation loss performs differently with different language data sets. With the Swedish data set and the pretrained Swedish RoBERTa model, validation loss keeps decreasing and finally converges, which means that there is still room for the pretrained Swedish RoBERTa model to progress. The pattern of loss during the process of finetuning a pretrained Japanese RoBERTa model is similar to that of Swedish, indicating that the base model of Japanese language is not fully trained. Trained for the same epochs, the model learns more and performs better with a larger learning rate.

However, in the cases of English and Chinese language models, validation loss does not decrease but fluctuates at a certain level, which indicates that the pretrained model is trained thoroughly and our current finetuning techniques will not improve the model performance on evaluating wiki-style sentences apparently.

Finally, the best RoBERTa finetune models of above-mentioned languages are concluded as follows:

- The best English model is finetuned from the pretrained '`roberta-base`' model with learning_rate = 5e-06, training_size = 36,000 for 40 epochs.
- The best simplified Chinese model is finetuned from the pretrained '`hfl/chinese-roberta-wwm-ext`' model with learning_rate = 2e-06, training_size = 36,000 for 60 epochs.
- The best Swedish model is finetuned from the pretrained '`birgermoell/roberta-swedish`' model with learning_rate = 5e-07, training_size = 36,000 for 40 epochs.
- The best Japanese model is finetuned from the pretrained '`nlp-waseda/roberta-base-japanese`' model with learning_rate = 2e-05, training_size = 36,000 for 60 epochs.

5.1.4 Applied in Abstract Wikipedia project

According to Section 4.3, the comparison between manual evaluation results and results generated by the language models shows that language model is a reasonable and useful technique to evaluate sentence fluency. Different language models perform slightly differently in this task:

1. **Google n-gram model** is not a good choice because it does not provide a smoothing or discounting technique to avoid the zero probability caused by unknown words and unseen contexts, and thus may lead to infinite loss and infinite perplexity.

2. In most cases, **n-gram model** trained from Wikipedia articles can distinguish the first place from several sentence candidates, showing potential in this task.

3. **RoBERTa base model** can always find out the sentence with grammatical mistakes and rank it as the last one. However, when dealing with two grammatically-correct sentences with the same meaning, it may generate a different result from the manual evaluation result. In particular, RoBERTa base model always prefers the use of a content word to the use of a pronoun.

4. English **RoBERTa finetuning model** performs similarly to its base one and it is in accordance with the unchanged validation loss shown in Fig. 4.8. Swedish RoBERTa finetuning model performs similarly to its base one and both of them can distinguish the grammatically-wrong sentence. But when dealing with two grammatically-correct sentences, it prefers to give a declarative sentence a higher score, which sometimes is slightly different from the manual evaluation criteria. However, Chinese RoBERTa finetuning model and Japanese RoBERTa finetuning model perform slightly differently from their base models. The ranking generated from the finetuning model is more consistent with the manual ranking, indicating that finetuning Chinese and Japanese RoBERTa base models with only Wikipedia articles can improve the ability to evaluate the fluency of a single sentence.

5.2 Future Work

1. For **data set**, it could be enlarged by the functionality of `auto_crawler`. Although the current data set contains 46 languages, data sets with Wikipedia articles from different languages could be explored.

2. For **language models**, RoBERTa models of other languages could be continued to finetune.

3. Pretrained RoBERTa models may not be provided for some languages and thus they should be trained from scratch. The ability to evaluate the fluency of a single sentence should be compared among different language models for other languages as well.

4. Other kinds of language models (e.g. GPT-3, DistilGPT-2, etc.) could be explored on their ability to evaluate sentence fluency.

5.3 Conclusion

The main purpose of this project is to investigate the ability of language models of the evaluation and selection of auto-generated Abstract Wikipedia sentences and to combine the language model part with the NLG part of Abstract Wikipedia project to improve the quality of auto-generated articles.

For that reason, we built multilingual data sets with 46 languages by crawling, processing and tokenizing Wikipedia articles. Based on these data sets, we conducted research on two language models: n-gram model and RoBERTa model. We trained n-gram models for each language and found the best choice of n and Laplace by optimizing on a validation set. We finetuned four RoBERTa models (i.e. English, simplified Chinese, Swedish and Japanese) and found the best choice of learning rate according to training loss and validation loss. Extrinsic evaluation is applied to four language data sets (i.e. English, simplified Chinese, Swedish and Japanese) by evaluating the quality of a set of sentence candidates that convey the same semantic contents but are with different structures and phraseology. The evaluation results generated from n-gram model, Google Ngram model, RoBERTa base model and RoBERTa finetune model are compared with the manual judgement results. As shown in Chapter 4, a suitable language model is capable of evaluating sentence fluency. The theory, architecture of the language model and the size, composition of the training data set have an effect on the model performance on the task of sentence fluency evaluation.

There are some limitations of this project and more extensive work could be done based on it. First, due to the limited time span and limited hardware configuration, the efficiency of finetuning a deep neural network model is relatively low. The limited RAM size also leads to a mediate training size. Thus, training a deep neural network from scratch is not available in our situation. Second, extrinsic evaluation only involves four languages because it requires people that speak the corresponding native languages to generate input examples and provide the manual evaluation criteria. Besides, the input examples for the extrinsic evaluation are in the pattern of a simple sentence structure and the manual evaluation criteria on some of them may be ambiguous and may differ from person to person. Finally, the large size of data sets and our limited human resource make manual checks on the quality of data sets impossible. Thus, the latent bias and deficiency of data sets may affect the language model performance.

Bibliography

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[2] D. Vrandečić, "Building a multilingual wikipedia," *Communications of the ACM*, vol. 64, pp. 38–41, 04 2021.

[3] A. Ranta, *Grammatical Framework: Programming with Multilingual Grammars*. Stanford: CSLI Publications, 2011.

[4] J. Eisenstein, *Natural Language Processing*, 10 2019.

[5] E. Kavlakoglu, "between three natural language processing concepts," 11 2020. [Online]. Available: https://www.ibm.com/blogs/watson/2020/11/nlp-vs-nlu-vs-nlg-the-differences-between-three-natural-language-processing-concepts/

[6] C. Asli, E. Clark, and J. Gao, "Evaluation of text generation: A survey," 2020.

[7] K. Kaljurand and T. Kuhn, "A multilingual semantic wiki based on attempto controlled english and grammatical framework," 03 2013.

[8] D. Vrandečić, "Invitation to the grammatical framework summer school," 2021. [Online]. Available: https://lists.wikimedia.org/hyperkitty/list/abstract-wikipedia@lists.wikimedia.org/message/X2U4XLID5TTQ7BHNLGYF6EQWCSIQDQDT

[9] E. Reiter, S. Sripada, and R. Robertson, "Acquiring correct knowledge for natural language generation," *J. Artif. Intell. Res. (JAIR)*, vol. 18, pp. 491–516, 2003.

[10] J. Hu, S. Ruder, A. Siddhant, G. Neubig, O. Firat, and M. Johnson, "Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization," *CoRR*, vol. abs/2003.11080, 2020.

[11] F. Ladhak, E. Durmus, C. Cardie, and K. McKeown, "Wikilingua: A new benchmark dataset for cross-lingual abstractive summarization," 01 2020, pp. 4034–4048.

[12] M. Guo, Z. Dai, D. Vrandecic, and R. Al-Rfou, "Wiki-40b: Multilingual language model dataset," in *LREC 2020*, 2020.

[13] G. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, pp. 1527–54, 08 2006.

[14] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "Mpnet: Masked and permuted pre-training for language understanding," in *NeurIPS 2020*. ACM, September 2020. [Online]. Available: https://www.microsoft.com/en-us/research/publication/mpnet-masked-and-permuted-pre-training-for-language-understanding/

[15] A. Feder, K. A. Keith, E. Manzoor, R. Pryzant, D. Sridhar, Z. Wood-Doughty, J. Eisenstein, J. Grimmer, R. Reichart, M. E. Roberts *et al.*, "Causal inference in natural language processing: Estimation, prediction, interpretation and beyond," *arXiv preprint arXiv:2109.00725*, 2021.
[16] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. [Online]. Available: http://arxiv.org/abs/1907.11692
[17] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805
[18] M. Schuster and K. Nakajima, "Japanese and korean voice search," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 5149–5152.
[19] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *CoRR*, vol. abs/1508.07909, 2015. [Online]. Available: http://arxiv.org/abs/1508.07909
[20] M. Ott, S. Edunov, D. Grangier, and M. Auli, "Scaling neural machine translation," *CoRR*, vol. abs/1806.00187, 2018. [Online]. Available: http://arxiv.org/abs/1806.00187
[21] M. Rohit and S. Richard, "Lecture notes: Part ii," *Deep Learning for NLP*, 2016.
[22] H. Tulu, Tilahun, Y. Junqing, and F. Tessfu, Geteye, "Intrinsic and extrinsic automatic evaluation strategies for paraphrase generation systems," *Journal of Computer and Communications*, vol. 8, no. 2, 02 2020.
[23] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS quarterly*, pp. 75–105, 2004.
[24] "lxml - xml and html with python." [Online]. Available: https://lxml.de/
[25] "re — regular expression operations." [Online]. Available: https://docs.python.org/3/library/re.html
[26] "Natural language toolkit." [Online]. Available: https://www.nltk.org/
[27] ""jieba" (chinese for "to stutter") chinese text segmentation." [Online]. Available: https://github.com/fxsjy/jieba
[28] "Mecab: Yet another part-of-speech and morphological analyzer." [Online]. Available: https://taku910.github.io/mecab/
[29] "Transformers." [Online]. Available: https://huggingface.co/docs/transformers/index
[30] "Pytorch." [Online]. Available: https://pytorch.org/
[31] "Google ngram viewer." [Online]. Available: https://books.google.com/ngrams
[32] "roberta-base." [Online]. Available: https://huggingface.co/roberta-base
[33] "hfl/chinese-roberta-wwm-ext." [Online]. Available: https://huggingface.co/hfl/chinese-roberta-wwm-ext
[34] "birgermoell/roberta-swedish." [Online]. Available: https://huggingface.co/birgermoell/roberta-swedish
[35] "nlp-waseda/roberta-base-japanese." [Online]. Available: https://huggingface.co/nlp-waseda/roberta-base-japanese

A

Appendix 1

A.1 n-gram Models for other 42 Languages

Table A.1 and Table A.2 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Aragonés language.

Laplace \ n	1	2	3	4	5
0.0001	241.705570	116.307706	252.561000	640.860762	1186.006466
0.0005	241.706809	97.221692	216.522211	593.552441	1174.282342
0.0010	241.708357	93.180579	215.672263	609.888164	1226.782664
0.0100	241.736314	100.039810	286.691019	868.248543	1706.842293
0.1000	242.024365	156.303228	597.571665	1687.531160	2853.400549
1.0000	245.555160	388.368591	1677.065282	3614.830663	4923.235159

Table A.1: Model perplexity of Aragonés n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.004137	0.008598	0.003959	0.001560	0.000843
0.0005	0.004137	0.010286	0.004618	0.001685	0.000852
0.0010	0.004137	0.010732	0.004637	0.001640	0.000815
0.0100	0.004137	0.009996	0.003488	0.001152	0.000586
0.1000	0.004132	0.006398	0.001673	0.000593	0.000350
1.0000	0.004072	0.002575	0.000596	0.000277	0.000203

Table A.2: Model scores of Aragonés n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.3 and Table A.4 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Belarusian language.

Laplace \ n	1	2	3	4	5
0.0001	1139.285639	598.282931	2270.663895	7686.049100	14380.898442
0.0005	1139.288541	538.690925	2375.430139	8444.194137	15684.879970
0.0010	1139.292170	545.084680	2572.180603	9112.859262	16657.602806
0.0100	1139.357815	735.673809	4135.021017	13249.075810	21960.724804
0.1000	1140.047482	1419.857664	8518.617902	21809.923219	31217.028961
1.0000	1149.536749	3749.602076	19768.491416	36918.420720	45394.694920

Table A.3: Model perplexity of Belarusian n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.000878	0.001671	0.000440	0.000130	0.000070
0.0005	0.000878	0.001856	0.000421	0.000118	0.000064
0.0010	0.000878	0.001835	0.000389	0.000110	0.000060
0.0100	0.000878	0.001359	0.000242	0.000075	0.000046
0.1000	0.000877	0.000704	0.000117	0.000046	0.000032
1.0000	0.000870	0.000267	0.000051	0.000027	0.000022

Table A.4: Model scores of Belarusian n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.5 and Table A.6 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Bulgarian language.

Laplace \ n	1	2	3	4	5
0.0001	1081.316591	453.269780	1893.822202	8991.294109	20218.809332
0.0005	1081.318503	430.249635	1994.385588	9961.774648	22112.313823
0.0010	1081.320894	442.802282	2189.432181	10836.560946	23495.365614
0.0100	1081.364165	624.107133	3829.908563	16259.403937	30671.005263
0.1000	1081.819766	1257.241280	8811.272022	27420.748248	42446.861775
1.0000	1088.183644	3539.553517	22394.871298	46569.408320	59116.604695

Table A.5: Model perplexity of Bulgarian n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.000925	0.002206	0.000528	0.000111	0.000049
0.0005	0.000925	0.002324	0.000501	0.000100	0.000045
0.0010	0.000925	0.002258	0.000457	0.000092	0.000043
0.0100	0.000925	0.001602	0.000261	0.000062	0.000033
0.1000	0.000924	0.000795	0.000113	0.000036	0.000024
1.0000	0.000919	0.000283	0.000045	0.000021	0.000017

Table A.6: Model scores of Bulgarian n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.7 and Table A.8 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Bosanski language.

Laplace \ n	1	2	3	4	5
0.0001	935.427076	655.102160	2435.756728	7140.220402	11918.804302
0.0005	935.430285	565.597578	2380.201910	7498.912275	12662.782784
0.0010	935.434297	562.230404	2501.889331	7934.330213	13281.463839
0.0100	935.506840	713.610292	3686.176535	10856.728600	16828.149079
0.1000	936.264586	1295.920095	7183.592051	17029.865442	23214.221356
1.0000	946.342828	3280.503547	15854.311183	27552.525099	32819.251248

Table A.7: Model perplexity of Bosanski n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.001069	0.001526	0.000411	0.000140	0.000084
0.0005	0.001069	0.001768	0.000420	0.000133	0.000079
0.0010	0.001069	0.001779	0.000400	0.000126	0.000075
0.0100	0.001069	0.001401	0.000271	0.000092	0.000059
0.1000	0.001068	0.000772	0.000139	0.000059	0.000043
1.0000	0.001057	0.000305	0.000063	0.000036	0.000030

Table A.8: Model scores of Bosanski n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.9 and Table A.10 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Danish language.

Laplace \ n	1	2	3	4	5
0.0001	847.519922	561.165523	2393.201350	9609.763288	19053.778646
0.0005	847.521839	486.965427	2308.461399	9987.367606	19931.626350
0.0010	847.524237	480.599000	2433.109827	10569.515798	20729.805902
0.0100	847.567603	586.964111	3722.290619	14409.125973	24987.107531
0.1000	848.021910	1052.295260	7617.924093	21896.607052	31719.295176
1.0000	854.187059	2783.623877	17443.293178	33623.350317	40786.847804

Table A.9: Model perplexity of Danish n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.001180	0.001782	0.000418	0.000104	0.000052
0.0005	0.001180	0.002054	0.000433	0.000100	0.000050
0.0010	0.001180	0.002081	0.000411	0.000095	0.000048
0.0100	0.001180	0.001704	0.000269	0.000069	0.000040
0.1000	0.001179	0.000950	0.000131	0.000046	0.000032
1.0000	0.001171	0.000359	0.000057	0.000030	0.000025

Table A.10: Model scores of Danish n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.11 and Table A.12 present the model perplexity and model score for different n-gram models with various values of n and Laplace for German language.

Laplace \ n	1	2	3	4	5
0.0001	1798.635048	747.532025	3796.606618	26584.591429	74387.598294
0.0005	1798.636725	734.599713	4443.413746	32255.659901	85537.801165
0.0010	1798.638821	760.692467	5062.955619	36250.806343	92226.063314
0.0100	1798.676773	1069.866235	9875.270024	58989.282154	123690.184186
0.1000	1799.078455	2193.616947	25310.653613	104281.878772	173064.155052
1.0000	1804.863663	6568.974126	70871.794259	182787.698136	242305.583524

Table A.11: Model perplexity of German n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.000556	0.001338	0.000263	0.000038	0.000013
0.0005	0.000556	0.001361	0.000225	0.000031	0.000012
0.0010	0.000556	0.001315	0.000198	0.000028	0.000011
0.0100	0.000556	0.000935	0.000101	0.000017	0.000008
0.1000	0.000556	0.000456	0.000040	0.000010	0.000006
1.0000	0.000554	0.000152	0.000014	0.000005	0.000004

Table A.12: Model scores of German n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.13 and Table A.14 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Greek language.

Laplace \ n	1	2	3	4	5
0.0001	1103.531423	497.370490	2760.470620	12514.299167	25256.736506
0.0005	1103.533464	454.061340	2720.325203	13154.534206	26475.745746
0.0010	1103.536016	459.730594	2897.009724	13958.208269	27476.219958
0.0100	1103.582212	613.410269	4573.736899	18979.154619	32585.217357
0.1000	1104.070459	1179.127658	9603.647145	28367.295725	40453.985166
1.0000	1111.023690	3324.890288	22309.898633	42737.048801	51044.048345

Table A.13: Model perplexity of Greek n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.000906	0.002011	0.000362	0.000080	0.000040
0.0005	0.000906	0.002202	0.000368	0.000076	0.000038
0.0010	0.000906	0.002175	0.000345	0.000072	0.000036
0.0100	0.000906	0.001630	0.000219	0.000053	0.000031
0.1000	0.000906	0.000848	0.000104	0.000035	0.000025
1.0000	0.000900	0.000301	0.000045	0.000023	0.000020

Table A.14: Model scores of Greek n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.15 and Table A.16 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Spanish language.

Laplace \ n	1	2	3	4	5
0.0001	1180.141517	345.131974	1330.308184	9305.890112	31521.397941
0.0005	1180.142205	339.988222	1459.368703	10893.245255	35889.001861
0.0010	1180.143064	351.006301	1632.420081	12234.212953	38830.406966
0.0100	1180.158636	488.132081	3174.315461	20943.646329	53832.487245
0.1000	1180.324469	988.258359	8730.750448	40846.625584	78834.140224
1.0000	1182.796106	2801.134341	27548.945322	79305.988234	114982.984521

Table A.15: Model perplexity of Spanish n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.000847	0.002897	0.000752	0.000107	0.000032
0.0005	0.000847	0.002941	0.000685	0.000092	0.000028
0.0010	0.000847	0.002849	0.000613	0.000082	0.000026
0.0100	0.000847	0.002049	0.000315	0.000048	0.000019
0.1000	0.000847	0.001012	0.000115	0.000024	0.000013
1.0000	0.000845	0.000357	0.000036	0.000013	0.000009

Table A.16: Model scores of Spanish n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.17 and Table A.18 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Finnish language.

Laplace \ n	1	2	3	4	5
0.0001	1301.046908	1020.862361	4976.573567	19262.368769	37184.625121
0.0005	1301.051499	912.428032	5285.236289	20901.612070	39457.375419
0.0010	1301.057241	922.018030	5712.165727	22276.235912	41139.969473
0.0100	1301.161000	1235.857792	8792.404582	30043.703486	49583.881544
0.1000	1302.239988	2339.597446	16479.528378	44175.144799	62587.337999
1.0000	1316.239758	5792.950082	34133.697456	66333.872316	80583.044171

Table A.17: Model perplexity of Finnish n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.000769	0.000980	0.000201	0.000052	0.000027
0.0005	0.000769	0.001096	0.000189	0.000048	0.000025
0.0010	0.000769	0.001085	0.000175	0.000045	0.000024
0.0100	0.000769	0.000809	0.000114	0.000033	0.000020
0.1000	0.000768	0.000427	0.000061	0.000023	0.000016
1.0000	0.000760	0.000173	0.000029	0.000015	0.000012

Table A.18: Model scores of Finnish n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.19 and Table A.20 present the model perplexity and model score for different n-gram models with various values of n and Laplace for French language.

Laplace \ n	1	2	3	4	5
0.0001	1122.298534	232.842145	671.417241	4174.639242	15612.360581
0.0005	1122.299070	232.856771	763.901200	5146.226329	18735.703859
0.0010	1122.299740	240.633894	866.482099	5931.397773	20890.779360
0.0100	1122.311877	328.108464	1776.512214	11367.780218	33074.656453
0.1000	1122.440789	644.413661	5258.868562	25944.982310	57226.554841
1.0000	1124.337757	1831.578349	18453.369616	60651.955684	99071.140371

Table A.19: Model perplexity of French n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.000891	0.004295	0.001489	0.000240	0.000064
0.0005	0.000891	0.004294	0.001309	0.000194	0.000053
0.0010	0.000891	0.004156	0.001154	0.000169	0.000048
0.0100	0.000891	0.003048	0.000563	0.000088	0.000030
0.1000	0.000891	0.001552	0.000190	0.000039	0.000017
1.0000	0.000889	0.000546	0.000054	0.000016	0.000010

Table A.20: Model scores of French n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.21 and Table A.22 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Irish language.

Laplace \ n	1	2	3	4	5
0.0001	544.388658	283.193926	953.689665	2966.422832	5705.778865
0.0005	544.390240	235.341942	836.548170	2894.782407	5794.496322
0.0010	544.392218	226.915473	849.205385	3018.119201	6011.000655
0.0100	544.427985	256.677535	1207.560126	4164.896050	7463.575358
0.1000	544.801589	442.935381	2514.554829	6822.637268	10108.953425
1.0000	549.778286	1191.049385	6153.564370	11461.378177	13989.877458

Table A.21: Model perplexity of Irish n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.001837	0.003531	0.001049	0.000337	0.000175
0.0005	0.001837	0.004249	0.001195	0.000345	0.000173
0.0010	0.001837	0.004407	0.001178	0.000331	0.000166
0.0100	0.001837	0.003896	0.000828	0.000240	0.000134
0.1000	0.001836	0.002258	0.000398	0.000147	0.000099
1.0000	0.001819	0.000840	0.000163	0.000087	0.000071

Table A.22: Model scores of Irish n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.23 and Table A.24 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Hakka Chinese language.

Laplace \ n	1	2	3	4	5
0.0001	193.932649	127.383174	223.722659	350.766010	449.270852
0.0005	193.934232	101.164259	191.228614	331.991381	447.026605
0.0010	193.936211	95.959386	189.758519	339.622299	463.126521
0.0100	193.971923	107.297772	244.282893	453.929657	621.435626
0.1000	194.338297	185.514390	453.142444	799.614687	1042.864129
1.0000	198.690813	435.816210	1069.698752	1659.329439	1983.071323

Table A.23: Model perplexity of Hakka Chinese n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.005156	0.007850	0.004470	0.002851	0.002226
0.0005	0.005156	0.009885	0.005229	0.003012	0.002237
0.0010	0.005156	0.010421	0.005270	0.002944	0.002159
0.0100	0.005155	0.009320	0.004094	0.002203	0.001609
0.1000	0.005146	0.005390	0.002207	0.001251	0.000959
1.0000	0.005033	0.002295	0.000935	0.000603	0.000504

Table A.24: Model scores of Hakka Chinese n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.25 and Table A.26 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Indonesian language.

Laplace \ n	1	2	3	4	5
0.0001	1275.915231	623.805873	2743.177408	9442.398528	16777.340159
0.0005	1275.916936	526.519144	2622.451579	10020.235976	17744.479522
0.0010	1275.919068	512.774400	2771.370059	10664.582853	18544.941786
0.0100	1275.957674	620.830145	4350.841745	14664.850008	22875.626852
0.1000	1276.365852	1210.895637	8974.556238	22346.116122	30060.748729
1.0000	1282.202668	3456.346911	19834.588642	34282.573820	39941.484974

Table A.25: Model perplexity of Indonesian n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.000784	0.001603	0.000365	0.000106	0.000060
0.0005	0.000784	0.001899	0.000381	0.000100	0.000056
0.0010	0.000784	0.001950	0.000361	0.000094	0.000054
0.0100	0.000784	0.001611	0.000230	0.000068	0.000044
0.1000	0.000783	0.000826	0.000111	0.000045	0.000033
1.0000	0.000780	0.000289	0.000050	0.000029	0.000025

Table A.26: Model scores of Indonesian n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.27 and Table A.28 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Ilocano language.

Laplace \ n	1	2	3	4	5
0.0001	278.051130	96.027008	184.510911	706.371940	1607.736406
0.0005	278.051942	90.757173	168.986217	691.114159	1676.640684
0.0010	278.052958	91.379248	174.721028	731.910284	1788.503729
0.0100	278.071306	113.261586	276.214890	1158.844180	2610.144209
0.1000	278.260938	199.908062	718.642213	2477.181732	4472.745066
1.0000	280.639056	529.716905	2450.077788	5708.583453	7949.756888

Table A.27: Model perplexity of Ilocano n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.003596	0.010414	0.005420	0.001416	0.000622
0.0005	0.003596	0.011018	0.005918	0.001447	0.000596
0.0010	0.003596	0.010943	0.005723	0.001366	0.000559
0.0100	0.003596	0.008829	0.003620	0.000863	0.000383
0.1000	0.003594	0.005002	0.001392	0.000404	0.000224
1.0000	0.003563	0.001888	0.000408	0.000175	0.000126

Table A.28: Model scores of Ilocano n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.29 and Table A.30 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Icelandic language.

Laplace \ n	1	2	3	4	5
0.0001	636.021725	468.705341	1582.453664	4990.104641	8889.138366
0.0005	636.024069	396.832786	1507.726852	5177.483685	9398.159747
0.0010	636.027001	388.563807	1579.864619	5487.106549	9866.851641
0.0100	636.079970	466.590473	2367.708146	7642.960801	12527.959765
0.1000	636.630066	823.276111	4753.284588	12147.766652	17079.530433
1.0000	643.711769	2102.560123	10805.207793	19677.578851	23614.313669

Table A.29: Model perplexity of Icelandic n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.001572	0.002134	0.000632	0.000200	0.000112
0.0005	0.001572	0.002520	0.000663	0.000193	0.000106
0.0010	0.001572	0.002574	0.000633	0.000182	0.000101
0.0100	0.001572	0.002143	0.000422	0.000131	0.000080
0.1000	0.001571	0.001215	0.000210	0.000082	0.000059
1.0000	0.001553	0.000476	0.000093	0.000051	0.000042

Table A.30: Model scores of Icelandic n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.31 and Table A.32 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Italian language.

Laplace \ n	1	2	3	4	5
0.0001	1723.574822	417.853722	1833.962645	13454.851790	41202.992295
0.0005	1723.575710	411.386867	2074.764952	16182.783175	47586.989290
0.0010	1723.576819	425.336467	2357.917766	18270.874510	51593.735920
0.0100	1723.596935	599.732484	4793.327979	30876.291509	70975.347541
0.1000	1723.812288	1262.028775	13260.961910	57588.972702	101848.594941
1.0000	1727.107622	3906.154817	40270.473346	105942.638132	145015.063515

Table A.31: Model perplexity of Italian n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.000580	0.002393	0.000545	0.000074	0.000024
0.0005	0.000580	0.002431	0.000482	0.000062	0.000021
0.0010	0.000580	0.002351	0.000424	0.000055	0.000019
0.0100	0.000580	0.001667	0.000209	0.000032	0.000014
0.1000	0.000580	0.000792	0.000075	0.000017	0.000010
1.0000	0.000579	0.000256	0.000025	0.000009	0.000007

Table A.32: Model scores of Italian n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.33 and Table A.34 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Javanese language.

Laplace \ n	1	2	3	4	5
0.0001	681.554196	514.973283	1460.003158	3422.659094	5353.302397
0.0005	681.556590	409.692103	1347.854206	3534.235471	5632.413551
0.0010	681.559585	391.855224	1396.117911	3733.889077	5917.818407
0.0100	681.613719	449.919969	2018.333342	5146.093773	7639.506254
0.1000	682.178096	794.014842	3852.102721	8147.958223	10761.216349
1.0000	689.603548	1991.712207	8247.414292	13283.416102	15481.371527

Table A.33: Model perplexity of Javanese n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.001467	0.001942	0.000685	0.000292	0.000187
0.0005	0.001467	0.002441	0.000742	0.000283	0.000178
0.0010	0.001467	0.002552	0.000716	0.000268	0.000169
0.0100	0.001467	0.002223	0.000495	0.000194	0.000131
0.1000	0.001466	0.001259	0.000260	0.000123	0.000093
1.0000	0.001450	0.000502	0.000121	0.000075	0.000065

Table A.34: Model scores of Javanese n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.35 and Table A.36 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Korean language.

Laplace \ n	1	2	3	4	5
0.0001	1393.921869	1119.333407	5538.915090	18792.053884	36689.622954
0.0005	1393.929119	1033.171112	5901.081127	20597.027254	39109.465802
0.0010	1393.938185	1062.843558	6359.634067	21949.433078	40803.495854
0.0100	1394.101903	1492.498366	9509.044382	29283.218688	49408.590925
0.1000	1395.793582	2819.658469	16890.110253	42851.679123	63379.615997
1.0000	1416.871641	6680.604753	33744.201337	65531.908453	83720.602084

Table A.35: Model perplexity of Korean n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.000717	0.000893	0.000181	0.000053	0.000027
0.0005	0.000717	0.000968	0.000169	0.000049	0.000026
0.0010	0.000717	0.000941	0.000157	0.000046	0.000025
0.0100	0.000717	0.000670	0.000105	0.000034	0.000020
0.1000	0.000716	0.000355	0.000059	0.000023	0.000016
1.0000	0.000706	0.000150	0.000030	0.000015	0.000012

Table A.36: Model scores of Korean n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.37 and Table A.38 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Latin language.

Laplace \ n	1	2	3	4	5
0.0001	460.962530	320.089367	793.787174	1844.168273	3090.455959
0.0005	460.965113	262.799273	739.901300	1879.513488	3244.163028
0.0010	460.968343	254.534439	763.859618	1985.152187	3429.376817
0.0100	461.026671	297.714825	1086.776508	2846.131267	4690.358955
0.1000	461.628870	510.555565	2132.365346	4998.412407	7325.420950
1.0000	469.090991	1241.895835	5088.737172	9393.182228	11877.597846

Table A.37: Model perplexity of Latin n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.002169	0.003124	0.001260	0.000542	0.000324
0.0005	0.002169	0.003805	0.001352	0.000532	0.000308
0.0010	0.002169	0.003929	0.001309	0.000504	0.000292
0.0100	0.002169	0.003359	0.000920	0.000351	0.000213
0.1000	0.002166	0.001959	0.000469	0.000200	0.000137
1.0000	0.002132	0.000805	0.000197	0.000106	0.000084

Table A.38: Model scores of Latin n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.39 and Table A.40 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Mongolian language.

Laplace \ n	1	2	3	4	5
0.0001	1542.860899	947.407261	3702.381845	10617.932332	18409.496112
0.0005	1542.864185	741.028603	3531.832908	11246.394941	19301.296169
0.0010	1542.868295	707.819729	3698.275341	11895.433769	20049.710349
0.0100	1542.942703	843.123160	5403.047081	15730.972911	23968.235922
0.1000	1543.730270	1639.993526	9853.835096	22533.827813	29946.004440
1.0000	1555.002990	4384.992135	19144.835137	32356.706178	37591.119385

Table A.39: Model perplexity of Mongolian n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.000648	0.001056	0.000270	0.000094	0.000054
0.0005	0.000648	0.001349	0.000283	0.000089	0.000052
0.0010	0.000648	0.001413	0.000270	0.000084	0.000050
0.0100	0.000648	0.001186	0.000185	0.000064	0.000042
0.1000	0.000648	0.000610	0.000101	0.000044	0.000033
1.0000	0.000643	0.000228	0.000052	0.000031	0.000027

Table A.40: Model scores of Mongolian n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.41 and Table A.42 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Mon language.

Laplace \ n	1	2	3	4	5
0.0001	196.791912	74.664497	182.368094	468.802363	943.784511
0.0005	196.794262	81.214044	214.564667	565.060598	1137.307861
0.0010	196.797200	88.480706	242.163664	638.354677	1274.506380
0.0100	196.850172	141.787239	429.111125	1090.552078	2058.194224
0.1000	197.388216	283.782122	907.845056	2111.136590	3658.039925
1.0000	203.386475	654.951739	2134.341962	4498.184755	7149.328012

Table A.41: Model perplexity of Mon n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.005082	0.013393	0.005483	0.002133	0.001060
0.0005	0.005081	0.012313	0.004661	0.001770	0.000879
0.0010	0.005081	0.011302	0.004129	0.001567	0.000785
0.0100	0.005080	0.007053	0.002330	0.000917	0.000486
0.1000	0.005066	0.003524	0.001102	0.000474	0.000273
1.0000	0.004917	0.001527	0.000469	0.000222	0.000140

Table A.42: Model scores of Mon n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.43 and Table A.44 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Malaysian language.

Laplace \ n	1	2	3	4	5
0.0001	992.864330	583.388627	1914.415236	4865.193427	7558.155027
0.0005	992.866088	462.605476	1730.914093	5014.576815	7915.148262
0.0010	992.868286	438.712607	1785.397367	5280.838098	8266.455872
0.0100	992.908073	490.249864	2601.258275	7123.806712	10353.974446
0.1000	993.328246	895.311236	5054.767355	10892.393473	14098.205005
1.0000	999.286279	2427.821509	10775.019306	17090.007502	19643.595409

Table A.43: Model perplexity of Malaysian n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.001007	0.001714	0.000522	0.000206	0.000132
0.0005	0.001007	0.002162	0.000578	0.000199	0.000126
0.0010	0.001007	0.002279	0.000560	0.000189	0.000121
0.0100	0.001007	0.002040	0.000384	0.000140	0.000097
0.1000	0.001007	0.001117	0.000198	0.000092	0.000071
1.0000	0.001001	0.000412	0.000093	0.000059	0.000051

Table A.44: Model scores of Malaysian n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.45 and Table A.46 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Burmese language.

Laplace \ n	1	2	3	4	5
0.0001	67.366158	55.517155	126.574207	308.321423	603.671867
0.0005	67.366965	54.256080	126.251716	314.252580	623.052437
0.0010	67.367973	56.018044	132.647718	330.739475	654.206228
0.0100	67.386156	74.114909	187.106694	454.502386	865.829499
0.1000	67.570956	122.476647	329.100531	753.724302	1342.653407
1.0000	69.640353	238.441217	693.567728	1499.813922	2497.539341

Table A.45: Model perplexity of Burmese n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.014844	0.018012	0.007901	0.003243	0.001657
0.0005	0.014844	0.018431	0.007921	0.003182	0.001605
0.0010	0.014844	0.017851	0.007539	0.003024	0.001529
0.0100	0.014840	0.013493	0.005345	0.002200	0.001155
0.1000	0.014799	0.008165	0.003039	0.001327	0.000745
1.0000	0.014359	0.004194	0.001442	0.000667	0.000400

Table A.46: Model scores of Burmese n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.47 and Table A.48 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Dutch language.

Laplace \ n	1	2	3	4	5
0.0001	980.400163	427.045857	1722.130664	7992.377658	18457.760775
0.0005	980.401343	398.842048	1821.802202	8999.341897	20397.497110
0.0010	980.402818	403.411479	1995.101586	9863.528272	21752.729189
0.0100	980.429534	523.243737	3485.830606	15218.021006	28854.602328
0.1000	980.712151	987.718929	8238.373328	26648.763650	41339.704631
1.0000	984.767114	2733.009376	22171.308752	47707.843812	60737.912146

Table A.47: Model perplexity of Dutch n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.001020	0.002342	0.000581	0.000125	0.000054
0.0005	0.001020	0.002507	0.000549	0.000111	0.000049
0.0010	0.001020	0.002479	0.000501	0.000101	0.000046
0.0100	0.001020	0.001911	0.000287	0.000066	0.000035
0.1000	0.001020	0.001012	0.000121	0.000038	0.000024
1.0000	0.001015	0.000366	0.000045	0.000021	0.000016

Table A.48: Model scores of Dutch n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.49 and Table A.50 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Norwegian language.

Laplace \ n	1	2	3	4	5
0.0001	592.336728	398.559709	1378.735314	4765.899042	9207.186876
0.0005	592.338468	335.286604	1264.318886	4792.533851	9477.282186
0.0010	592.340644	325.709432	1308.637511	5033.892236	9839.590660
0.0100	592.379991	377.055999	1937.931482	6865.406399	11981.448180
0.1000	592.791229	652.924507	3956.170427	10694.899781	15585.164705
1.0000	598.287749	1709.637080	9200.435223	17012.833390	20667.940124

Table A.49: Model perplexity of Norwegian n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.001688	0.002509	0.000725	0.000210	0.000109
0.0005	0.001688	0.002983	0.000791	0.000209	0.000106
0.0010	0.001688	0.003070	0.000764	0.000199	0.000102
0.0100	0.001688	0.002652	0.000516	0.000146	0.000083
0.1000	0.001687	0.001532	0.000253	0.000094	0.000064
1.0000	0.001671	0.000585	0.000109	0.000059	0.000048

Table A.50: Model scores of Norwegian n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.51 and Table A.52 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Polish language.

Laplace \ n	1	2	3	4	5
0.0001	2312.776860	781.318641	3597.189816	17678.505254	40786.848215
0.0005	2312.779895	769.374944	4296.579068	21474.189410	47390.069721
0.0010	2312.783691	808.569976	4924.862580	24119.662521	51512.956397
0.0100	2312.852403	1242.013251	9540.156785	39105.750996	71870.016940
0.1000	2313.578697	2766.589441	22941.486645	68954.638140	105220.593900
1.0000	2323.953680	8230.156354	58612.934019	120980.139996	153839.932523

Table A.51: Model perplexity of Polish n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.000432	0.001280	0.000278	0.000057	0.000025
0.0005	0.000432	0.001300	0.000233	0.000047	0.000021
0.0010	0.000432	0.001237	0.000203	0.000041	0.000019
0.0100	0.000432	0.000805	0.000105	0.000026	0.000014
0.1000	0.000432	0.000361	0.000044	0.000015	0.000010
1.0000	0.000430	0.000122	0.000017	0.000008	0.000007

Table A.52: Model scores of Polish n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.53 and Table A.54 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Portuguese language.

Laplace \ n	1	2	3	4	5
0.0001	1273.186696	411.198127	1624.103583	9147.090741	22811.009231
0.0005	1273.187606	393.155324	1723.162695	10556.719027	26002.068118
0.0010	1273.188744	402.127151	1907.183981	11732.321432	28089.660050
0.0100	1273.209364	551.581465	3581.831916	18997.004587	38489.343968
0.1000	1273.429622	1126.445896	9320.453735	34313.104423	55467.175352
1.0000	1276.759314	3299.359095	26663.001221	61252.309756	79338.575220

Table A.53: Model perplexity of Portuguese n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.000785	0.002432	0.000616	0.000109	0.000044
0.0005	0.000785	0.002544	0.000580	0.000095	0.000038
0.0010	0.000785	0.002487	0.000524	0.000085	0.000036
0.0100	0.000785	0.001813	0.000279	0.000053	0.000026
0.1000	0.000785	0.000888	0.000107	0.000029	0.000018
1.0000	0.000783	0.000303	0.000038	0.000016	0.000013

Table A.54: Model scores of Portuguese n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.55 and Table A.56 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Romanian language.

Laplace \ n	1	2	3	4	5
0.0001	1098.223317	495.735216	2068.617842	8558.621189	17921.232700
0.0005	1098.225076	449.179759	2065.718470	9203.751778	19264.102866
0.0010	1098.227276	453.235973	2215.548758	9897.537917	20320.046239
0.0100	1098.267126	602.244239	3628.060632	14336.382763	25908.272762
0.1000	1098.690620	1168.188855	7989.332246	23310.248471	34974.750202
1.0000	1104.897965	3249.010199	19488.610691	38112.327740	47419.822346

Table A.55: Model perplexity of Romanian n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.000911	0.002017	0.000483	0.000117	0.000056
0.0005	0.000911	0.002226	0.000484	0.000109	0.000052
0.0010	0.000911	0.002206	0.000451	0.000101	0.000049
0.0100	0.000911	0.001660	0.000276	0.000070	0.000039
0.1000	0.000910	0.000856	0.000125	0.000043	0.000029
1.0000	0.000905	0.000308	0.000051	0.000026	0.000021

Table A.56: Model scores of Romanian n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.57 and Table A.58 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Russian language.

Laplace \ n	1	2	3	4	5
0.0001	2885.442305	776.734816	3913.034292	25241.411399	71153.958889
0.0005	2885.444728	800.492977	4926.149804	32111.715426	85102.817398
0.0010	2885.447758	855.192945	5775.981770	36774.251862	93534.172769
0.0100	2885.502634	1379.698449	12124.437520	63656.726016	134980.454881
0.1000	2886.085142	3220.206591	31840.908463	119929.236836	203827.425108
1.0000	2894.609337	9979.901405	89344.448698	223379.731795	305283.496275

Table A.57: Model perplexity of Russian n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.000347	0.001287	0.000256	0.000040	0.000014
0.0005	0.000347	0.001249	0.000203	0.000031	0.000012
0.0010	0.000347	0.001169	0.000173	0.000027	0.000011
0.0100	0.000347	0.000725	0.000082	0.000016	0.000007
0.1000	0.000346	0.000311	0.000031	0.000008	0.000005
1.0000	0.000345	0.000100	0.000011	0.000004	0.000003

Table A.58: Model scores of Russian n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.59 and Table A.60 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Sicilian language.

Laplace \ n	1	2	3	4	5
0.0001	323.036331	193.549203	500.360347	1224.809818	1966.993108
0.0005	323.037973	158.400203	429.118600	1159.159404	1974.419241
0.0010	323.040027	151.218510	428.145236	1189.367201	2042.752241
0.0100	323.077106	162.375705	559.629814	1575.841615	2578.943879
0.1000	323.459435	259.639398	1094.621371	2665.722973	3791.410829
1.0000	328.167698	659.187411	2769.836263	5002.881566	6041.791933

Table A.59: Model perplexity of Sicilian n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.003096	0.005167	0.001999	0.000816	0.000508
0.0005	0.003096	0.006313	0.002330	0.000863	0.000506
0.0010	0.003096	0.006613	0.002336	0.000841	0.000490
0.0100	0.003095	0.006159	0.001787	0.000635	0.000388
0.1000	0.003092	0.003851	0.000914	0.000375	0.000264
1.0000	0.003047	0.001517	0.000361	0.000200	0.000166

Table A.60: Model scores of Sicilian n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.61 and Table A.62 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Scottish language.

Laplace \ n	1	2	3	4	5
0.0001	571.986039	367.564815	1167.541046	3247.510229	5866.543580
0.0005	571.987497	300.334243	1040.619202	3232.672815	6057.872524
0.0010	571.989320	288.358690	1064.060010	3388.024498	6327.423659
0.0100	572.022307	326.552424	1513.469192	4665.006773	7994.037443
0.1000	572.368879	558.113378	2998.106752	7499.087308	10899.745678
1.0000	577.135607	1404.609761	6925.050842	12357.742994	15079.643831

Table A.61: Model perplexity of Scottish n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.001748	0.002721	0.000857	0.000308	0.000170
0.0005	0.001748	0.003330	0.000961	0.000309	0.000165
0.0010	0.001748	0.003468	0.000940	0.000295	0.000158
0.0100	0.001748	0.003062	0.000661	0.000214	0.000125
0.1000	0.001747	0.001792	0.000334	0.000133	0.000092
1.0000	0.001733	0.000712	0.000144	0.000081	0.000066

Table A.62: Model scores of Scottish n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.63 and Table A.64 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Slovak language.

Laplace \ n	1	2	3	4	5
0.0001	879.423328	602.831477	2132.135292	6155.440446	10426.649477
0.0005	879.426581	515.197320	2099.512948	6435.858315	10977.712956
0.0010	879.430649	509.952819	2208.092222	6793.324991	11470.300565
0.0100	879.504187	636.477261	3209.580925	9159.850953	14294.471373
0.1000	880.271734	1144.525448	6095.979968	14215.749843	19526.253805
1.0000	890.428002	2878.931426	13372.608241	23260.533481	27893.812304

Table A.63: Model perplexity of Slovak n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.001137	0.001659	0.000469	0.000162	0.000096
0.0005	0.001137	0.001941	0.000476	0.000155	0.000091
0.0010	0.001137	0.001961	0.000453	0.000147	0.000087
0.0100	0.001137	0.001571	0.000312	0.000109	0.000070
0.1000	0.001136	0.000874	0.000164	0.000070	0.000051
1.0000	0.001123	0.000347	0.000075	0.000043	0.000036

Table A.64: Model scores of Slovak n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.65 and Table A.66 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Serbian language.

Laplace \ n	1	2	3	4	5
0.0001	1304.086542	721.767409	3208.620725	11457.761192	20906.958973
0.0005	1304.090089	661.573699	3338.893388	12503.457653	22649.352861
0.0010	1304.094524	674.429809	3598.855987	13420.539034	23906.943916
0.0100	1304.174731	925.208280	5712.516303	19039.677176	30681.685315
0.1000	1305.014920	1805.585976	11753.430057	30385.829373	42345.055455
1.0000	1316.389480	4821.906720	26827.767584	49493.959884	59544.506680

Table A.65: Model perplexity of Serbian n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.000767	0.001385	0.000312	0.000087	0.000048
0.0005	0.000767	0.001512	0.000300	0.000080	0.000044
0.0010	0.000767	0.001483	0.000278	0.000075	0.000042
0.0100	0.000767	0.001081	0.000175	0.000053	0.000033
0.1000	0.000766	0.000554	0.000085	0.000033	0.000024
1.0000	0.000760	0.000207	0.000037	0.000020	0.000017

Table A.66: Model scores of Serbian n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.67 and Table A.68 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Thai language.

Laplace \ n	1	2	3	4	5
0.0001	62.918893	40.606102	84.726049	201.588018	398.030417
0.0005	62.919388	40.679712	87.252408	211.782018	419.351505
0.0010	62.920008	42.201955	92.839457	226.275462	445.032305
0.0100	62.931180	55.870653	137.673076	328.484870	614.671254
0.1000	63.045091	93.279898	255.799102	569.680108	988.043192
1.0000	64.350954	188.262115	556.366200	1150.705198	1855.088054

Table A.67: Model perplexity of Thai n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.015893	0.024627	0.011803	0.004961	0.002512
0.0005	0.015893	0.024582	0.011461	0.004722	0.002385
0.0010	0.015893	0.023696	0.010771	0.004419	0.002247
0.0100	0.015890	0.017898	0.007264	0.003044	0.001627
0.1000	0.015862	0.010720	0.003909	0.001755	0.001012
1.0000	0.015540	0.005312	0.001797	0.000869	0.000539

Table A.68: Model scores of Thai n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.69 and Table A.70 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Filipino language.

Laplace \ n	1	2	3	4	5
0.0001	546.367652	149.453331	287.042886	777.280667	1314.462165
0.0005	546.368611	141.206348	291.244401	876.766348	1558.462430
0.0010	546.369810	143.603039	316.705734	985.952082	1762.817007
0.0100	546.391516	190.385078	575.266112	1823.806338	3095.667392
0.1000	546.620383	366.885959	1559.161828	4176.261924	6176.409312
1.0000	549.839557	1044.477802	5101.899925	9936.294687	12540.795685

Table A.69: Model perplexity of Filipino n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.001830	0.006691	0.003484	0.001287	0.000761
0.0005	0.001830	0.007082	0.003434	0.001141	0.000642
0.0010	0.001830	0.006964	0.003158	0.001014	0.000567
0.0100	0.001830	0.005253	0.001738	0.000548	0.000323
0.1000	0.001829	0.002726	0.000641	0.000239	0.000162
1.0000	0.001819	0.000957	0.000196	0.000101	0.000080

Table A.70: Model scores of Filipino n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.71 and Table A.72 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Turkish language.

Laplace \ n	1	2	3	4	5
0.0001	1622.338550	953.697679	4263.589685	15302.995813	29372.179023
0.0005	1622.342123	818.285879	4396.981138	16731.097435	31468.810383
0.0010	1622.346593	814.135458	4734.363498	17955.397249	32985.965735
0.0100	1622.427476	1071.481471	7449.660101	24939.474132	40489.225723
0.1000	1623.280841	2114.156291	14586.235805	37338.437886	51675.154124
1.0000	1635.299231	5616.586933	30721.088994	55847.700050	66268.467994

Table A.71: Model perplexity of Turkish n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.000616	0.001049	0.000235	0.000065	0.000034
0.0005	0.000616	0.001222	0.000227	0.000060	0.000032
0.0010	0.000616	0.001228	0.000211	0.000056	0.000030
0.0100	0.000616	0.000933	0.000134	0.000040	0.000025
0.1000	0.000616	0.000473	0.000069	0.000027	0.000019
1.0000	0.000612	0.000178	0.000033	0.000018	0.000015

Table A.72: Model scores of Turkish n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.73 and Table A.74 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Ukrainian language.

Laplace \ n	1	2	3	4	5
0.0001	2132.993350	658.423248	2659.140432	11850.955795	26380.465177
0.0005	2132.996190	646.050860	3146.341092	14418.000892	31076.725883
0.0010	2132.999742	675.732380	3587.360874	16237.703861	34091.179468
0.0100	2133.064017	1014.896940	6891.911053	27180.988032	50145.850750
0.1000	2133.741775	2223.385063	17028.002329	51407.938836	79901.705454
1.0000	2143.296988	6684.738401	46031.306066	98285.160961	128421.017586

Table A.73: Model perplexity of Ukrainian n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.000469	0.001519	0.000376	0.000084	0.000038
0.0005	0.000469	0.001548	0.000318	0.000069	0.000032
0.0010	0.000469	0.001480	0.000279	0.000062	0.000029
0.0100	0.000469	0.000985	0.000145	0.000037	0.000020
0.1000	0.000469	0.000450	0.000059	0.000019	0.000013
1.0000	0.000467	0.000150	0.000022	0.000010	0.000008

Table A.74: Model scores of Ukrainian n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.75 and Table A.76 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Vietnamese language.

Laplace \ n	1	2	3	4	5
0.0001	984.297673	178.154831	452.381861	2247.855560	6344.416196
0.0005	984.298365	162.611858	432.320809	2425.841787	6999.907735
0.0010	984.299231	159.397096	458.447199	2672.084329	7560.762270
0.0100	984.314889	174.901075	808.891493	4551.597131	10839.331829
0.1000	984.479043	306.789975	2318.948149	9372.328450	17068.420424
1.0000	986.727661	1006.859989	7846.437574	19399.727439	26845.983027

Table A.75: Model perplexity of Vietnamese n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.001016	0.005613	0.002211	0.000445	0.000158
0.0005	0.001016	0.006150	0.002313	0.000412	0.000143
0.0010	0.001016	0.006274	0.002181	0.000374	0.000132
0.0100	0.001016	0.005718	0.001236	0.000220	0.000092
0.1000	0.001016	0.003260	0.000431	0.000107	0.000059
1.0000	0.001013	0.000993	0.000127	0.000052	0.000037

Table A.76: Model scores of Vietnamese n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.77 and Table A.78 present the model perplexity and model score for different n-gram models with various values of n and Laplace for Waray language.

Laplace \ n	1	2	3	4	5
0.0001	106.365231	8.874703	7.414110	13.457090	21.841800
0.0005	106.365967	9.056254	7.971109	14.539546	24.363533
0.0010	106.366887	9.331822	8.545236	15.729177	26.782127
0.0100	106.383470	11.573307	13.099224	25.523704	45.591369
0.1000	106.551673	18.208901	29.857864	62.026677	112.444368
1.0000	108.416996	47.582357	117.686927	232.919067	394.004638

Table A.77: Model perplexity of Waray n-gram models with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.009402	0.112680	0.134878	0.074310	0.045784
0.0005	0.009402	0.110421	0.125453	0.068778	0.041045
0.0010	0.009401	0.107160	0.117024	0.063576	0.037338
0.0100	0.009400	0.086406	0.076340	0.039179	0.021934
0.1000	0.009385	0.054918	0.033492	0.016122	0.008893
1.0000	0.009224	0.021016	0.008497	0.004293	0.002538

Table A.78: Model scores of Waray n-gram models with different n and Laplace values. The higher the score, the better the model.

Table A.79 and Table A.80 present the model perplexity and model score for different n-gram models with various values of n and Laplace with ‘manual’ text segmentation method for Chinese Classical language.

Laplace \ n	1	2	3	4	5
0.0001	534.805038	433.760506	1006.463738	1730.571935	2490.057944
0.0005	534.805079	334.405078	678.465223	1501.884924	2378.495721
0.0010	534.805131	301.241113	613.077268	1490.035167	2401.349591
0.0100	534.806075	234.079726	632.492751	1786.505993	2765.202899
0.1000	534.817371	263.051037	1082.501112	2617.052394	3538.479449
1.0000	535.082961	537.323390	2278.563421	3936.734848	4571.204396

Table A.79: Model perplexity of Chinese Classical n-gram models (with ‘manual’ text segmentation method) with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.001870	0.002305	0.000994	0.000578	0.000402
0.0005	0.001870	0.002990	0.001474	0.000666	0.000420
0.0010	0.001870	0.003320	0.001631	0.000671	0.000416
0.0100	0.001870	0.004272	0.001581	0.000560	0.000362
0.1000	0.001870	0.003802	0.000924	0.000382	0.000283
1.0000	0.001869	0.001861	0.000439	0.000254	0.000219

Table A.80: Model scores of Chinese Classical n-gram models (with ‘manual’ text segmentation method) with different n and Laplace values. The higher the score, the better the model.

Table A.81 and Table A.82 present the model perplexity and model score for different n-gram models with various values of n and Laplace with ‘jieba’ text segmentation method for Chinese Classical language.

Laplace \ n	1	2	3	4	5
0.0001	280.027239	272.492230	1112.554185	3978.693047	8193.172518
0.0005	280.028730	240.166643	1053.678303	3941.301996	8242.505929
0.0010	280.030595	238.545846	1090.565765	4087.583417	8483.473051
0.0100	280.064261	289.070341	1499.218468	5275.506131	10160.242432
0.1000	280.411038	474.128407	2634.756992	7945.221718	13373.239385
1.0000	284.656075	1021.531748	5585.439549	13144.268120	18487.899452

Table A.81: Model perplexity of Chinese Classical n-gram models (with ‘jieba’ text segmentation method) with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.003571	0.003670	0.000899	0.000251	0.000122
0.0005	0.003571	0.004164	0.000949	0.000254	0.000121
0.0010	0.003571	0.004192	0.000917	0.000245	0.000118
0.0100	0.003571	0.003459	0.000667	0.000190	0.000098
0.1000	0.003566	0.002109	0.000380	0.000126	0.000075
1.0000	0.003513	0.000979	0.000179	0.000076	0.000054

Table A.82: Model scores of Chinese Classical n-gram models (with ‘jieba’ text segmentation method) with different n and Laplace values. The higher the score, the better the model.

Table A.83 and Table A.84 present the model perplexity and model score for different n-gram models with various values of n and Laplace with 'manual' text segmentation method for Chinese Traditional language.

Laplace \ n	1	2	3	4	5
0.0001	888.835811	145.703414	171.750128	874.902799	3966.279269
0.0005	888.836004	142.024671	172.898617	1033.145126	4843.968981
0.0010	888.836246	141.236512	185.248645	1196.919186	5523.301705
0.0100	888.840613	146.392187	335.655598	2560.690149	9787.756519
0.1000	888.885857	192.558082	1090.159496	7174.721028	19575.432891
1.0000	889.464685	460.938868	4976.337705	20781.371574	38701.875666

Table A.83: Model perplexity of Chinese Traditional n-gram models (with 'manual' text segmentation method) with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.001125	0.006863	0.005822	0.001143	0.000252
0.0005	0.001125	0.007041	0.005784	0.000968	0.000206
0.0010	0.001125	0.007080	0.005398	0.000835	0.000181
0.0100	0.001125	0.006831	0.002979	0.000391	0.000102
0.1000	0.001125	0.005193	0.000917	0.000139	0.000051
1.0000	0.001124	0.002169	0.000201	0.000048	0.000026

Table A.84: Model scores of Chinese Traditional n-gram models (with 'manual' text segmentation method) with different n and Laplace values. The higher the score, the better the model.

Table A.85 and Table A.86 present the model perplexity and model score for different n-gram models with various values of n and Laplace with ‘jieba’ text segmentation method for Chinese Traditional language.

Laplace \ n	1	2	3	4	5
0.0001	2550.838688	746.084347	4146.219391	24932.829323	60420.899310
0.0005	2550.842052	742.495532	5023.707682	31096.775284	71821.395575
0.0010	2550.846259	777.042717	5803.906574	35221.793789	78708.006757
0.0100	2550.922315	1174.982137	11690.714339	58547.316829	112893.846055
0.1000	2551.716225	2735.914864	30140.226807	106531.540221	170394.386250
1.0000	2562.305920	9056.904186	83159.298182	192431.326087	254789.324508

Table A.85: Model perplexity of Chinese Traditional n-gram models (with ‘jieba’ text segmentation method) with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.000392	0.001340	0.000241	0.000040	0.000017
0.0005	0.000392	0.001347	0.000199	0.000032	0.000014
0.0010	0.000392	0.001287	0.000172	0.000028	0.000013
0.0100	0.000392	0.000851	0.000086	0.000017	0.000009
0.1000	0.000392	0.000366	0.000033	0.000009	0.000006
1.0000	0.000390	0.000110	0.000012	0.000005	0.000004

Table A.86: Model scores of Chinese Traditional n-gram models (with ‘jieba’ text segmentation method) with different n and Laplace values. The higher the score, the better the model.

Table A.87 and Table A.88 present the model perplexity and model score for different n-gram models with various values of n and Laplace with ‘manual’ text segmentation method for Cantonese language.

Laplace \ n	1	2	3	4	5
0.0001	729.042886	229.439646	505.925006	1219.024790	2118.405911
0.0005	729.043338	188.056661	383.400053	1122.553679	2138.194560
0.0010	729.043904	174.525427	366.789312	1156.438523	2234.397877
0.0100	729.054136	153.607483	469.625990	1625.964154	2946.596540
0.1000	729.160945	209.787153	1030.230676	2861.885076	4338.490151
1.0000	730.589031	567.516481	2752.770709	5115.427617	6337.482881

Table A.87: Model perplexity of Cantonese n-gram models (with ‘manual’ text segmentation method) with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.001372	0.004358	0.001977	0.000820	0.000472
0.0005	0.001372	0.005318	0.002608	0.000891	0.000468
0.0010	0.001372	0.005730	0.002726	0.000865	0.000448
0.0100	0.001372	0.006510	0.002129	0.000615	0.000339
0.1000	0.001371	0.004767	0.000971	0.000349	0.000230
1.0000	0.001369	0.001762	0.000363	0.000195	0.000158

Table A.88: Model scores of Cantonese n-gram models (with ‘manual’ text segmentation method) with different n and Laplace values. The higher the score, the better the model.

Table A.89 and Table A.90 present the model perplexity and model score for different n-gram models with various values of n and Laplace with 'jieba' text segmentation method for Cantonese language.

Laplace \ n	1	2	3	4	5
0.0001	791.025275	443.566146	1489.859780	4154.486235	6933.985410
0.0005	791.028272	360.304640	1348.598948	4295.009427	7419.403861
0.0010	791.032019	345.504906	1390.528155	4553.409315	7838.946905
0.0100	791.099682	397.241675	2046.466556	6362.736378	10174.487034
0.1000	791.798206	737.486799	4159.105765	10186.955622	14160.982842
1.0000	800.484398	2051.810206	9448.284841	16522.393631	19716.862291

Table A.89: Model perplexity of Cantonese n-gram models (with 'jieba' text segmentation method) with different n and Laplace values. The lower the perplexity, the better the model.

Laplace \ n	1	2	3	4	5
0.0001	0.001264	0.002254	0.000671	0.000241	0.000144
0.0005	0.001264	0.002775	0.000742	0.000233	0.000135
0.0010	0.001264	0.002894	0.000719	0.000220	0.000128
0.0100	0.001264	0.002517	0.000489	0.000157	0.000098
0.1000	0.001263	0.001356	0.000240	0.000098	0.000071
1.0000	0.001249	0.000487	0.000106	0.000061	0.000051

Table A.90: Model scores of Cantonese n-gram models (with 'jieba' text segmentation method) with different n and Laplace values. The higher the score, the better the model.