Automatic Text Summarization for Mongolian Language

Biligsaikhan Khurtsbayar, 2600220460-5

Jan 20, 2025

1 Introduction

This report summarizes my research during the Fall 2024 semester for Graduation Research 1, focusing on Automatic Text Summarization (ATS). The selected papers address the distinct challenges presented by the Mongolian language-based text summarization task, requiring a look into more specialized techniques for effective processing. The report explores a wide range of topics, from foundational NLP concepts to key technologies in text summarization, including topic modelling, graph-based ranking, evaluation metrics, and state-of-the-art deep learning methods. The research aims to formulate an optimal system of methods to generate high-quality text summaries for low-resource languages, such as Mongolian.

2 Automatic Text Summarization (ATS)

As the data available on the internet consistently grows, the need to reduce its redundancy and transform it into human-interpretable, concise information becomes increasingly necessary. However, manually performing it is a labor-intensive process that demands significant expertise. As a result, ATS remains an open problem in NLP that requires constant optimization [1].

2.1 ATS operations

By analyzing ATS systems, human experts identified nine operations used to generate summaries, categorized into single-sentence and multi-sentence types. Single-sentence operations focus on removing or modifying phrases and syntactic elements to enhance clarity. On the other hand, multi-sentence operations manage merging, ordering, clustering, and selecting sentences to optimize the summary structure. Through various combinations of these operations, ATS systems effectively achieve sentence compression and produce coherent, concise summaries [2].

2.2 ATS objectives

A successful ATS system achieves two key objectives: extracting the main idea of the input text while reducing its size and eliminating repetitive information. The process typically involves three main steps: pre-processing (NLP pipeline), processing (summarization model), and post-processing (refinement). ATS uses a wide range of technologies, and to achieve the best performance for a specific system, a structured approach is necessary. ATS classification serves as this starting point, narrowing down the options for system optimization. These systems are classified commonly based on their input size, output nature, algorithm, content type, domain, summary type, base language, and, most commonly, the summarization approach [3].

2.3 Summarization approach

Generally, based on the summarization approach, ATS systems are classified into abstractive and extractive text summarization. The abstractive text summarization creates a summary by analyzing the key concept of the input. To produce a human-like summary, it constructs new phrases or utilizes compression and fusion technology to rephrase existing ones. On the other hand, extractive text summarization is when the keywords and statistical features within the input are selected based on ranking to form a summary. While abstractive summarization offers advantages in generating more natural-sounding summaries, it requires high computational power and domain-specific knowledge. However, a third approach, the hybrid approach, combines the strengths of both methods to mitigate their drawbacks. These hybrid systems typically start with extractive techniques to identify key information, followed by merging and rephrasing technologies to produce a cohesive, abstract summary [2].

3 Natural Language Processing (NLP)

Natural language processing is a collection of computational techniques designed to process language in a human-like manner. Additionally, to create these high-performing text summarization models, the model needs processing technology, and the training data also requires pre-processing steps and tools. For language understanding, NLP systems follow a structured seven-layer approach: phonology, morphology, lexical, syntactic, semantic, discourse, and pragmatic. Phonology analyzes speech sounds, encoding them as digital signals. Morphology splits words into their smallest meaningful parts (morphemes). Lexical resolves ambiguities within words with multiple meanings. Syntactic uncovers grammatical structures and dependencies in sentences. Semantics are like lexical but deal with sentences instead of words. Discourse connects sentences to derive coherent meaning from text. Pragmatic interprets implied meanings using context and world

knowledge. Combining solutions across these layers enables NLP systems to address increasingly complex language tasks [4].

3.1 NLP pipeline

The NLP pipeline is a sequence of common steps taken to process textual data. The pipeline begins with preliminary processing, which includes data collection and text cleaning to ensure input quality. Non-essential elements, such as stop words, numbers, and punctuation, are removed to reduce noise. Following this, initial processing prepares the text for analysis, including language identification, which solves issues in multilingual datasets where language switching and transliteration (using a different script than the original language) are frequent. Once language-specific pre-processing is complete, text normalization converts inconsistent text into its canonical form. As not every language is the same, some require graphematic analysis to adjust their tokenization to their needs. The next step focuses on reducing words to their base form, using either stemming, which removes suffixes from the word stem, or lemmatization, which generates linguistically accurate dictionary base forms of words. While stemming is faster, lemmatization provides precision. When pre-built linguistic tools are available, lemmatization is often preferred. Finally, POS tagging assigns parts of speech to each word (noun, adjective, verb). It enables tasks such as named entity recognition, which identifies proper nouns representing individuals or organizations [5].

3.2 Tokenization

Tokenization plays a key role in text pre-processing by breaking textual data into meaningful component elements. As deep-learning language models use these elements called tokens as input, tokenization significantly impacts its output performance. One of the main reasons for the drop in language models' performance is their ability to handle out-of-word vocabulary (OOV). Due to these unique tokens not being in the model input vocabulary, they get mapped to the embedding without consideration of its context. Still, its negative impacts can be mitigated by either increasing the vocabulary size of the input or by using a tokenization algorithm that creates tokens in different levels of representation (word, sub-word, character) [6].

3.3 SentencePiece

SentencePiece is a state-of-the-art unsupervised sub-word tokenizer created to solve the issue of OOV words. It learns sub-words from the input text using byte-pair encoding (BPE). This algorithm iteratively merges the most commonly occurring substrings until the vocabulary reaches the desired size. This approach enables SentencePiece to handle OOV words effectively by potentially splitting them into known BPE-generated

sub-words. The SentencePiece framework consists of four main components: normalizer, trainer, encoder, and decoder. Normalizer standardizes semantically equivalent characters into normalized form. The trainer uses the normalized corpus to train the sub-word segmentation model. The encoder uses the trained model and tokenizes the input into a sequence of sub-words. The decoder transforms the sub-word sequence into normalized text. In addition to not requiring a pre-tokenized dataset, SentencePiece uses sub-word occurrence count instead of researched linguistic rules. These traits make SentencePiece a suitable tool for low-resource languages [7].

4 Mongolian Text Summarization

Mongolian is an agglutinative language with two key morphological characteristics. Firstly, it allows the addition of multiple suffixes to the word stem. Derivational suffixes transform the stem into a new, related word, while inflectional suffixes modify grammatical features such as tense or voice. Secondly, when a word utilizes both types of suffixes, derivational is applied first, as the inflectional suffixes modify the newly derived form of the stem word after it [8].

4.1 Prefix-Root-Postfix Encoding (PRPE)

When developing pre-training language models for low-resource languages (LRLs) like Mongolian, the issue of limited training data often leads to underperformance due to issues such as OOV. Although SentencePiece manages the challenges of OOV words sufficiently, alternative case-specific tailored approaches, such as PRPE, should be further researched for optimal performance.

PRPE is a sub-word segmentation technique that helps to mitigate the issues by reducing vocabulary size and better modelling morphologically rich, agglutinative languages like Mongolian, where affixation creates large vocabularies and rare words. PRPE is a semi-supervised sub-word segmentation algorithm that helps mitigate this by reducing vocabulary size. It is more suitable for morphologically rich, agglutinative languages like Mongolian, where affixation creates extensive vocabularies and rare words (OOV words). PRPE works in the learning and application phases. The learning phase analyzes a corpus to identify frequent prefixes, roots, and postfixes, ranking these based on frequency. The application phase then segments data using these ranked lists, choosing the most likely sub-words [9].

4.2 Research gap

There is a research gap in the application of text summarization in the Mongolian language, especially for abstractive approaches. These approaches lack global semantic information integration, and their heavy reliance on autoregressive decoders makes them

prone to exposure bias. As the global semantic information problem solution, topic modelling can extract and group core ideas of the source text into different topics. These topics improve the generation process by working as its guidance. As for exposure bias, instead of evaluating the generated summaries in comparison to their labels, contrastive learning can improve upon it by generating candidate summaries and calculating their similarities to the original text. It enables the model to identify and remove inconsistent sections from the output summary [10].

5 Topic Modeling

Topic modelling is a task to find hidden themes behind an input document. Latent Dirichlet Allocation (LDA), a generative model, first divides the input sequence into tokens and then iteratively calculates probability distributions. It identifies and generates the topics representing the entire document using the document-topic, topic-word distribution. In the summarization task, to better implement the importance of the main themes of the input text to the generated summaries, topic modelling can be used in combination with other technology, such as graph-based ranking algorithms [11].

Topic modelling is a sensitive task that becomes unreliable due to sparsity and noise issues. Adding to this issue, LDA requires the user for the parameter to specify the number of topics as the method output. As the model performance varies depending on this input, the topic modelling becomes more unreliable. User initialization in LDA setup should be optimized to ensure consistent model performance. Through improvements in pre-processing steps, the Zimm approach stabilizes LDA topic modelling. It applies a coherent utility process (CUP) to a bag of words (BoW) to generate a word cloud and reduces its noise to extract meaningful insights. Then, a prominent extraction technique (PET) was applied to reduce the size of BoW. Finally, principal component analysis (PCA) creates an eigenvalue heuristic to determine the optimal number of topics for LDA. Through such a process, the approach creates a smaller, noise-free word list without losing relevant information and providing a more stable output [12].

6 Graphs-based ranking algorithms

Graphs used for information extraction in text summarization, implemented to display relationships between nodes, are strong tools capable of solving problems with only three key pieces of information. The base unit of a problem is to set up vertices. The relationship type for measuring is to make weighted edges between the vertices and the ranking algorithm to determine the most influential vertex within the graph [13].

6.1 PageRank

PageRank is a ranking algorithm used by Google to order search engine results. Starting the algorithm with all nodes assigned to random values, with an iterative voting process among the nodes in the graph, PageRank ranks the nodes independent from their initial value. This voting process only considers two details: the amount of votes a node gets from other nodes within the graph and the importance or the weight of the nodes that voted for the node [14].

6.2 TextRank

TextRank is a graph-based ranking algorithm built on PageRank, which essentially replaces web pages with input document sentences as nodes for creating the graph. After the graph node creation, the algorithm uses a similarity measure, like cosine similarity, to set edge weights between dependent sentences. Through iteration, it recalculates node scores using the edge weights with its neighbouring nodes, and that process continues until the scores stabilize, and then a summary is created using it [15].

6.3 LexRank

LexRank is an alternative extractive summarization ranking algorithm that uses a graph-based approach to estimate the importance of sentence input text. To start the algorithm, LexRank uses term frequency (TF) to determine the number of occurrences for each token in the input and calculates inverse document frequency (IDF) to check how many sentences contain specific tokens. Then, a graph of the input text is constructed, with sentences as the nodes, and edge weights are assigned to represent the strength of relationships between them, which the cosine similarity between TF-IDF vectors calculates the relationship of each connected sentence. A threshold on the edge creates a binary graph based on their weight, which is then adjusted based on how many connections each sentence has actively. Finally, it iteratively repeats until the scores stabilize and summaries are created by selecting top-scoring sentences. Compared to TextRank, which is effective for short text, LexRank is better suited for long texts with multiple topics [15].

7 Deep Learning ATS models

While ranking-based extractive summarization model-guided topic modelling can output more contextual summaries, transformer-based encoder-decoder models are also shown to be useful for more complex summarization tasks. Examples such as BART and PEGASUS, both using the pre-training, generate higher-quality outputs for text summarization. The encoder-decoder, a commonly used architecture in sequence-to-sequence modelling,

consists of two working parts: encoder and decoder. The encoder uses RNN to process an input sequence one token at a time, transforming it into its semantic representation. The representation created at the final step of the encoding process is called the final hidden state or context vector, and it compactly contains all the information from the input text. The decoder then takes this context vector with the previously generated token, hidden state and produces the output sequence token by token until it generates an end-of-sequence token. However, its ability to only utilize context vectors often leads to information loss and limits the encoder-decoder from generating high-quality summaries [16].

7.1 Long short-term memory (LSTM)

LSTM is an improved version of RNN that mitigates the vanishing gradient problem, enabling it to handle long-term dependencies. The vanishing gradient is an issue that arises when processing long-sequence inputs, where the gradient becomes too small during weight updates, leading to information loss and difficulty in learning. LSTM, however, reduces the impact of this problem through three key mechanisms: the forget, input, and output gates. The forget gate determines which information from the previous time step is unnecessary and to be removed. The input gate, based on the importance of the current input, decides what to remember. Finally, the output gate controls the impact of the stored information on the current output [17].

7.2 Attention mechanism

The basic-attention mechanism, commonly used for tasks like machine translation and text classification, operates through scoring, normalizing, and encoding stages. In the scoring phase, attention calculates the relevancy between tokens using pairwise similarity measures. These scores are then normalized, often using the SoftMax function, to highlight the strong dependencies. Finally, the normalized scores are used to assign weights to the values combined to represent the context of the dependencies and passed to the output.

Due to the effectiveness of this attention mechanism in NLP, several variations have emerged to improve task performance and address various challenges. Multi-dimensional attention uses vector scores in multiple spaces and captures complex input interactions. Hierarchical attention processes information at different levels, from words to sentences and documents, making it useful for summarization. Self-attention, used in transformers, enables models to catch long-range dependencies by calculating attention across the sequence without relying on sequential processing. Memory-based attention, used for more complex tasks, adds an external memory that enables models to access and update stored information. Task-specific attention customizes the attention mechanism to improve performance for specific tasks.

Through such variations, attention mechanisms play many roles depending on the task. In an ensemble, it acts as a voting mechanism emphasizing the most relevant input. In RNNs, it acts as a gating mechanism, dynamically selecting the most relevant information. For pre-training, attention helps to learn contextual embedding by encoding the relationships of inputs considering their context [18].

7.3 Transformer block

The Transformer block, used in pre-trained encoder models like BERT and decoder models like GPT, combines multi-headed self-attention and a fully connected feedforward neural network (FFNN). The multi-headed self-attention mechanism, to assign weights, calculates each element in input against every other element. Its multi-head enables it to pay attention to different tasks using the input. Moreover, its self-attending nature allows it to process these attentions simultaneously. As for FFNN, the output from the attention mechanism is linearly transformed into a representation that can be more easily used for the next steps of the process [19].

7.4 BART

BART is a transformer-based sequence-to-sequence model trained to reconstruct the original text by reversing the effects of artificially introduced noise, predicting it back to its original form. It can be seen as a combination of BERT and GPT models, utilizing the BERT as an encoder for text understanding and GPT as a decoder for text generation tasks. When the original text is corrupted through shuffling sentence order (sentence permutation) and replacing text spans with mask tokens (text infilling), the model performs best in pre-training. In addition, for most generation tasks, BART can use its decoder part directly with fine-tuning applied to its pre-trained encoder. However, for machine translation, it adds a new set of parameters to the encoder. As a result of this flexibility, BART performs highly on many NLP tasks, including text summarization [20].

7.5 PEGASUS

PEGASUS is a self-supervised transformer-based sequence-to-sequence model with a pre-training objective of predicting gap sentences created by masking sentences from the original text. When selecting the sentences to mask to generate the gap sentences, there are three different approaches. The random approach selects sentences randomly, the lead approach selects sentences from the beginning of the text, and the principal approach identifies sentences with the highest importance scores. Within the principal approach, sentences are either selected independently by scoring each sentence and taking the top results or by choosing the best sentence through multiple iterations to maximize the ROUGE-1 F1 scores. From these strategies, as these gap sentences create a pseudo

summary for comparison, the principal sentence selection strategy outperforms the rest. Additionally, due to this pseudo summary generation, PEGASUS is well suited to dealing with low-resource languages that lack labelled data for fine-tuning [23].

8 ATS evaluation metrics

8.1 ROUGE

ATS evaluation is a complicated process that requires unbiased metrics to determine the quality of a summary. A commonly used ATS evaluation metric, ROUGE, measures the similarity between the overlapping words and phrases in the generated summary and the original text without considering the semantic or syntactic. In abstract text summarization, new words are generated for summarization, producing different n-grams than those found in the original text. On the other hand, extractive text summarization uses parts of the original text to create its summary, meaning there is a potential overlap difference between these approaches. However, through experimentation, there is no significant difference between them. Additionally, when combining both summarization approaches, the ROUGE evaluates it with the highest score. However, this can produce incoherent summaries due to over-compression [22].

8.2 BERTScore

BERTScore is an evaluation metric that calculates the similarity between a generated summary and a reference sentence by summing the cosine similarities of their embedding tokens. It is valuable for low-resource language text summarization evaluation due to its reduced reliance on dataset labels. Unlike traditional metrics such as ROUGE that use exact matches, BERTScore utilizes contextual embeddings (provides different embedding values for the same word depending on the surrounding context) from models like BERT to assess semantic similarity. BERTScore uses a greedy matching approach to identify the most similar tokens between sentences (local best). The results are then normalized to match the cosine similarity range, solving the issue of n-gram-based metrics giving lower performance scores for paraphrasing [23].

9 Conclusion

In conclusion, the report researches text summarization techniques and their application to low-resource languages like Mongolian. Through earlier introductions of the base technology, such as NLP pipelines and encoder-decoder models, the report builds the knowledge to learn advanced techniques such as LDA, graph-based ranking, and transformer models like BART and PEGASUS. Additionally, evaluation metrics, like ROUGE

and BERTScore, were discussed, highlighting the complexities and trade-offs dealt with to assess summary quality. Through diverse research knowledge learnt through the report contents, the Mongolian language text summarization task is a step closer to being effectively dealt with. Future research should investigate using existing language corpora as training data or research key feature considerations in creating new datasets for text summarization purposes.

References

[1] M. F. Mridha, A. A. Lima, K. Nur, S. C. Das, M. Hasan, and M. M. Kabir, "A Survey of Automatic Text Summarization: Progress, Process and Challenges," *IEEE Access*, vol. 9, pp. 156043–156070, 2021, doi: https://doi.org/10.1109/access.2021.3129786.

[2] G. Sharma and D. Sharma, "Automatic Text Summarization Methods: A Comprehensive Review," *SN Computer Science*, vol. 4, no. 1, Oct. 2022, doi: https://doi.org/10.1007/s42979-022-01446-w.

[3] W. S. El-Kassas, C. R. Salama, A. A. Rafea, and H. K. Mohamed, "Automatic Text Summarization: A Comprehensive Survey," *Expert Systems with Applications*, vol. 165, p. 113679, Jul. 2020, doi: https://doi.org/10.1016/j.eswa.2020.113679.

[4] M. Shahri and R. Taban, "ML REVOLUTION IN NLP: A REVIEW OF MACHINE LEARNING TECHNIQUES IN NATURAL LANGUAGE PROCESSING," doi: https://doi.org/10.22034/jaisis.2021.306194.1034.

[5] B. B. Elov, S. M. Khamroeva, and Z. Y. Xusainova, "The pipeline processing of NLP," *E3S Web of Conferences*, vol. 413, p. 03011, 2023, doi: https://doi.org/10.1051/e3sconf/202341303011.

[6] C. Toraman, E. H. Yilmaz, ŞahiŋuçF., and O. Ozcelik, "Impact of Tokenization on Language Models: An Analysis for Turkish," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 22, no. 4, pp. 1–21, Mar. 2023, doi: https://doi.org/10.1145/3578707.

[7] S. Choo and W. Kim, "A study on the evaluation of tokenizer performance in natural language processing," *Applied Artificial Intelligence*, vol. 37, no. 1, Feb. 2023, doi: https://doi.org/10.1080/08839514.2023.2175112.

[8] Muhan Na, R. Liu, F. Bao, and G. Gao, "Pre-training Language Model for Mongolian with Agglutinative Linguistic Knowledge Injection," *2022 International Joint Conference on Neural Networks (IJCNN)*, vol. 33, pp. 1–8, Jun. 2024, doi: https://doi.org/10.1109/ijcnn60899.2024.10650812.

[9] W. Chen and B. Fazio, "Morphologically-Guided Segmentation For Translation of Agglutinative Low-Resource Languages," *ACL Anthology*, pp. 20–31, Aug. 2021, Available: https://aclanthology.org/2021.mtsummit-loresmt.3/.

[10] Y. Zhao, H. Hou, J. Ma, S. Sun, W. Chen, and Shi Guodong, "Optimizing Mongolian Abstractive Summarization with Semantic Consistency Enhancement," pp. 1–8, Jun. 2024, doi: https://doi.org/10.1109/ijcnn60899.2024.10651533.

[11] Dhannuri Saikumar and P. Subathra, "Two-Level Text Summarization Using Topic Modeling," *Advances in intelligent systems and computing*, pp. 153–167, Aug. 2020, doi: https://doi.org/10.1007/978-981-15-5400-$1_1$6.

[12] J. Zimmermann, L. E. Champagne, J. M. Dickens, and B. T. Hazen, "Approaches to improve preprocessing for Latent Dirichlet Allocation topic modeling," *Decision Support Systems*, vol. 185, p. 114310, Aug. 2024, doi: https://doi.org/10.1016/j.dss.2024.114310.

[13] R. Elbarougy, G. Behery, and A. El Khatib, "Extractive Arabic Text Summarization Using Modified PageRank Algorithm," *Egyptian Informatics Journal*, Dec. 2019, doi: https://doi.org/10.1016/j.eij.2019.11.001.

[14] R. Mihalcea, P. Tarau, and E. Figa, "PageRank on Semantic Networks, with Application to Word Sense Disambiguation." Available: https://aclanthology.org/C04-1162.pdf.

[15] Shreyas Ghorpade, A. Khan, A. Chaurasia, V. Rao, and A. Chhabria, "A Comparative Analysis of TextRank and LexRank Algorithms Using Text Summarization," *Algorithms for intelligent systems*, pp. 379–393, Jan. 2024, doi: https://doi.org/10.1007/978-981-97-0180-$3_3$0.

[16] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," *AI Open*, vol. 3, Oct. 2022, doi: https://doi.org/10.1016/j.aiopen.2022.10.001.

[17] S. Kumar and A. Solanki, "An abstractive text summarization technique using transformer model with self-attention mechanism," *Neural Computing and Applications*, vol. 35, no. 25, pp. 18603–18622, Jun. 2023, doi: https://doi.org/10.1007/s00521-023-08687-7.

[18] D. Hu, "An Introductory Survey on Attention Mechanisms in NLP Problems," *Advances in Intelligent Systems and Computing*, pp. 432–448, Aug. 2019, doi: https://doi.org/10.1007/978-3-030-29513-$4_3$1.

[19] B. Yang, X. Luo, K. Sun, and M. Y. Luo, "Recent Progress on Text Summarisation Based on BERT and GPT," *Lecture Notes in Computer Science*, pp. 225–241, Jan. 2023, doi: https://doi.org/10.1007/978-3-031-40292-$0_1$9.

[20] M. Lewis et al., "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," Association for Computational Linguistics, 2020. Available: https://aclanthology.org/2020.acl-main.703.pdf.

[21] J. Zhang et al., "PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization," 2020. Available: https://proceedings.mlr.press/v119/zhang20ae/zhang20ae.pdf.

[22] M. Barbella and G. Tortora, "Rouge Metric Evaluation for Text Summarization Techniques," *SSRN Electronic Journal*, 2022, doi: https://doi.org/10.2139/ssrn.4120317.

[23] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "BERTScore: Evaluating Text Generation with BERT," arXiv:1904.09675 [cs], Feb. 2020, Available: https://arxiv.org/abs/1904.09675.