TOWARDS AUTOMATED GENERATION OF OPEN DOMAIN WIKIPEDIA ARTICLES

BY

YUNQIAN BAO

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois Urbana-Champaign, 2023

Urbana, Illinois

Adviser:

Professor ChengXiang Zhai

Abstract

Wikipedia has increasingly become an important resource for education and knowledge acquisition, and the automatic generation of Wikipedia articles could greatly improve the usefulness of Wikipedia.

In this thesis, we aim at extending existing work to ad hoc open-domain Wikipedia article generation and address two limitations of existing work: the first limitation is that the generated articles either lack readability or lack citations for verifiability; the second limitation is that the generated articles only include a flattened list of sections without forming a multi-level hierarchy.

We propose to extend Wikipedia generation to open domain by employing fine-grained entity typing for template generation (generating headings for the article). For template generation, we follow existing work and leverage the headings of similar articles to generate initial headings. But in our work, we identify the importance of entity type and formulated similar articles as articles about entities of the same type with the target entity. We employ ChatGPT for zero-shot Natural Language Processing (NLP) tasks. This further extends our approach to open domain setting. Also, we used ChatGPT for document summarization. By designing appropriate prompts, we are able to generate both readable and verifiable content (which addresses the first limitation mentioned earlier). To address the second limitation, inspired by prior work, we use state-of-the-art topic modeling solutions to enable the hierarchical structure of article content.

The experiment results show that our approach can successfully generate plausible Wikipedia articles while being able to ensure both the readability and trustfulness of the content. Our approach also exhibits certain progress toward the hierarchical organization of content. Still, the generated articles may include artifacts and misplaced information, and the results of subsection generation are preliminary, suggesting that ad hoc open-domain Wikipedia article generation remains a significant challenge.

Table of Contents

Chapter 1: Introduction

Wikipedia is an online collaborative encyclopedia with more than 6.6 million English articles as of July 2023. Over two decades of operation, it has now become the most popular knowledge source in school[1]. For knowledge acquisition, the significance of Wikipedia is irreplaceable: while search engines can retrieve information, they are unable to organize or summarize the information collected; ChatGPT is a recently developed technology that can offer plausible summaries for much knowledge, but its trustfulness cannot be guaranteed.

Behind Wikipedia's effectiveness and success is the numerous collective effort of volunteers. While Wikipedia allows anyone to create or edit an article, the generation of a well-written article is not easy. On one hand, Wikipedia has specified a set of policies and guidelines[1], and they need to be taken seriously to generate high-quality content. On the other hand, a well-written Wikipedia article is often generated over time: an article is not what it looked like upon creation, and it has been continually improved by a multitude of editors; so, it is more like a "convergence" of a series of contributions from the community. Therefore, creating a high-quality Wikipedia article takes a lot of time and effort.

This raises a need for automatic generation (or improvement) of Wikipedia articles. While Wikipedia has millions of articles, many notable entities still do not have corresponding articles. For example, BART (released in 2019) is a widely used language model in the NLP domain, but as of July 2023, it still has not been included in Wikipedia. Automated generation of Wikipedia articles, on the other hand, would make it possible to create these articles ad-hoc. Furthermore, in the digital age, the amount of online resources has been increasing exponentially, and this adds to the necessity of updating Wikipedia articles regularly, while an automatic Wikipedia generation system, if designed properly, would be able to automate the article updating process.

Automatic generation of Wikipedia articles has been studied for over a decade, and there already exist several approaches and systems for this task. These methods produce plausible results for entities. Nevertheless, these methods are domain-specific and have the following limitations: 1) readability and verifiability of the generated content cannot be reached at the same time; 2) the organization of content is limited to a flattened list of sections. These limitations shall be discussed in detail in section 2.1. In this thesis, we aim at open-domain automatic Wikipedia article generation and address these two challenges.

Our contributions are listed as follows:

- We extend automatic Wikipedia article generation problem to open domain– we aim

[1]`https://en.wikipedia.org/wiki/Wikipedia:Policies_and_guidelines`

at generating Wikipedia articles in any domain ad hoc. We proposed approaches to solve the problem and evaluated the effectiveness of the proposed approaches.

- We improve the quality of generated content by ensuring readability and trustfulness without sacrificing either property: we generated content with abstractive summarization while retaining source citations.
- Under the open domain setting, we improve template generation by enabling hierarchical structure of generated Wikipedia articles rather than dividing article contents into a plain list of single-level sections.

In chapter 2, we shall discuss several attempts toward automatic Wikipedia article generation and their limitations, and we will also present a few problems and works related to our task. In chapter 3, we shall present an approach that addresses some of the limitations and extends auto Wiki article generation to open domain. In chapter 4, our method will be tested with entities in several distinct domains. The limitations of our approach and future work will be discussed in chapter 5.

Chapter 2: Related Work

2.1 AUTOMATIC WIKIPEDIA ARTICLE GENERATION

There already exist several works about automatic Wikipedia article generation: a structure-aware approach by Sauper and Barzilay[2], a semi-supervised approach by Pochampally et al.[3], and WikiWrite[4].

2.1.1 Sauper and Barzilay

Sauper and Barzilay[2] proposed a structure-aware approach. They assumed that given the target entity, the relevant information is available on the internet, and they are already provided with a set of documents (along with sections and headings) in the same domain. They generated a template (a list of section headings) by clustering all headings in the given document and choosing the most frequent heading for each selected cluster. The set of chosen headings constitutes the template. Then for each section heading, they search the internet for 10 result pages and extract excerpts from these pages. The excerpts are then ranked and selected with integer linear programming so that the selected excerpts are relevant and have little redundancy. The selected excerpts are the final output for each section, and these sections together form the final generated Wikipedia article.

2.1.2 Pochampally et al.

Pochampally et al.[3] studied a very similar problem setting, except that they specified "a set of documents in the same domain" as a set of Wikipedia articles in a Wikipedia category. A set of headings from these documents is induced, and it is noteworthy that Pochampally et al. ordered the common headings with frequent pattern mining (FP-growth) and topological sorting. Meanwhile, information related to the target entity is collected from the internet at once (instead of making one query for each heading). Leveraging the contents of the training documents (those in the same Wikipedia category), the authors classified the initial headings into two categories and produced contents from the online-retrieved information via supervised learning and unsupervised learning respectively. Nevertheless, Pochampally et al. noticed that not all online information can be correctly classified into these initial headings. For example, Barack Obama is both a U.S. president and a lawyer, and it is unlikely to induce the heading "Legal Career", which is specific to lawyers, from a set of Wikipedia articles about U.S. presidents; so the information about Obama's legal career cannot be classified into any induced heading. So, Pochampally et al. employed topic modeling to cluster the unclassifiable information. All produced contents are aggregated to form the Wikipedia article. Finally, Pochampally et al. employed an entity linking system[5] to link entities in the generated article to existing Wikipedia articles.

2.1.3 WikiWrite

WikiWrite[4] aimed at producing Wikipedia articles for entities that are mentioned (red-linked) in Wikipedia. It is assumed that articles mentioning the target entity are similar to the target entity. The authors created a template from these similar articles and classified relevant web content into sections in the template. A novel part of WikiWrite is its summarization subsystem. While previous work mainly employed text extraction for content generation, WikiWrite further applied paraphrastic summarization to generate novel sentences while ensuring coherence.

2.1.4 Liu et al.

Liu et al.[6] formulated Wikipedia article generation as the multi-document summarization task. They mainly focused on the summarization task and produced a preliminary result for full Wikipedia article generation as a byproduct of their summarization system. They fine-tuned their transformer model to produce the article in an end-to-end manner. So, we do not formally acknowledge their work as a solution to automatic Wikipedia article generation, and their work shall be discussed in section 2.6.

2.1.5 Limitations of Existing Works

As a summary, here is a list of limitations of existing work:

- Domain: the work by Sauper and Barzilay and that by Pochampally et al. both focused on domain-specific Wikipedia article generation, and they needed human labor to determine the domain or the Wikipedia category for the target entity. WikiWrite avoided this issue by leveraging adjacent documents (documents mentioning the target entity) for their approach, but in this way, they limited the target entities to those that are mentioned in Wikipedia, while in practice, we often need to generate Wikipedia articles about newly defined concepts or recently emerging entities, which are unlikely to be red-linked in Wikipedia; further, even if the target entity is indeed red-linked, we cannot guarantee that the entity is mentioned in enough existing articles to produce satisfactory results. Therefore, the domain of current approaches is quite limited, and currently, we are unable to automatically produce Wikipedia articles in arbitrary domain ad hoc.

- Template: all the above existing approaches are able to generate a template (a set of headings) for the target article. The system by Sauper and Barzilay can only produce a static template given a domain; the template generated by WikiWrite depends on neighboring documents, but the template still cannot dynamically adapt to available source information (documents about the target entity). Pochampally et al. further made it possible to generate dynamic headings dependent on the online information collected, but they did not demonstrate their system's capability of generating dynamic headings in their experiments: in their provided example of generated Wikipedia article, the headings are: "early_life", "career", "personal_life", "filmography", and "awards"; and this set of headings is actually a subset of the initial template (headings). And, we cannot replicate their work because Pochampally et al. did not provide enough detail about their approach (see section 3.5). Moreover, in many Wikipedia articles, sections and subsections (and possibly sub-subsections) form a hierarchy rather than being flattened as a plain list of sections. So, the dynamic and hierarchical properties of headings are not well explored in current works.

- Style and citation: Wikipedia has a set of policies and guidelines[2]. These rules, including Manual of Style[3], are ignored in most of the current approaches. Pochampally et al. observed some of the guidelines: they used entity linking to link entities in the generated text to corresponding Wikipedia articles, and they appended citations to the end of each sentence. Still, there exists a "tradeoff" between observance of the citation guidelines and readability: the implementation of citation by Pochampally et al. is trivial because the content is generated mainly by reordering extracted excerpts, but if we were to further improve the contents and readability with abstractive summarization, the reference information could be lost with existing methods. Apart from the citation guideline, there exist many other guidelines that an automatic Wikipedia article generation system should follow to create high-quality results, while current works are quite limited in this aspect.

[2]`https://en.wikipedia.org/wiki/Wikipedia:Policies_and_guidelines`
[3]`https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style`

Note that the above is only an incomplete list of the missing pieces toward a fully automatic generation of Wikipedia articles. Some limitations are identified but not addressed in this thesis, and they shall be discussed later in section 5.2.

2.2 WORKS RELATED TO WIKIPEDIA GENERATION

In section 2.1 we discussed works targeted at generating the body sections of Wikipedia articles. The body sections are not adequate for a full Wikipedia article. According to Wikipedia's Manual of Style/Layout, a simple article should have a lead section (introduction) and references, and a standardized full Wikipedia article, though not required, would contain many other elements: Infoboxes, See also, Further reading, Categories, etc. In this section, we shall briefly cover some other works about generating these elements.

TWAG[7] aimed at automatically generating Wikipedia abstracts (lead section). They viewed the task as the multi-document summarization problem and used extraction and abstraction the approach the problem.

Sáez and Hogan[8] proposed a method to automatically generate Wikipedia Infoboxes from Wikidata. While their method would require that the entity already exists in Wikidata (hence also in Wikipedia), it is possible to incorporate their method with knowledge base population systems to realize fully automatic Infobox generation for entities that do not exist in Wikipedia.

2.3 CHATGPT

In our approach, we extensively use ChatGPT, an LLM launched by OpenAI in November 2022.

ChatGPT can be deemed as a generic solution to many NLP tasks. To resolve NLP tasks, many earlier attempts were training language models or finetuning pretrained models; prompt-based learning makes it possible to resolve NLP tasks by leveraging large LLMs pretrained for other NLP tasks (like QaNER[9]); and now, we can perform zero-shot NLP tasks by designing prompts for ChatGPT.

ChatGPT is trained with reinforcement learning to follow instructions in a prompt. While it may not currently perform well for complex instructions, it can generate plausible results for many well-designed simple prompts. This feature not only enables ChatGPT to accomplish existing fundamental NLP tasks (e.g., NER), but it also grants ChatGPT the capability to deal with many unseen NLP problems.

ChatGPT is trained with huge amounts of data across a broad domain. It is reported that Wikipedia is used as part of the training data. Thus, ChatGPT is fused with knowledge from almost all domains, and hence it has the potential of performing many zero-shot domain-specific tasks.

Admittedly, the capability of ChatGPT is limited, and the performance of ChatGPT is yet under-explored for many NLP tasks. Still, OpenAI is releasing new versions of ChatGPT on a regular basis, and we can expect increasing performance from this chatbot. Therefore, by incorporating ChatGPT in our framework, we can expect that our methodology could produce better results over time.

2.4 FINE-GRAINED ENTITY TYPING

In our approach, fine-grained entity typing is used to extend our method to open domain.

Fine-grained entity typing is the task of predicting the type of entity in a text. It is derived from Named Entity Recognition (NER), which is aimed at identifying named entities and classifying the name entities into several categories. Fine-grained entity typing extended the taxonomy to more than hundreds of types. The FIGER dataset[10] contains 112 types; Gillick et al. released the OntoNotes dataset[11].

Ultra-fine entity typing pushes a step even forward. The UFET dataset[12] contains more than 10,000 types. The emergence of ultra-fine entity typing, together with the advancement of large language models, has inspired new paradigms approaching entity typing: while entity typing is traditionally considered a classification task, many works have been adopting novel ways to approach the task. MLMET[13] prompted masked language model to "fill in" the missing type, and LITE[14] used Natural Language Inference(NLI) to estimate how likely that an entity is of a given type. And it is noteworthy that some works (like LITE) are using natural language representation for the entity types.

2.5 TOPIC MODELING

2.5.1 Mallet

Topic modeling is used by Pochampally et al.[3] to deal with unclassifiable information. They used the Mallet[15] toolbox to discover topics and to assign sentences to topics.

Mallet is a toolbox for many NLP applications like topic modeling and document classification. Its topic modeling module includes a set of algorithms and topic models, such as

Latent Dirichlet Allocation (LDA), LDA's several variations, Hierarchical Pachinko Allocation Model, and word embeddings (word2vec).

2.5.2 BERTopic

In our approach, we shall use BERTopic[16], a more recently released topic modeling technique. BERTopic approached topic modeling by clustering embeddings. While this idea has already been attempted[17, 18], what is novel about BERTopic is that it adopts transformer-based embeddings and proposed class-based TF-IDF. BERTopic has achieved state-of-the-art results and has surpassed LDA and NMF on these datasets: 20 NewsGroups, BBC News, and Trump's tweets.

The author of BERTopic has also released BERTopic as a topic modeling toolbox in Python. Here is the BERTopic pipeline:

1. First, BERTopic generates embeddings for the text. This is typically done with sentence-transformers, but BERTopic also supports a few frameworks (e.g., Flair, Spacy, OpenAI) which make a variety of embeddings accessible.

2. The embeddings are mapped to low dimensional space. This step is crucial to clustering. Vectors are much more likely to be orthogonal in high-dimensional space than in low-dimensional space. Hence, the curse of dimensionality would make it difficult to cluster embeddings, which have high dimensions. The supported algorithms for dimension reduction include UMAP, PCA, SVD, etc.

3. BERTopic runs clustering algorithms (HDBSCAN, k-Means, etc.) on reduced embeddings.

4. BERTopic uses Vectorizer and c-TF-IDF to find representations for the clusters to interpret the topics. In this step, topics are represented with Bag-of-Words and the Bag-of-Words representation is weighted with c-TF-IDF, which is an extension of TF-IDF to cluster of documents, whereas Vectorizer is essentially a counter for n-grams.

5. The final step is optional. Users can fine-tune topics via representation models, prompting generative AI to generate labels for topics, or combining different topic representations. This step further adds to the explainability of the topics.

2.6 DOCUMENT SUMMARIZATION

Document summarization is critical to generating content for Wikipedia articles.

Liu et al. [6] approached the multi-document summarization task with a two-stage process: extractive summarization for coarsely selecting salient information and abstractive summarization for content generation. For the extractive stage, they compared several existing methods (tf-idf, TextRank[19], and SumBasic[20]). For the abstractive model, they proposed a transformer model that drops the encoder, and they further combined the model with memory-compressed attention. The proposed model architecture can deal with long sequences and the whole system is able to extract relevant factual information and produce fluent and coherent text.

Deutsch and Roth[21] studied document summarization in a topic-focused setting. They formulated the summary cloze task that aims at producing the next sentence of a partial summary given reference documents and the topic.

In this thesis, we did not use these methods for extraction or summarization, but we borrow the idea of the two-step approach (extraction and then abstractive summarization) when designing our system.

Chapter 3: Approach

3.1 OVERVIEW

In section 2.1 we have identified a list of limitations of existing methods for automatic Wikipedia article generation. In this thesis, we aim at ad hoc auto Wikipedia article generation and address two existing limitations: in the current work, the template of each generated Wikipedia article is a flattened list of section headings; in currently generated Wikipedia articles, readability and trustfulness cannot be achieved simultaneously.

In our problem setting, we take the surface of the entity as input, and we also assume that we already have a set of documents $\{d_i\}$ containing the entity. We expect the system to output a Wikipedia article with contents organized into appropriate sections and subsections, and all the contents should be paraphrased, readable, and correctly cited.

Note that, unlike most existing works, we take a set of documents related to the target entity as input, as opposed to retrieving the documents from the internet (which is a common practice for many earlier works). We make this assumption for the following reasons:

- Many important documents are not on openly available web pages. For example, for entities in research domains, documents of great significance are often journal/conference papers, book chapters, etc. These documents are mostly pdf files rather than webpages, and they may not be openly accessible.

- Many documents are noisy. The source documents may include symbols, equations, code boxes, HTML tags, etc.; and not all retrieved text is relevant to the target entity (the text might even include advertisements and contents recommended by the website). Instead of exploring different methods and toolboxes to clean various types of noises, we simply clean the noises with human labor.

- Wikipedia ask editors to use information from reliable sources.[4][5] Pochampally et al.[3] did notice a disparity of "goodness" between source documents, and they used a weighted dictionary of websites to address this issue. But their approach relies on training data (Wikipedia articles) in the domain of the target entity, and to migrate their method to the open domain we would need additional efforts. Also, Pochampally et al. simply formulated the "goodness" of a source web domain as its frequency of

[4]`https://en.wikipedia.org/wiki/Wikipedia:Reliable_sources`
[5]`https://en.wikipedia.org/wiki/Wikipedia:Verifiability`

citations in articles of the same Wikipedia category, and this metric does not necessarily imply reliability. For simplicity, we choose to manually decide the sources.

We admit that the above issues should not be neglected towards a fully automated system for open-domain Wikipedia article generation, but in this thesis, we are unable to fully explore these problems. So, these issues are deemed as future work.

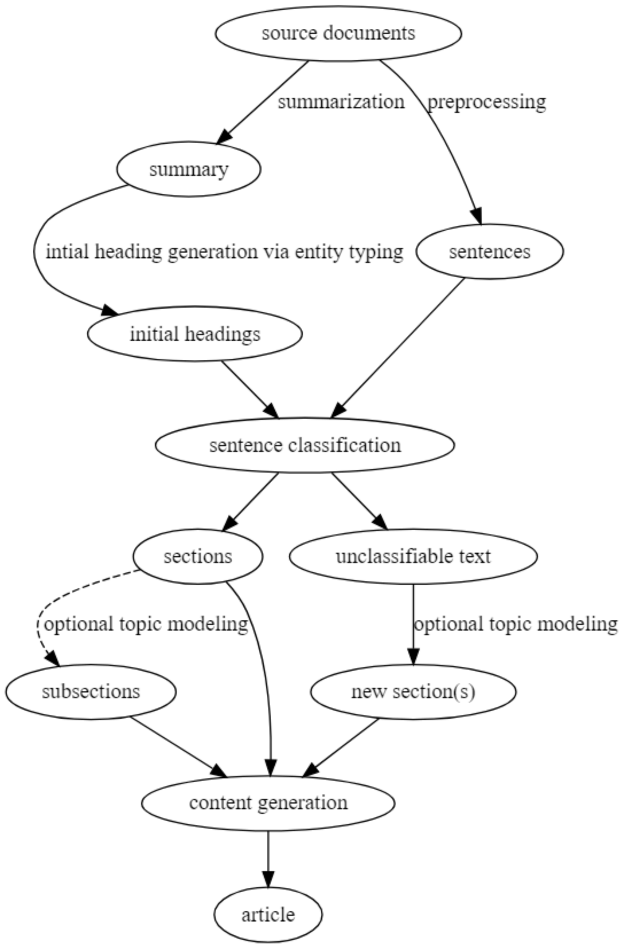


Figure 3.1: The structure of our approach

Now we briefly introduce our approach (Figure 3.1). With the documents that we manually collected, we preprocess the documents (section 3.2), and we generate a list of initial headings (section 3.3). The text will be classified into the initial headings (section 3.4), and for the unclassifiable text and for sections that contain too many sentences, we further cluster the text into new sections and new subsections, respectively (section 3.5). After all texts are assigned to sections and subsections, we generate passages with the text (section 3.6). Finally, we generate the full Wikipedia article (section 3.7).

Note that in these stages, we have used ChatGPT to solve some NLP tasks, and the exact prompts are listed in appendix A.

3.2 PREPROCESSING

In the preprocessing step, for each document, we prompt ChatGPT to do the following:

1. Select excerpts from the document that are relevant to the target entity.
2. If an excerpt contains more than one topic, then segment the excerpt into smaller ones.
3. Paraphrase or summarize the excerpts into complete sentences.
4. Perform entity resolution for each sentence: Leveraging original context, replace all pronouns with appropriate names
5. Output the modified sentences, each on a new line.

The above task is separated into two tasks and we designed two prompts to preprocess the documents with two passes. We do so because the above sequence of instructions is too complex for ChatGPT, and ChatGPT may misbehave with complicated prompts (for example, sometimes ChatGPT would ignore the output format and output all sentences in one line), especially when the input document contains too many tokens.

In the preprocessing stage, salient information extraction is done by extracting excerpts related to the target entity. We choose to extract "excerpts" rather than sentences or paragraphs because: 1) paragraphs may be difficult to identify in noisy input text; 2) paragraphs may contain multiple topics, which could be unfriendly to the classification step (see section 3.4); 3) some sentences have the same topic with neighboring sentences, and it may make more sense to take these sentences together (for example, sentences in a food recipe would only make sense if they are brought together). The extracted excerpts are further processed by being decomposed into smaller ones if they contain multiple topics, and this is because some long sentences in a document may contain multiple topics, which adds to hardship for the future classification step.

Admittedly, extracting relevant excerpts does sound like cheating as we are letting ChatGPT use its own "knowledge" to decide when to segment the text. We also tried sentence segmentation, but by comparing the quality of the articles generated via the two segmentation strategies, we eventually decide that extracting excerpts is overall better than sentence segmentation.

The remaining steps of the preprocessing stage are done to ensure the "independence" of the extracted text: in future stages, we would need to perform sentence classification, clustering, and reordering, so it is crucial that the extracted text can be understood independently of the context. So, we paraphrase the extracted text into complete sentences, and entity resolution is done to prevent the pronouns in the sentence from being mistaken in the dynamic context.

In the experiment stage, we expected that ChatGPT would return a list of sentences, each on a new line, but we observed that some lines might contain more than one sentence. Nevertheless, this does not have much influence on the quality of our generated article. For convenience, we still deem the output as a list of sentences.

Now, for each document d_i we have a set of sentences $s_{i1}, s_{i2}, \ldots$.

3.3 INITIAL HEADING GENERATION WITH FINE-GRAINED ENTITY TYPING

In existing works, an initial set of section headings is derived either from a set of Wikipedia articles that are in the same domain as the target entity[2, 3] or from similar articles (derived from articles mentioning the target entity)[4]. These solutions would either require prior knowledge of the domain/category of the target entity or require that the target entity is mentioned (red-linked) in other articles.

But in our problem setting, all we know about the target entity is simply its surface form and a set of documents containing the entity. And we aim at deriving a set of plausible headings from these inputs alone without any human labor.

A straightforward idea is to infer the Wikipedia category of the target entity from the set of relevant documents, and with the Wikipedia articles in the inferred Wikipedia category, we can produce a set of common headings, just as what has been done in previous works. However, by observation of the existing Wikipedia, we decide that fine-grained entity type is preferable to the Wikipedia category.

A latent assumption in previous works is that articles in the same domain or in the same Wikipedia category would have similar headings. This assumption holds for many domains or categories of named entities, but it would fail for many concepts. For example, as of July 2023, the concept "Dinic's algorithm" has two Wikipedia categories: "Network flow problem" (this category comes first) and "Graph algorithms". While this algorithm "is-a" graph algorithm, it is related to the network flow problem, rather than being an instance of a computer science problem. This is problematic because articles in "Network flow problem" include various problems, algorithms, and theorems, and these articles do not have similar headings: a typical article about a computer science problem may include headings like "Problem formulation" and "Applications"; for an algorithm, headings typically include "Implementation" and "Analysis"; for theorems, we would expect "Definition" and "Proof". This example has two implications. First, we can conclude that Wikipedia categories do not imply hyponymy (is-a relations), and articles could be classified into a specific Wikipedia category because they are within the same domain. Second, in terms of similar headings, compared with the type of entity, the Wikipedia category is not a good solution in the general sense.

Therefore, in our approach, we aim at determining the type of the target entity, and then we leverage Wikipedia articles about entities of that type to determine the initial set of headings. Note that many entities may have more than one fine-grained type (for instance, Donald Trump is both a businessman and a U.S. president), but for convenience, we simply choose an arbitrary type. In our experiments we shall show that choosing only one fine-grained entity type can already produce plausible headings; nevertheless, this might be improved by considering other types, and we leave this as future work.

To solve the entity typing problem, we use ChatGPT rather than the methods mentioned in section 2.4. One reason is that existing fine-grained or ultra-fine entity typing datasets are not enough for our task: UFET[12] has more than 10k entity types, but Wikipedia has 593,125 categories as of the year 2014 [22]; also, we observed that the UFET entity types are insufficient when it comes to scientific domains. Another reason is that only identifying the type, which is simply a label (or a set of labels) in the taxonomy, is not enough as the label alone may cause ambiguity issues (for example, "model" can refer to both "mathematical models" and model as an occupation). We believe ChatGPT is a better solution than existing entity typing methods. ChatGPT has been pretrained on various domains of data, so we expect that it has the capability to identify fine-grained types in much more domains than current methodologies, which are mainly pretrained and finetuned on datasets in some specific domains. Also, we can leverage ChatGPT to provide additional information about the predicted type in order to prevent ambiguity.

We go through the following steps to produce an initial set of headings:

1. We first summarize each document into a sentence about the target entity, and then aggregate all summaries to apply summarization again. The final summary is then fed to ChatGPT to determine a fine-grained type of the target entity. Although there exist many works about fine-grained or ultra-fine-grained entity typing, current taxonomies for entity typing problems are still not fine-grained enough to accommodate with do-

mains in Wikipedia in terms of breadth and depth, so we prompt ChatGPT to leverage the knowledge that it is trained on to produce a free-form phrase in natural language as the representation of the target entity type. During development, we noticed that some free-form phrases may suffer from ambiguity, so we also prompt ChatGPT to provide the domain of the type if necessary (e.g., "model (mathematics)" or "model (person)").

2. Given the entity type (and domain, which is optional), we once again ask ChatGPT to provide five example entities of that type. Then we query Wikipedia for these example entities. Given an example entity, for each of the top $k = 5$ articles returned from Wikipedia's internal search engine (or from Google's search engine, filtering results only from English Wikipedia), we feed the summary of the Wikipedia article into ChatGPT along with the entity type to decide whether the result entity is the example entity that we are looking for (this step is done to eliminate any disambiguation issues in Wikipedia).

3. Now we have a set of Wikipedia articles with respect to the fine-grained type of the target entity. We could follow previous work to generate the set of headings, but for simplicity, we simply aggregate all sets of headings from the articles and prompt ChatGPT to produce the headings.

4. We prompt ChatGPT to sort the headings in a reasonable order. While Pochampally et al. use frequent pattern mining for this purpose since Wikipedia has no general standard for section order[6], we choose not to follow their practice. In the experiments, we shall observe that ChatGPT can sort sections in a fairly reasonable order.

Now we have a set of headings $\{h_i\}$.

3.4 SENTENCE CLASSIFICATION

In the sentence classification stage, we need to classify sentences into appropriate headings produced in the last section. The challenge is that we only have a set of classes represented by headings and sentences to classify. So, we prompt ChatGPT to perform the zero-shot classification task.

Given the target entity, a set of section headings, and a list of sentences, we prompt ChatGPT to return a JSON dictionary in which each key is a heading and each value is a list of sentences relevant to the heading.

[6] `https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Layout#Section_order`

Pochampally et al. identified that some information cannot be categorized into any of the headings[3], so we prompted ChatGPT to assign the unclassifiable sentences to a “NULL” category.

3.5 DYNAMIC HEADING AND SUBSECTION GENERATION VIA TOPIC MODELING

In the sentence classification step, it is possible that some sentences are irrelevant to any of the provided headings (categories). To deal with the unclassifiable sentences, we follow Pochampally et al.[3] and formulate this subproblem as topic modeling.

We did not use Mallet[15] as Pochampally et al. did, nor did we replicate their work for heading generation via topic modeling. This is because Pochampally et al. did not specify which algorithm they used in the Mallet toolbox. Also, compared with the algorithms and models in Mallet (e.g., LDA and hierarchical LDA), the solution provided by the BERTopic toolbox[16] is state-of-the-art for topic modeling.

Thus, we use BERTopic for topic modeling. To recap, BERTopic approaches the topic modeling problem as text embedding clustering, and the pipeline is a sequence of the following modules: (neural) embeddings, dimension reduction, clustering, and topic representation (vectorizers, c-TF-IDF, and optional representation finetuning).

Our configuration of BERTopic is listed as follows:

- Embedding: We tried BERTopic’s default sentence transformer (“all-MiniLM-L6-v2”) and OpenAI’s “text-embedding-ada-002”, and we retrieved similar results in terms of the quality of generated Wikipedia articles.

- Dimension reduction: BERTopic uses UMAP by default. But in practice, we noticed that UMAP would require a minimum of hundreds of input documents (the sentences to cluster, in our case), which is not always available. Also, the runtime of UMAP is much longer than many other dimension reduction algorithms in BERTopic. So, we eventually choose PCA (n_components=5) for dimension reduction.

- Clustering: we use the defaulted HDBSCAN algorithm (min_cluster_size=2) for clustering, and we also enabled automatic topic reduction in BERTopic. Automatic topic reduction merges similar topics and produces a reasonable amount of topics in practice. Since other clustering algorithms in BERTopic (like k-Means) would require the number of topics as input, they are incompatible with automatic topic reduction, so we did not use these algorithms.

- Topic representation: we used the CountVectorizer for the vectorizer, and we further used generative LLM to generate labels for the topics. Following BERTopic's tutorial [7], we prompted ChatGPT to generate a short label for each topic, given the keywords describing each topic and documents in the topic.

While developing and experimenting with our system, we noticed that in many cases the amount of unclassifiable text is too little for the topic modeling algorithm to produce reasonable results, and in some extreme cases BERTopic would report an error if the number of input sentences is too small (like 2-3). We make an assumption that the significance of a topic scales with the amount of relevant text. So, if the number of unclassifiable sentences is smaller than a predefined threshold, we simply aggregate the unclassifiable text into an "Others" section. And we reckon that this measure would not introduce any difficulty for the readers to understand the contents of the generated article.

We used topic modeling not just for the unclassifiable text. For the sentences that can be classified, it is possible that some headings may include too many relevant sentences (we identify this scenario if the number of sentences under a heading is above a predefined threshold). This becomes a problem because we will be using ChatGPT for content generation from sentences, and too many sentences may exceed ChatGPT's limit on input tokens; while this could be resolved by segmenting our prompt (input to ChatGPT) into several parts, the performance of ChatGPT might be deteriorated due to ChatGPT's fixed context length[8].

To deal with this problem, we employ topic modeling to generate a set of topics (subsections) and to classify sentences to the generated subsections.

Note that even with subsections, it is possible that there are still too many sentences in one subsection. While it is not yet implemented in our approach, we can repeat the topic modeling process recursively until the number of sentences is small enough. Since Wikipedia headings follow a six-level hierarchy[9], the recursion should be enough to deal with most cases.

3.6 CONTENT GENERATION WITH CHATGPT

Now, for each section and subsection, we have a list of relevant sentences, and we want to produce readable content from these sentences.

[7] https://maartengr.github.io/BERTopic/getting_started/representation/llm.html

[8] https://platform.openai.com/docs/guides/gpt-best-practices

[9] https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Layout#Headings_and_sections

We first append `<ref name="source"/>` after each sentence, where `source` is a unique string that identifies the source document.

We then prompt ChatGPT to do the following:

1. Reorganize the modified sentences into a coherent passage while keeping the references tags

2. Merge sentences that are duplicated or similar in meaning. When sentences are merged, merge the reference tags as well (by simply appending one reference tag after another).

3. Paraphrase and improve the generated passage to increase its readability.

3.7 FULL WIKIPEDIA ARTICLE GENERATION

The last step is essentially assembling previously produced results to generate a full Wikipedia article.

Wikipedia's manual of style requires that an article begins with an introductory lead section. But in our work, we focus on generating the body sections of the Wikipedia article; also, previous works like TWAG[7] have already achieved significant success in Wikipedia abstract generation. So, we simply use the summary in section 3.3 as the lead section. And we can check from our experiment results that, despite being a "placeholder", the summary is already a good introduction to the Wikipedia article.

The headings of the body sections are the union of initial headings and dynamic headings. For convenience, we simply place the dynamic headings after the initial headings regardless of order. Note that it is possible that the unclassifiable text is so little in the amount that there is no need for topic modeling, and in section 3.5 we simply place these unclassifiable texts in an "Others" section.

All the generated texts are placed in the article following the above structure. Still, we did not specify the order of subsections, and this will be improved in future work.

Finally, a "References" section is added to the end of the article.

The whole article is written in Wikitext, the markup language for Wikipedia article editing. We render the articles with Wikipedia sandbox[10], and the printed version of the generated articles (which look different from what is rendered in Wikipedia's webpages) is shown in the section 4.2.

[10]`https://en.wikipedia.org/wiki/Wikipedia:About_the_sandbox`

Chapter 4: Experiments

4.1 OVERVIEW

In this chapter, we will examine the performance of our approach, and we shall determine whether our method has fulfilled our aim and addressed the limitations that we have previously discussed.

Our goal is ad hoc open-domain auto Wikipedia article generation. This would require that our method produce satisfactory results given an arbitrary entity.

To evaluate our method in this aspect, ideally, we would need to sample vast amounts of entities from a broad range of low-level domains, run our method on these entities, and evaluate generated articles.

Unfortunately, this is impractical:

- In order to assess the generated articles we will need human evaluation. While existing works extensively use the ROUGE[23] for their evaluation, as a metric for summary evaluation, ROUGE alone cannot represent the quality of generated Wikipedia articles.
- The source documents are collected manually (as discussed in section 3.1), so in this thesis we are unable to run the experiments on a large scale.

As an alternative, we provide a few examples. We shall evaluate the quality of generated articles, and we will investigate the intermediate results of our method to assess different components of our system. While this cannot rigorously prove the effectiveness of our system, we can get some insights and identify some drawbacks of our system.

4.2 GENERATED ARTICLES AND INTERMEDIATE STEPS

We generated Wikipedia articles for "Nicole Ameline" (a French politician) and "BART" (language model). In this section, we shall present the generated articles as well as intermediate results.

4.2.1 Nicole Ameline

In this example, the target entity is "Nicole Ameline", who is a French politician. Three documents are retrieved online and labelled as `diplomatie.txt`[11] , `sapgeric.txt`[12] , and `europeanleadershipnetwork.txt`[13], , respectively.

In the initial heading generation step, the three documents are summarized as follows:

> Nicole Ameline is a French politician, diplomat, and jurist who is internationally recognized for her commitment to women's rights and expertise in gender equality.

The entity is typed as "politician", and ChatGPT provided the following examples of the type: "Angela Merkel", "Barack Obama", "Jacinda Ardern", "Justin Trudeau", and "Theresa May".

The system successfully found all Wikipedia articles of these example entities and retrieved section headings:

> Background and early life, Academic career, Early political career ...
>
> Early life, Political beginnings, Opposition, 2008–2015, Prime Minister of Canada (2015–present) ...
>
> ...

Despite the noise in retrieved headings (e.g., "2008-2015"), our model still managed to produce plausible headings for "politician": "Background and early life", "Education and academic career", "Political career", "Personal life", and "Legacy".

The generated article is displayed in figures 4.1 and 4.2. From the article, we can tell that the sentences are correctly classified, except for some sentences in the section "Legacy" (figure 4.2), which should be classified into "Political career".

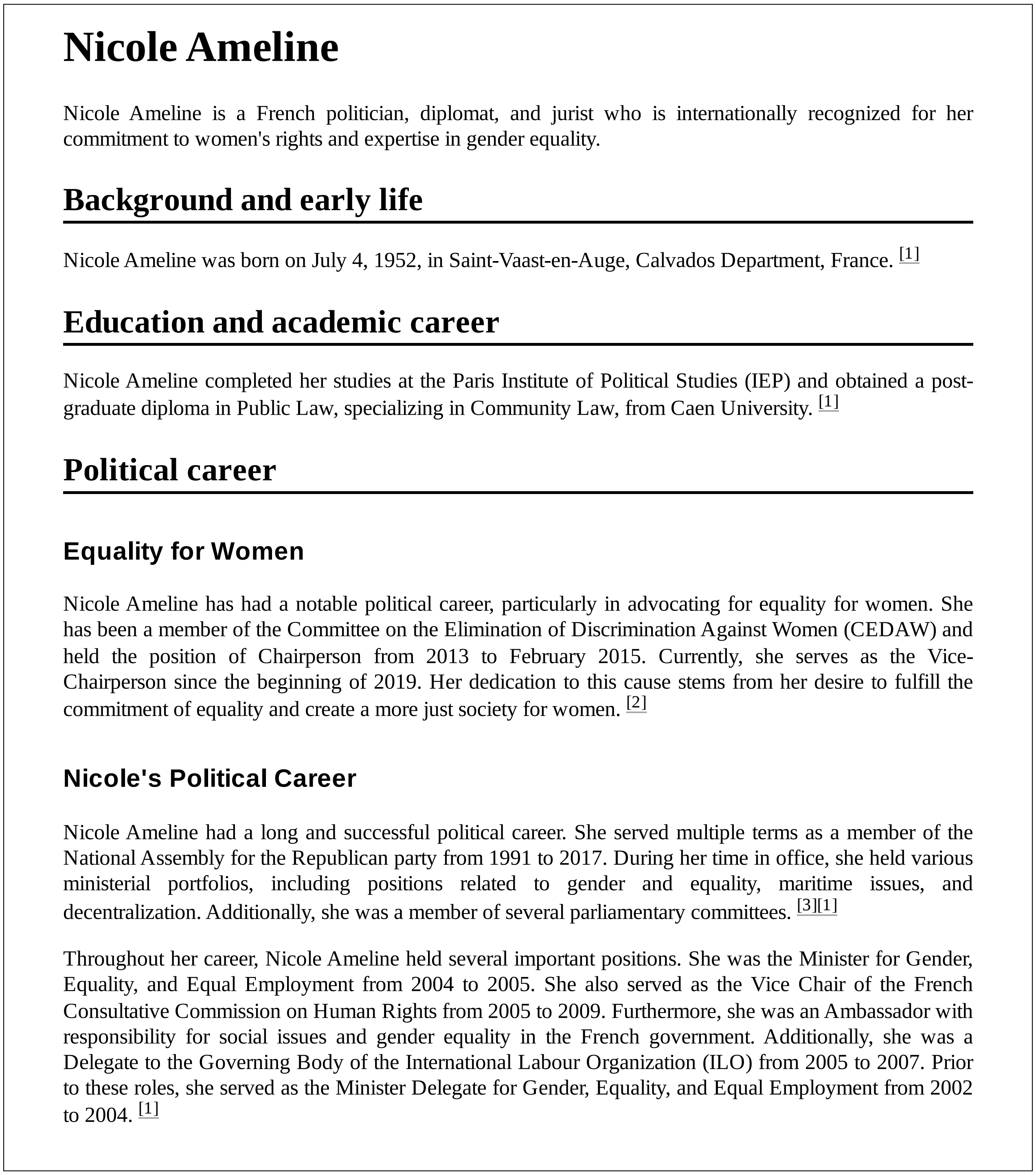
In the topic modeling stage, sentences in the section "Political career" are further clustered into: "Equality for Women", "Nicole's Political Career", 'Women's Rights and Gender Parity", "Women's empowerment and equality", and "Women in NATO Assembly". While the sentences in each subsection are indeed relevant to corresponding subsection titles, we can find much duplication in the semantic meanings of these headings. This is because the number of clusters determined by BERTopic is too large.

For content generation, all passages are readable and the references are cited correctly.

[11]`https://www.diplomatie.gouv.fr/en/french-foreign-policy/united-nations/news-and-events/news/news-2020/article/nicole-ameline-candidate-for-the-united-nations-committee-for-the-elimination`

[12]`http://www.sapgeric.eu2013.vu.lt/wp-content/uploads/2013/10/Mme.-Nicole-Ameline-CV.pdf`

[13]`https://www.europeanleadershipnetwork.org/person/nicole-ameline/`



Nicole Ameline

Nicole Ameline is a French politician, diplomat, and jurist who is internationally recognized for her commitment to women's rights and expertise in gender equality.

Background and early life

Nicole Ameline was born on July 4, 1952, in Saint-Vaast-en-Auge, Calvados Department, France. [1]

Education and academic career

Nicole Ameline completed her studies at the Paris Institute of Political Studies (IEP) and obtained a post-graduate diploma in Public Law, specializing in Community Law, from Caen University. [1]

Political career

Equality for Women

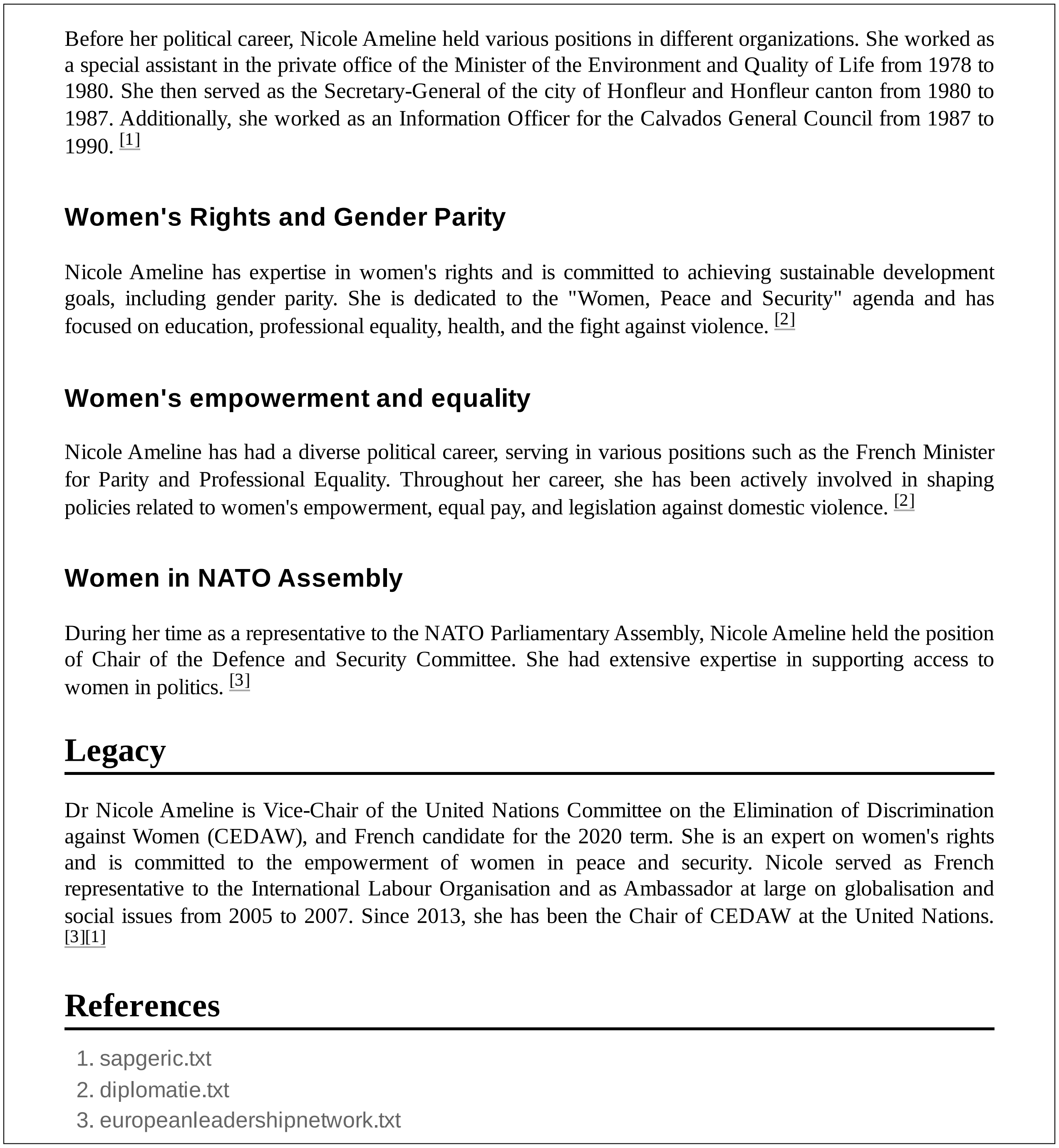
Nicole Ameline has had a notable political career, particularly in advocating for equality for women. She has been a member of the Committee on the Elimination of Discrimination Against Women (CEDAW) and held the position of Chairperson from 2013 to February 2015. Currently, she serves as the Vice-Chairperson since the beginning of 2019. Her dedication to this cause stems from her desire to fulfill the commitment of equality and create a more just society for women. [2]

Nicole's Political Career

Nicole Ameline had a long and successful political career. She served multiple terms as a member of the National Assembly for the Republican party from 1991 to 2017. During her time in office, she held various ministerial portfolios, including positions related to gender and equality, maritime issues, and decentralization. Additionally, she was a member of several parliamentary committees. [3][1]

Throughout her career, Nicole Ameline held several important positions. She was the Minister for Gender, Equality, and Equal Employment from 2004 to 2005. She also served as the Vice Chair of the French Consultative Commission on Human Rights from 2005 to 2009. Furthermore, she was an Ambassador with responsibility for social issues and gender equality in the French government. Additionally, she was a Delegate to the Governing Body of the International Labour Organization (ILO) from 2005 to 2007. Prior to these roles, she served as the Minister Delegate for Gender, Equality, and Equal Employment from 2002 to 2004. [1]

Figure 4.1: Generated Wikipedia article of “Nicole Ameline” (page 1)



Before her political career, Nicole Ameline held various positions in different organizations. She worked as a special assistant in the private office of the Minister of the Environment and Quality of Life from 1978 to 1980. She then served as the Secretary-General of the city of Honfleur and Honfleur canton from 1980 to 1987. Additionally, she worked as an Information Officer for the Calvados General Council from 1987 to 1990. [1]

Women's Rights and Gender Parity

Nicole Ameline has expertise in women's rights and is committed to achieving sustainable development goals, including gender parity. She is dedicated to the "Women, Peace and Security" agenda and has focused on education, professional equality, health, and the fight against violence. [2]

Women's empowerment and equality

Nicole Ameline has had a diverse political career, serving in various positions such as the French Minister for Parity and Professional Equality. Throughout her career, she has been actively involved in shaping policies related to women's empowerment, equal pay, and legislation against domestic violence. [2]

Women in NATO Assembly

During her time as a representative to the NATO Parliamentary Assembly, Nicole Ameline held the position of Chair of the Defence and Security Committee. She had extensive expertise in supporting access to women in politics. [3]

Legacy

Dr Nicole Ameline is Vice-Chair of the United Nations Committee on the Elimination of Discrimination against Women (CEDAW), and French candidate for the 2020 term. She is an expert on women's rights and is committed to the empowerment of women in peace and security. Nicole served as French representative to the International Labour Organisation and as Ambassador at large on globalisation and social issues from 2005 to 2007. Since 2013, she has been the Chair of CEDAW at the United Nations. [3][1]

References

1. sapgeric.txt
2. diplomatie.txt
3. europeanleadershipnetwork.txt

Figure 4.2: Generated Wikipedia article of “Nicole Ameline” (page 2)

4.2.2 BART

In this example, we generate an article for the BART language model. The source documents used are: `huggingface.txt`[14], `paper.txt`[15], and `paper_abstract.txt`[15]. Note that `paper.txt` is the original paper of BART, and the sections “Comparing Pre-training Objectives” and “Related Work”, which are the least relevant sections compared with others, are removed due to ChatGPT’s token limit.

The generated summary of these documents is:

> BART is a model proposed in 2019 that uses a seq2seq architecture with a bidirectional encoder and a left-to-right decoder, achieving ...

The type and domain of BART are predicted as “model (natural language processing)”, and the system suggested the following example entities: “GPT-3”, “BERT”, “RoBERTa”, “XLNet”, and “T5”.

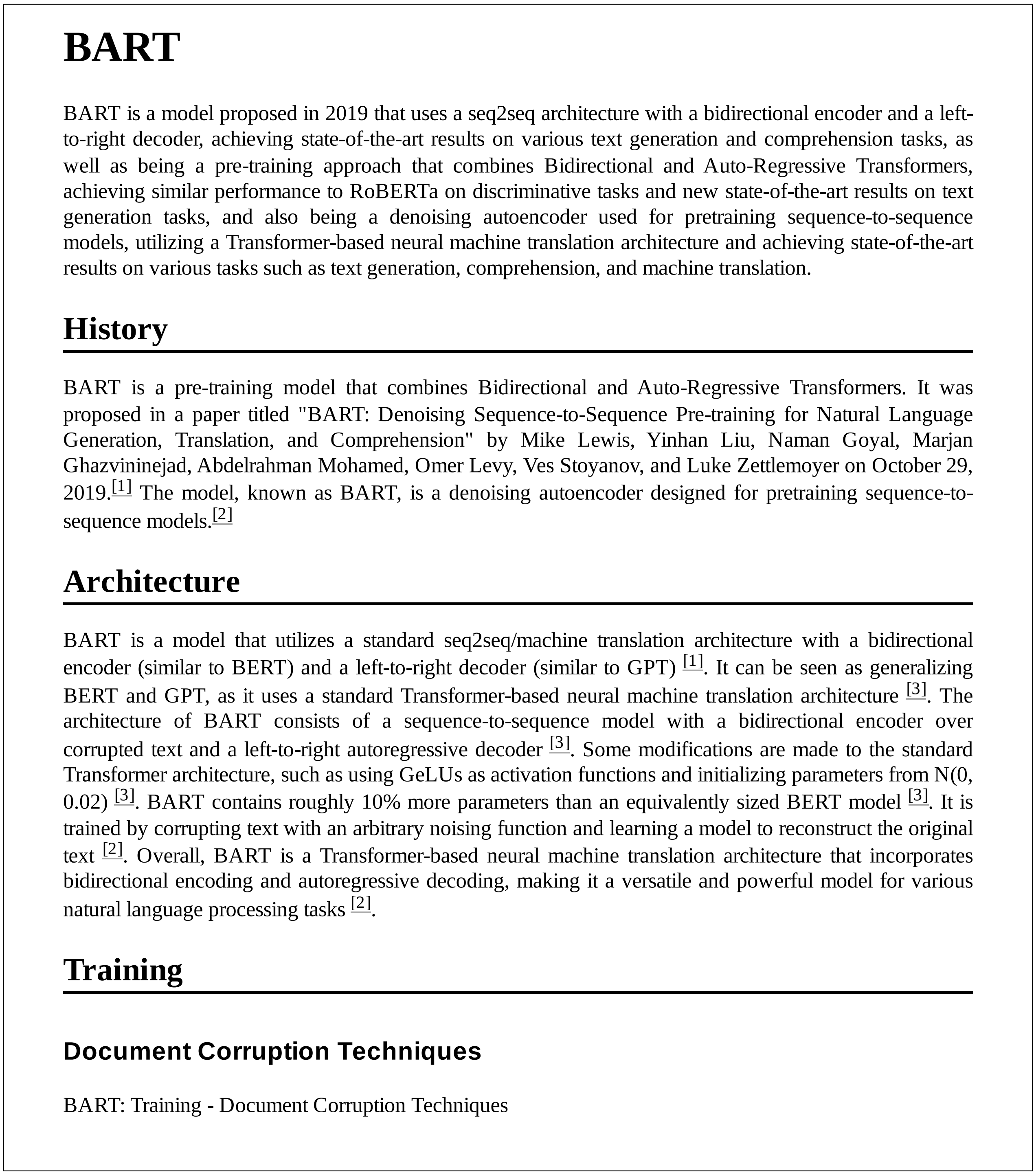
Wikipedia fails to suggest correct articles for “RoBERTa”, “XLNet”, and “T5” because these entities do not exist in Wikipedia. But our system mistakenly resolved two of the nonexisting entities as “Large language model” and “Language model”, which are “compatible” to the predicted entity type. Hence, our system still managed to give reasonable initial headings: “History”, “Architecture”, “Training”, “Application to downstream tasks”, and “Properties”.

The generated article is presented in figures 4.3, 4.4, and 4.5. By examining the contents of sections, we conclude that most sentences are correctly classified. And among the sentences that are not properly classified, most of them are categorized into a “Properties” section, which is not typical of a Wikipedia article about language models. In fact, “Properties” is a heading in the article “Large language model” and it does not exist in any other retrieved articles. Although “Large language model” is “compatible” to the predicted target entity type “model (natural language processing)”, it is actually a subtype, instead of an instance, of the entity type.

For content generation, we can observe that the text exhibits fine readability within each section or subsection. Meanwhile, all the text is cited, and references are correctly merged during the summarization step.

However, we can observe some artifacts in some of the generated text. In the section “Properties” of the generated article “BART” (Figure 4.5), the references were preserved but misplaced at the end of the passage.

[14] `https://huggingface.co/docs/transformers/model_doc/bart`
[15] `https://arxiv.org/pdf/1910.13461.pdf`



BART

BART is a model proposed in 2019 that uses a seq2seq architecture with a bidirectional encoder and a left-to-right decoder, achieving state-of-the-art results on various text generation and comprehension tasks, as well as being a pre-training approach that combines Bidirectional and Auto-Regressive Transformers, achieving similar performance to RoBERTa on discriminative tasks and new state-of-the-art results on text generation tasks, and also being a denoising autoencoder used for pretraining sequence-to-sequence models, utilizing a Transformer-based neural machine translation architecture and achieving state-of-the-art results on various tasks such as text generation, comprehension, and machine translation.

History

BART is a pre-training model that combines Bidirectional and Auto-Regressive Transformers. It was proposed in a paper titled "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension" by Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer on October 29, 2019.[1] The model, known as BART, is a denoising autoencoder designed for pretraining sequence-to-sequence models.[2]

Architecture

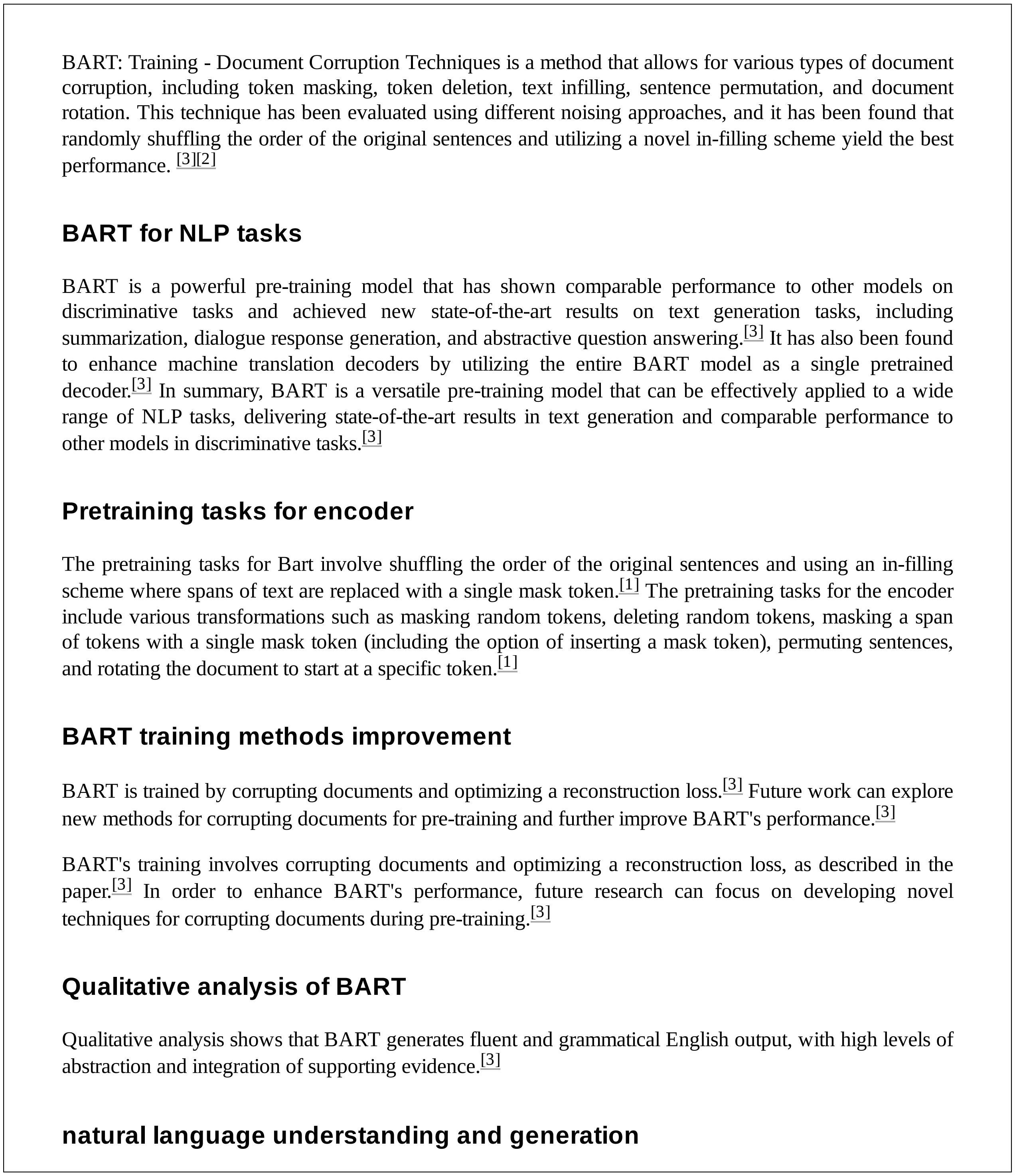
BART is a model that utilizes a standard seq2seq/machine translation architecture with a bidirectional encoder (similar to BERT) and a left-to-right decoder (similar to GPT) [1]. It can be seen as generalizing BERT and GPT, as it uses a standard Transformer-based neural machine translation architecture [3]. The architecture of BART consists of a sequence-to-sequence model with a bidirectional encoder over corrupted text and a left-to-right autoregressive decoder [3]. Some modifications are made to the standard Transformer architecture, such as using GeLUs as activation functions and initializing parameters from N(0, 0.02) [3]. BART contains roughly 10% more parameters than an equivalently sized BERT model [3]. It is trained by corrupting text with an arbitrary noising function and learning a model to reconstruct the original text [2]. Overall, BART is a Transformer-based neural machine translation architecture that incorporates bidirectional encoding and autoregressive decoding, making it a versatile and powerful model for various natural language processing tasks [2].

Training

Document Corruption Techniques

BART: Training - Document Corruption Techniques

Figure 4.3: Generated Wikipedia article of “BART” (page 1)



BART: Training - Document Corruption Techniques is a method that allows for various types of document corruption, including token masking, token deletion, text infilling, sentence permutation, and document rotation. This technique has been evaluated using different noising approaches, and it has been found that randomly shuffling the order of the original sentences and utilizing a novel in-filling scheme yield the best performance. [3][2]

BART for NLP tasks

BART is a powerful pre-training model that has shown comparable performance to other models on discriminative tasks and achieved new state-of-the-art results on text generation tasks, including summarization, dialogue response generation, and abstractive question answering.[3] It has also been found to enhance machine translation decoders by utilizing the entire BART model as a single pretrained decoder.[3] In summary, BART is a versatile pre-training model that can be effectively applied to a wide range of NLP tasks, delivering state-of-the-art results in text generation and comparable performance to other models in discriminative tasks.[3]

Pretraining tasks for encoder

The pretraining tasks for Bart involve shuffling the order of the original sentences and using an in-filling scheme where spans of text are replaced with a single mask token.[1] The pretraining tasks for the encoder include various transformations such as masking random tokens, deleting random tokens, masking a span of tokens with a single mask token (including the option of inserting a mask token), permuting sentences, and rotating the document to start at a specific token.[1]

BART training methods improvement

BART is trained by corrupting documents and optimizing a reconstruction loss.[3] Future work can explore new methods for corrupting documents for pre-training and further improve BART's performance.[3]

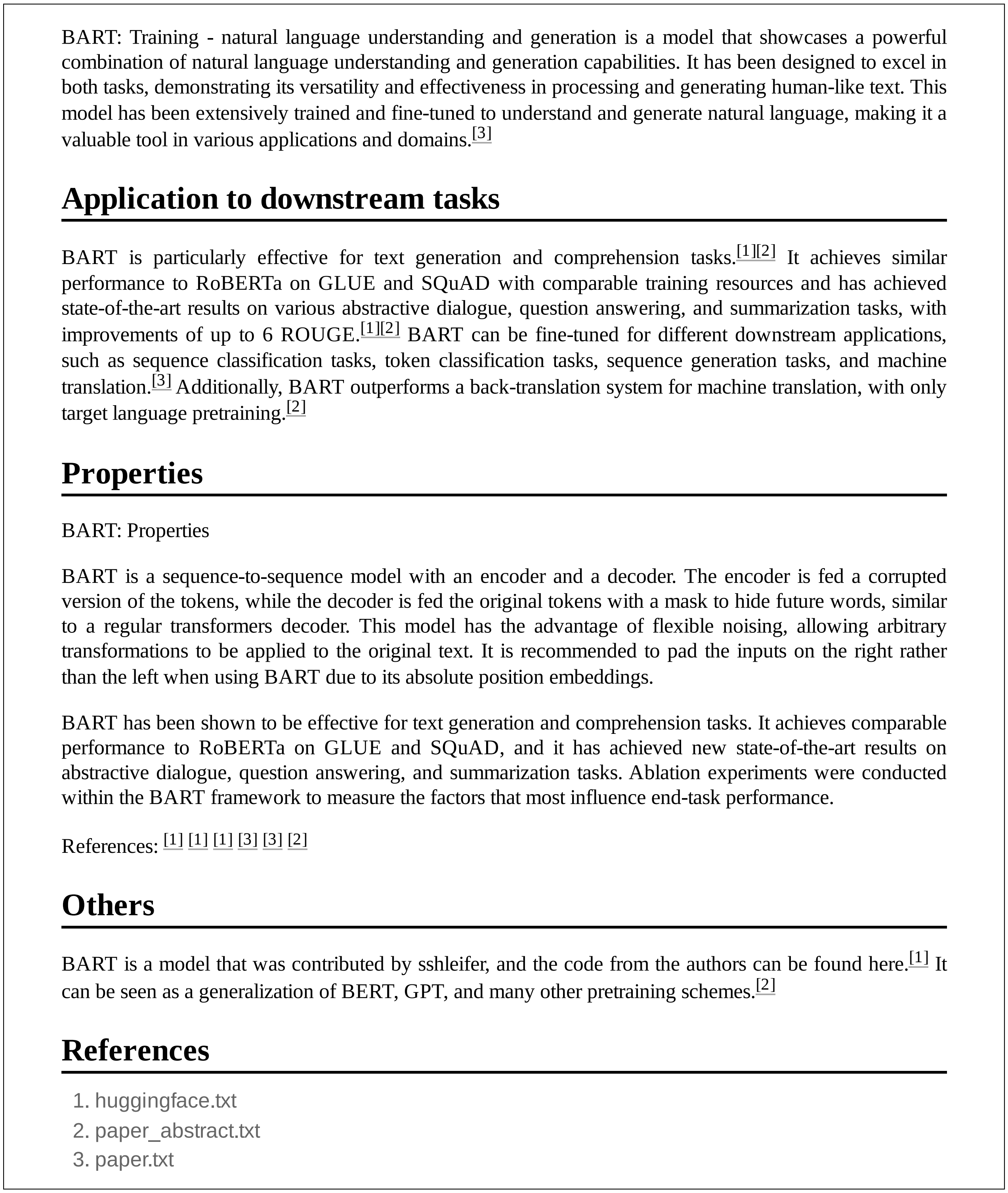
BART's training involves corrupting documents and optimizing a reconstruction loss, as described in the paper.[3] In order to enhance BART's performance, future research can focus on developing novel techniques for corrupting documents during pre-training.[3]

Qualitative analysis of BART

Qualitative analysis shows that BART generates fluent and grammatical English output, with high levels of abstraction and integration of supporting evidence.[3]

natural language understanding and generation

Figure 4.4: Generated Wikipedia article of “BART” (page 2)



BART: Training - natural language understanding and generation is a model that showcases a powerful combination of natural language understanding and generation capabilities. It has been designed to excel in both tasks, demonstrating its versatility and effectiveness in processing and generating human-like text. This model has been extensively trained and fine-tuned to understand and generate natural language, making it a valuable tool in various applications and domains.[3]

Application to downstream tasks

BART is particularly effective for text generation and comprehension tasks.[1][2] It achieves similar performance to RoBERTa on GLUE and SQuAD with comparable training resources and has achieved state-of-the-art results on various abstractive dialogue, question answering, and summarization tasks, with improvements of up to 6 ROUGE.[1][2] BART can be fine-tuned for different downstream applications, such as sequence classification tasks, token classification tasks, sequence generation tasks, and machine translation.[3] Additionally, BART outperforms a back-translation system for machine translation, with only target language pretraining.[2]

Properties

BART: Properties

BART is a sequence-to-sequence model with an encoder and a decoder. The encoder is fed a corrupted version of the tokens, while the decoder is fed the original tokens with a mask to hide future words, similar to a regular transformers decoder. This model has the advantage of flexible noising, allowing arbitrary transformations to be applied to the original text. It is recommended to pad the inputs on the right rather than the left when using BART due to its absolute position embeddings.

BART has been shown to be effective for text generation and comprehension tasks. It achieves comparable performance to RoBERTa on GLUE and SQuAD, and it has achieved new state-of-the-art results on abstractive dialogue, question answering, and summarization tasks. Ablation experiments were conducted within the BART framework to measure the factors that most influence end-task performance.

References: [1] [1] [1] [3] [3] [2]

Others

BART is a model that was contributed by sshleifer, and the code from the authors can be found here.[1] It can be seen as a generalization of BERT, GPT, and many other pretraining schemes.[2]

References

1. huggingface.txt
2. paper_abstract.txt
3. paper.txt

Figure 4.5: Generated Wikipedia article of “BART” (page 3)

4.3 DISCUSSION

4.3.1 Quality of the Generated Articles

We first examine the performance of our approach with respect to the experiments that we have run.

Judging from the section headings, we conclude that our model can generate plausible section headings from the provided documents without other prior knowledge about the entity.

The classifier can correctly classify most sentences into headings. But for headings that are too general or not typical to the entity type, we observe the precision is significantly lower (many irrelevant sentences are classified into that category). Still, we consider the error as acceptable as the classification task is in a zero-shot setting.

Due to a lack of unclassifiable text, the system did not use topic modeling to generate new dynamic headings. But in both examples, the system used topic modeling to generate subsections. However, the only benefit of the topic modeling module seems to be ensuring that sentences in the same cluster are of the same topic. From the example of “Nicole Ameline”, we find that although we enabled the automatic topic reduction feature in BERTopic, the topic modeling algorithm still did not manage to merge topics when it should. And in the “BART” example, we find hardship in comprehending the section with subsections because the subsection headings are not informative enough or compatible with the section heading. The underlying reasons behind this include: 1) in BERTopic’s topic representation finetuning stage, when prompting LLM to produce a short label for topic representation, we did not incorporate entity and section heading information into the prompt, and the LLM only used topic words and sentences in the topic to for subsection heading generation; 2) some sentences are incorrectly classified into the section, hence the generated subsection heading does not fit the section heading.

For content generation within each section and subsection, the overall result is satisfactory. In almost all sections and subsections, the passage is readable and correctly cited.

4.3.2 Effectiveness in Addressing Limitations of Existing Work

Recall that we want to approach ad hoc open-domain auto Wikipedia article generation, and we aim at achieving hierarchical section structure and achieving readability and trustfulness at once.

We first examine the feature of the hierarchical organization of sections. Inspired by

Pochampally et al.'s formulation, we attempted to address the issue by (recursively) employing topic modeling. While we can construct subsections, the quality of subsections is limited and vulnerable to errors in other subsystems. Our effort toward this feature is more preliminary than practical.

In terms of readability and trustfulness, our results are satisfactory. Despite the occasional artifacts that can be easily solved with little human labor, the passages within each section and subsection are readable and verifiable.

We can trivially prove that our system has achieved ad hoc article generation since our system, given relevant documents about the target entity, is able to generate a Wikipedia article in around 3 minutes without any prior knowledge (e.g., domain or Wikipedia category) of the entity.

But regarding open-domain, our work is actually limited. The experiments from two distinct domains show that our approach can produce plausible results, but we have not exhaustively tested our system on every general domain. In fact, we should expect that our system would not work well for some scientific domains because our approach (as well as existing works) is unable to deal with equations and symbols in domains like mathematics and physics; similarly, our method is currently unable to handle algorithms in the computer science domain. Additionally, since we have filtered all relevant information about the entity in the preprocessing stage, our approach may not be able to generate content such that 1) the content is significant to the entity 2) each element of the content alone is irrelevant to an entity. To exemplify, consider a recipe for a dish, or consider the steps in a proof of a theorem. Furthermore, for entities that are closely related to temporal information, our method and current methods cannot produce headings as desired: for instance, the headings in the article "History of the United States" include "Beginnings (before 1630)", "Colonization, settlement, and communities (1630-1753)", "Revolution and the new nation (1763-1815)", etc.; these headings are specific to the entity, and we cannot derive similar headings from similar entities; on the other hand, topic modeling is limited for clustering documents by time.

4.3.3 Robustness of the System

In section 4.3.1 we examined the quality of the generated articles. In this section, we discuss how robust the system is to generate these articles. We will discuss how sensitive the quality of the generated article is to input and choice of hyper-parameters and configurations.

For initial heading generation, we ask ChatGPT to give 5 headings from the reference lists of headings. This number is carefully selected: during the development of our system, we observe that ChatGPT tends to provide as much output as possible, and as we prompt ChatGPT to provide at most k headings, ChatGPT often provides k headings regardless of whether the headings meet our (soft) criteria. If k is set to 10, for example, the headings returned would be of low quality: some headings may overlap with others semantically, while other headings may be too specific but irrelevant to the target entity (this is partly due to some entity-specific headings in the reference headings, like "Prime Minister of Canada (2015–present)" in the article "Justin Trudeau"). The former case may cause sections to have similar contents that are supposed to be merged. The latter case, however, makes it much more difficult to classify sentences into correct headings, as the classifier (ChatGPT) can not determine whether a sentence is related to a low-quality heading. On the other hand, with a smaller k, the coverage of initial headings becomes quite limited.

During the classification stage, the classification results are sensitive to the initial headings. As previously discussed, headings that are specific but irrelevant to the entity would lower the accuracy of classification. Also, the accuracy may also suffer from uninformative headings (e.g., headings that are too general or have ambiguity).

While running our method, there are occasions when the system may fail. The errors are caused by ChatGPT's responses. In our approach, we extensively use ChatGPT to perform NLP tasks. Although ChatGPT's responses are easily comprehensible to humans, in order to incorporate ChatGPT into software we need to carefully design prompts so that the chatbot can produce output in the desired format. With our currently designed prompts, ChatGPT would generally behave as expected, but during development, we noticed that ChatGPT is virtually sensitive to prompt selection. Besides the limitations of the current version of ChatGPT, there may be other factors:

1. The complexity of instruction: some prompts are lengthy due to the complexity of the task, and ChatGPT's performance becomes unstable with complex instructions.

2. The length of input: there are cases where we need to deal with a very lengthy document, and we have observed that ChatGPT is more likely to produce undesirable outputs when processing long documents or input.

Chapter 5: Conclusion and Future Work

5.1 CONCLUSION

In this thesis, we proposed an approach that leverages LLM for ad hoc automatic Wikipedia article generation in the open-domain setting. Our system is able to produce articles without prior knowledge about the target entity except for a set of documents related to the entity. We improved the content of the generated articles by ensuring both readability and verifiability. Furthermore, compared with existing works, we attempted to extend the structure of generated articles from flattened sections to a hierarchy of sections and subsections.

From the experiment results, we conclude that our system is able to generate overall plausible Wikipedia articles. It can generate passages with readability and verifiability. It is able to produce a hierarchical structure for Wikipedia article sections, but it has limited success in producing high-quality headings for subsections and dynamic sections from the source documents. Regarding open-domain, our system can work only with the entity and source documents as input without any other prior knowledge about the entity, but the domains that our system can work well in are still limited.

We further conclude that while our system can produce generally good Wikipedia articles, we still need some human labor to edit the articles for practical use. Our method can be further improved in the following aspects: 1) extend fine-grained entity typing with multiple types and Wikipedia categories; 2) explore more robust initial headings generation from reference headings and design conditioned heading generation strategies for some specific types of entities; 3) improve topic modeling to produce more reasonable clusters and topic representations; 4) improve the use of ChatGPT with better prompts and the chat-completion feature. Beyond addressing the limitations related to our method, there are also many other many unexplored challenges that should be resolved to build a fully automated system for Wikipedia article generation: adding support to various structures and forms of data in source documents and generated articles; retrieving and processing reliable source documents; adding other elements to Wikipedia articles; observing Wikipedia's policies and guidelines in article generation; and supporting automatic updating of Wikipedia articles.

5.2 FUTURE WORK

5.2.1 Further Improvement

- Initial heading generation
  - Fine-grained entity typing: Again, entities of the same type (rather than the same Wikipedia category or the same domain) tend to have similar headings, so it is indeed a good idea to focus on the type of the target entity. But many entities may have more than one type, and the selection of the entity type may have a great impact on the quality of generated headings. For example, Barack Obama is both a politician and a lawyer, and intuitively, classifying this entity as a politician would produce much more plausible headings. To address this issue, we could try to choose the best type from the distribution of topics in the source documents. Or alternatively, we could use all identified types along with the distribution of topics to produce a hybrid list of headings.
  - Wikipedia category: Although we have argued that the fine-grained entity type is more suitable than Wikipedia categories for finding example articles for initial heading generation (see section 3.3), we cannot disregard the usefulness of the Wikipedia category graph. In previous experiments, ChatGPT may suggest example entities that do not exist in Wikipedia. But with the Wikipedia category, we can easily retrieve all articles in that category without prompting ChatGPT to use its own knowledge to make suggestions (we still need entity type and ChatGPT to recheck if the articles in the category are of the target entity's type). If Wikipedia categories of the target entities could be predicted and if we could filter categories that also are entity types, then we could improve the robustness of our system and possibly improve the quality of generated headings.
  - Heading generation from reference headings: In section 4.3.3 we discussed that this step is sensitive to the number of headings that we ask the generative model to produce. Apart from finetuning the prompt for this step, if we could retrieve adequate similar articles with Wikipedia categories (as discussed earlier), it is also worthwhile to perform clustering on all reference headings as done by Sauper and Barzilay[2], and then prompt ChatGPT to generate a heading for each cluster with high intracluster similarity.
  - Conditioned heading generation: In section 4.3.2, we have discussed some entities (those strongly related to temporal information) for which existing approaches

to generate initial headings would fail. For these entities, one possible solution might be: to find out all dates and times from the documents, convert them to timestamps, and then run clustering algorithms on the timestamps, we could then prompt generative AI to produce headings for each cluster. We may then employ a conditioned strategy for heading generation: if we, using entity typing, identify that an entity is strongly related to time, then we can adopt the above solution; otherwise we use the methodology proposed in this thesis. Besides entities related to time, there could be other entities where current approaches for initial heading generation would fail, and we need to differentiate these entities and handle them specially. For instance, subjects and domains (like "Computer Science") also need Wikipedia articles, and the headings for these entities are entity-specific and we cannot leverage similar articles to generate initial headings. It is especially necessary to generate Wikipedia articles for newly emerging domains and subjects, and intuitively these new domains might lack overviews, so we need to aggregate the information from vast amounts of documents in the domain to generate an overview. This task has been investigated in some works like AutoOverview[24], and we may want to incorporate their approach into ours.

- Topic modeling for generating subsections and dynamic sections
  - The number of clusters: we have observed that for BERTopic's automatic topic reduction feature might not merge topics as expected. One possible way to resolve this is to perform an extra topic reduction step: we could try to prompt ChatGPT to further examine the generated cluster labels and determine whether the labels overlap semantically.
  - Ordering of clusters: The ordering of clusters (subsections or newly generated sections) could be resolved by simply prompting generative AI to sort the labels. As an alternative, we can also leverage the order of sentences in the original source documents to infer the order of clusters.
  - The label representation of topics needs to be further improved. In our current system, we only use topic words and a few sentences in the cluster to generate the label (i.e., section or subsection heading). The generative AI producing the label is unaware that the label is used as a (sub)section heading for the Wikipedia article about the target entity (or the target entity along with the parent section). We can hence revise the prompt so that more reasonable headings can be generated.
- In the current implementation, the potential of ChatGPT is yet to be explored. Here

are some aspects of potential improvements:

- The prompts could be further improved to increase ChatGPT's performance and stability. For example, some of the tasks that we ask ChatGPT to perform may still be too complicated, and further efforts are needed to break these tasks into smaller ones and to design simpler and clearer prompts for these tasks.
- We only used the text-completion feature of OpenAI's API, while ChatGPT is a chatbot that is able to memorize the context of the conversation. If we prompt ChatGPT to perform all NLP tasks in one conversation session, then ChatGPT might be able to use contextual information to produce better results.

5.2.2 Under-Explored Problems

Apart from the issues discussed in subsection 5.2.1, we are still far from an ideal system for ad hoc automatic Wikipedia generation. In this section, we discuss some challenges or problems that are not explored in this thesis.

- One under-explored domain is the structure and form of data. When humans create and edit Wikipedia articles, the input (source documents) and the output (generated articles) often contain titles, lists, tables, equations, code boxes, etc. Current methods (including ours) only deal with sentences and paragraphs for both input and output. This limits the quality of generated articles because the models are unable to leverage the information implied by structures (i.e., titles, lists, tables, etc.). Also, the domain of article generation is restricted because, without supporting equations and code boxes, it is impossible to produce high-quality articles for many scientific (math, computer science, etc.) concepts. Furthermore, much important information is multi-modal: for instance, documents often have charts and figures that are crucial for understanding the material; videos like lectures and speeches, which include rich audio-visual information, could also be reliable sources. All these structures and forms of data need to be properly handled to boost the quality of the generated articles and to broaden the domain of the application of Wikipedia article generation systems.
- Retrieving and processing source documents are also challenges. As discussed in section 3.1, the source documents may be noisy, not publicly available, etc., and it is difficult to access or process these documents. Current work mostly retrieves data from openly available web pages, so there is a limited scope of source documents for these systems, and the quality of these system's generated articles may be subject to

the noises in retrieved webpages. The reliability of the sources is also under-explored: the weighted dictionary proposed by Pochampally et al.[3], which counts the frequency of citations of each web domain in Wikipedia articles, is a preliminary effort toward the source selection problem, and we cannot migrate the weighted dictionary to the source reliability problem. Here is an example that the weighted dictionary would fail: consider the personal blog of a person; the blog is unlikely to be cited by many articles (hence gaining a low score in the weighted dictionary), but the blog is a reliable source for generating a Wikipedia article about the person. While we can get rid of misinformation from unreliable sources, we still cannot guarantee truthfulness as information could be outdated, and even reliable sources may conflict with one another. Further efforts need to be done to identify up-to-date information and to resolve conflicting sources.

- There exist many other elements in a Wikipedia article, such as See Also, Infoboxes, etc.[16] While these elements are not required by Wikipedia, a high-quality standardized Wikipedia article usually includes many of these elements. The tasks of generating some of these elements have been studied but not fully explored, and it remains a challenge to generate these elements in the open domain.

- Wikipedia has a set of policies, rules, and style guidelines. Due to the limited progress of current work, many of these rules are often neglected. While observing these rules could be a challenge (for example, Wikipedia requires articles to be based on reliable sources[17], but as prevxiously discussed, the reliability of source documents is not well addressed).

- Beyond Wikipedia article generation, there is also a need for automatic updating of Wikipedia articles (as discussed in chapter 1). It could be straightforwardly solved by regenerating the whole article, but in practice, articles might have gone through a sequence of editing by humans, and we would want to retain the content produced by human editors. So, it would be another challenge to properly adjust Wikipedia article content and organization given newly retrieved sources.

[16]https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Layout#Order_of_article_elements

[17]https://en.wikipedia.org/wiki/Wikipedia:Reliable_sources

References

[1] M. Blikstad-Balas, "“you get what you need” : A study of students' attitudes towards using wikipedia when doing school assignments," *Scandinavian Journal of Educational Research*, vol. 60, no. 6, pp. 594–608, 2016. [Online]. Available: https://doi.org/10.1080/00313831.2015.1066428

[2] C. Sauper and R. Barzilay, "Automatically generating Wikipedia articles: A structure-aware approach," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore: Association for Computational Linguistics, Aug. 2009. [Online]. Available: https://aclanthology.org/P09-1024 pp. 208–216.

[3] Y. Pochampally, K. Karlapalem, and N. Yarrabelly, "Semi-supervised automatic generation of wikipedia articles for named entities," *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 10, no. 2, pp. 72–79, Aug. 2021. [Online]. Available: https://ojs.aaai.org/index.php/ICWSM/article/view/14834

[4] S. Banerjee and P. Mitra, "Wikiwrite: Generating wikipedia articles automatically," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, ser. IJCAI'16. AAAI Press, 2016, p. 2740–2746.

[5] P. Ferragina and U. Scaiella, "TAGME: on-the-fly annotation of short text fragments (by wikipedia entities)," *CoRR*, vol. abs/1006.3498, 2010. [Online]. Available: http://arxiv.org/abs/1006.3498

[6] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer, "Generating wikipedia by summarizing long sequences," *CoRR*, vol. abs/1801.10198, 2018. [Online]. Available: http://arxiv.org/abs/1801.10198

[7] F. Zhu, S. Tu, J. Shi, J. Li, L. Hou, and T. Cui, "TWAG: A topic-guided Wikipedia abstract generator," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, Aug. 2021. [Online]. Available: https://aclanthology.org/2021.acl-long.356 pp. 4623–4635.

[8] T. Sáez and A. Hogan, "Automatically generating wikipedia info-boxes from wikidata," in *Companion Proceedings of the The Web Conference 2018*, ser. WWW '18. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2018. [Online]. Available: https://doi.org/10.1145/3184558.3191647 p. 1823–1830.

[9] A. T. Liu, W. Xiao, H. Zhu, D. Zhang, S.-W. Li, and A. Arnold, “Qaner: Prompting question answering models for few-shot named entity recognition,” *arXiv preprint arXiv:2203.01543*, 2022.

[10] X. Ling and D. Weld, “Fine-grained entity recognition,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 26, no. 1, pp. 94–100, Sep. 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/8122

[11] D. Gillick, N. Lazic, K. Ganchev, J. Kirchner, and D. Huynh, “Context-dependent fine-grained entity type tagging,” *ArXiv*, vol. abs/1412.1820, 2014.

[12] E. Choi, O. Levy, Y. Choi, and L. Zettlemoyer, “Ultra-fine entity typing,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018. [Online]. Available: https://aclanthology.org/P18-1009 pp. 87–96.

[13] H. Dai, Y. Song, and H. Wang, “Ultra-fine entity typing with weak supervision from a masked language model,” in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, Aug. 2021. [Online]. Available: https://aclanthology.org/2021.acl-long.141 pp. 1790–1799.

[14] B. Li, W. Yin, and M. Chen, “Ultra-fine entity typing with indirect supervision from natural language inference,” *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 607–622, 2022. [Online]. Available: https://aclanthology.org/2022.tacl-1.35

[15] A. K. McCallum, “Mallet: A machine learning for language toolkit,” 2002, http://www.cs.umass.edu/ mccallum/mallet.

[16] M. Grootendorst, “Bertopic: Neural topic modeling with a class-based tf-idf procedure,” *arXiv preprint arXiv:2203.05794*, 2022.

[17] S. Sia, A. Dalmia, and S. J. Mielke, “Tired of topic models? clusters of pretrained word embeddings make for fast and good topics too!” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020. [Online]. Available: https://aclanthology.org/2020.emnlp-main.135 pp. 1728–1736.

[18] D. Angelov, “Top2vec: Distributed representations of topics,” 2020. [Online]. Available: https://arxiv.org/abs/2008.09470

[19] R. Mihalcea and P. Tarau, “Textrank: Bringing order into text,” in *Conference on Empirical Methods in Natural Language Processing*, 2004.

[20] A. Nenkova and L. Vanderwende, “The impact of frequency on summarization,” 2005. [Online]. Available: https://www.cs.bgu.ac.il/~elhadad/nlp09/sumbasic.pdf

[21] D. Deutsch and D. Roth, "Summary cloze: A new task for content selection in topic-focused summarization," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019. [Online]. Available: https://aclanthology.org/D19-1386 pp. 3720–3729.

[22] R. Kawase, P. Siehndel, and B. P. Nunes, "Supporting contextualized information finding with automatic excerpt categorization," *Procedia Computer Science*, vol. 35, pp. 551–559, 2014, knowledge-Based and Intelligent Information & Engineering Systems 18th Annual Conference, KES-2014 Gdynia, Poland, September 2014 Proceedings. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050914011016

[23] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004. [Online]. Available: https://aclanthology.org/W04-1013 pp. 74–81.

[24] J. Wang, *AutoOverview: A Framework for Generating Structured Overviews over Many Documents*. Cham: Springer International Publishing, 2020, pp. 113–150. [Online]. Available: https://doi.org/10.1007/978-3-030-41672-0_8

Appendix A: ChatGPT Prompts

This chapter includes ChatGPT Prompts used in our system, and each section corresponds to a stage in our approach.

A.1 PREPROCESSING

Listing A.1: ChatGPT prompts for preprocessing

```
prompt_preprocessing_1 = lambda entity, text: f"""
Given the text delimited by triple backticks, \
select all excerpts that are related to {entity}. \
Excerpts that include multiple sentences or topics should be \
split into shorter excerpts.

For each excerpt, do the following:
1. Paraphrase or summarize the excerpt into complete sentences \
so that each sentence alone is complete and understandable \
without any context.
2. Replace all pronouns (like "he", "she", "it") in the sentences \
to appropriate names.
3. Output the modified the sentences, each on a new line.

In your output, only include the sentences. \
Each sentence is on a new line. \
Do not output anything else.

```
{text}
```
"""

prompt_preprocessing_2 = lambda entity, text: f"""
Given the text about {entity}, delimited by triple backticks, \
```

Listing A.1 (Cont.): ChatGPT prompts for preprocessing

```
identify complete sentences and split the text into \
complete sentences.

For each sentence, replace all pronouns (like "he", "she", "it") \
in the sentences to appropriate names.

Output the modified sentences, each on a new line.

In your output, only include the sentences. \
Each sentence is on a new line. \
Do not output anything else.

```
{text}
```
"""
```

A.2 INITIAL HEADING GENERATION WITH FINE-GRAINED ENTITY TYPING

Listing A.2: ChatGPT prompts for heading generation

```
prompt_entity_type_examples = lambda entity, text: f"""
Given the text delimited by triple backticks, \
determine the type of {entity}.
The type should be as fine grained as possible \
(e.g., doctor is more fine grained than person).

Use a word or phrase for the type, and \
append the domain of the entity after the type.
For example:
model (mathematics)

Then use your own knowledge to provide at most 5 examples \
```

Listing A.2 (Cont.): ChatGPT prompts for heading generation

~~~
of the same type within the same domain \
(not including {entity}).

```
{text}
```


In your output, only include the type and the examples. \
The type is on the first line, \
and the examples are on the next lines, each on one line.

Do not output anything else.

For example, given text about the Pascal (programming language), \
output:
computer programming language
C++
Java
Python
C programming language
Golang
"""

prompt_entity_type_recheck = lambda entity, et, text: f"""
Given the text delimited by triple backticks, \
determine whether {entity} is of the type {et}.
Output "1" if yes, and output "0" if no. \
Do not output anything else.
```
{text}
```
"""

prompt_common_headings = lambda et, temp_headings: f"""
~~~

Listing A.2 (Cont.): ChatGPT prompts for heading generation

```
I am creating a Wikipedia article for an entity of type {et}, \
and I want to determine the headings for the article.

For reference, delimited by triple backticks are \
{len(temp_headings)} lists of headings in Wikipedia articles \
about other entities of type {et}. \
Each line contains a list for an article.

```
{(NEWLINE + NEWLINE).join([', '.join(l) for l in temp_headings])}
```


Ignore headings that exist in every Wikipedia article, \
such as References, External links, and See Also.

Generate (but do not output) a list of at most five short headings \
strongly related to the type {et}. \
Make sure that the headings are simple and very general in meaning.
Again make sure that there are at most five headings.

Sort the headings in a reasonable way.

Your output should only include the sorted headings \
(at most five). \
Each line is the name of a heading.
DO NOT output anything else.
"""
```

A.3 SENTENCE CLASSIFICATION

Listing A.3: ChatGPT prompt for sentence classification

```
_newline = '\n'
```

Listing A.3 (Cont.): ChatGPT prompt for sentence classification

```
NULL_CATEGORY = 'NULL'

_classify_example = """\
{
    "History": [
        "Java was created at Sun Microsystems.",
        "Java was created by a team led by James Gosling."
    ],
    "Syntax": [
        "The syntax of Java is largely influenced by C++ and C. "
    ],
    "NULL": [
        "C++ is a programming language."
    ]
}
"""

prompt_classify_text = lambda entity, et, headings, sentences: f"""
I am creating a Wikipedia page of {entity}, which is a {et}. \
I have a list of sentences about \
{entity} and a list of categories (sections) \
for this Wikipedia article.

The list of categories are delimited by double backticks:
``
{_newline.join(headings)}
``

In the text delimited by triple backticks, each line is a sentence.

For each sentence, do the following:
Determine if the sentence can be classified \
into any provided categories.\
```

Listing A.3 (Cont.): ChatGPT prompt for sentence classification

```
If yes, then classify this sentence \
into the most relevant category; \
classify the text into the dummy category "{NULL_CATEGORY}".

Here is the text about {entity}:
```
{_newline.join(sentences)}
```


In your response, you should return a json dictionary \
where each key is the name of each category \
 and each value is a list of text in strings.

An example response is:
{_classify_example}
"""
```

A.4 CONTENT GENERATION WITH CHATGPT

Listing A.4: ChatGPT prompts for content generation

```
prompt_generate_text = lambda topic, sentences: f"""
I am writing a passage about "{topic}".

In the text delimited by triple backticks, \
each line is a sentence from a source. \
Each sentence ends with a reference, \
which is in the form of "<ref name=xxx>".
```
{NEWLINE.join(sentences)}
```


While performing the following steps, do not \
```

Listing A.4 (Cont.): ChatGPT prompts for content generation

```
change the format of any references. Now do the following:

First, reorder the sentences along with their corresponding \
references into a coherent passage.
Do not remove the references after each sentence.

Second, if sentences contains duplicate information, \
merge sentences, and merge their references \
by moving the unique references to the end of the merged sentence.
For example:
1. I like apple. <ref name="1"/>
2. I like banana. <ref name="2"/>
3. I like pear. <ref name="1"/>
can be merged into:
I like apple, banana, and pear.<ref name="1"/><ref name="2"/>

Third, without changing references, \
improve and paraphrase the passage to make it \
more coherent and readable.

Finally, output the passage (with references after each sentence). \
Do not output anything else.
"""
```