Universiteit Leiden

Master Computer Science

Reference attribution in AI-powered voting guides

Name: Bram van Kooten

Student ID: s3738574

Date: 25/04/2025

Specialisation: Artificial Intelligence

1st supervisor: Peter van der Putten

2nd supervisor: Suzan Verberne

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Acknowledgements

I would first like to thank Stiching Open Politiek, for allowing me to enhance their existing project for my thesis, as well as giving me the opportunity to test my hypotheses on real end-users during the European Parliament elections. Special thanks go to Floris Hoogenboom, who was my contact at Stichting Open Politiek and helped me greatly with any questions I had.

I also want to thank my supervisor dr. Peter van der Putten. Your guidance was of great help steering my thesis to its final form. I also greatly appreciated our talks about current affairs during our meetings. I also want to thank my second supervisor, Prof. Dr. Suzan Verberne, for their valuable insights and constructive feedback

Finally, many thanks to my parents, who always stood by me during this rather long graduation process.

Abstract

This thesis introduces AI powered reference attribution in the online voting guide from openverkiezingen.nl. OpenVerkiezingen is a Retrieval Augmented Generation based system that summarizes the views of political parties based on the information in their party manifesto. To generate the references, an LLM based solution is designed to fit within the ecosystem of OpenVerkiezingen. To test the effect of adding references on user behavior, an AB-test was performed. This showed an increase in the number of users that interacted with the system, as well as an increase in the number of interactions. In order to further verify the performance of the chosen reference generation method, a comparison is made with a number of alternative methods. From this comparison, we found that the baseline of BM25 outperforms the other models. Further investigation of the results of individual parties provided more insight in which of the chosen methods work better given certain writing styles. In general, it seems that references are more easily identified correctly if the manifesto text is divided into clear (sub-)topics, and has a higher information density.

Contents

1. Introduction

Every so often, elections are held in the Netherlands. For a large majority of the population, the period around the elections is the only time when people actively engage with politics [1]. In order to help people figure out which party to vote for, there are many different types of voting guides, such as StemWijzer by ProDemos[1] or Kieskompas[2]. These voting guides generally consist of a number of curated statements in which the user is asked if they agree or disagree with the statement. After answering all the statements, the given answers are compared to the answers given by the different political parties, resulting in an ordered list based on the similarity to these parties. Although this methodology provides a decent method that helps its users find the political party that best suits them, it does have a number of shortcomings.

The main shortcoming relates to the fact that political parties provide their own answers to the different statements, usually on a 5-point scale (strongly agree, agree, neutral, disagree, strongly disagree). These answers are generally accompanied by an explanation to justify the given answer. Although this answer should reflect the views of the party as mentioned in their manifesto, the answers are generally not linked back to any sources and do not include any references to the party manifestos. This makes it difficult for voters to verify or gain more in-depth knowledge of the positions provided by the parties. It also means that, for parties that have very similar positions, it is often hard to compare statements between these parties. It is also possible that two parties both disagree with a statement, but the reasons why they disagree are completely different. An example of this can be found during the 2023 Dutch elections. When asked about storing CO2 underground, both PVDAGL and PVV disagree for different reasons. PVDAGL thinks that storing CO2 is a counterproductive way of combating the climate crisis, while the PVV believes that no money should be spend on CO2 reduction.

In addition to this, the fact that the statements are selected in advance also limits the information that a user can get. While the statements reflect a general view of what topics are important, this might not properly represent the main issues that any given

[1] ProDemos voting guide for European Parliament elections: https://eu.stemwijzer.nl/
[2] Kieskompas voting guide for European Parliament elections: https://eu.kieskompas.nl/nl/

user wants to base their vote on. It also does not allow the user to easily get more in depth questions answered in the topics of their liking.

One approach for dealing with these shortcomings is to take the answering of the statements away from the parties. This can be done by using a Large Language Model (LLM) to answer the statements for the political parties. However, using an LLM as is makes it difficult to verify the factual basis of the claims made. This is solved by using a well-known augmentation for LLM's, namely Retrieval Augmented Generation (RAG) [2]. In RAG, each LLM query is augmented with a number of sources that are relevant to the provided query [3, 4]. Using RAG means that the generated information will always be linked back to provided documents, such as the party manifestos of the different political parties. Using RAG also has the added benefit that it is possible to provide the source on which the generated result is based [4]. This approach is being used by Stichting Open Politiek to make openverkiezingen.nl. Here, it is possible to provide a statement and select a number of political parties to see what they have to say about this topic. For each statement, the three most relevant sections from a party's manifesto are retrieved, and instructions are given to the LLM to answer the statement using the information in these sections. This summary can then be easily compared to the other selected parties. Similar research exists for different European countries. In Belgium, UGent researchers created "KamerRaad", which summarizes relevant parliamentary reports related to user queries [5]. For the European Parliament elections in 2024, Chalkidis researched whether the stances of different political parties can be predicted on the "EU and I"[3] voting assistance questionnaire [6].

Although this approach already addresses part of the main shortcoming, it still leaves some room for improvement. The current solution already helps to narrow down the relevant content from the party manifesto, but it does not yet point to the precise claims and facts that the generated summary is based upon. In LLM and RAG literature this task is called Claim Verification [7] or Attributed Question Answering [8]. Certain parts of the relevant sections might be extremely relevant to the answer, while others might not be at all. In order to improve upon this, we have augmented the existing system created by OpenVerkiezingen with references that attribute generated sentences with their sources. We have achieved this by prompting the RAG model to provide the source sentences the generated text was based on. This enhancement has been deployed on openverkiezingen.nl, where we have tested if this addition improves the experience of its users and whether it leads to more interaction with the source information of the political parties.

[3] "EU and I" voting guide for European Parliament elections: https://euandi.eu/

This initial approach of generating references is simple and straightforward, using RAG prompts to find the source sentences. However in a political climate that gets more and more polarized every day, it is important that the generated references are accurate. Inaccurate references might decrease confidence in politics, leading to even more polarization. In order to find the best way to generate references, we designed five additional methods that we compare against our first approach in terms of performance. Because there currently are no datasets available to benchmark this kind of application, we have created a dataset consisting of a number of political statements of which the resulting summaries, as generated by OpenVerkiezingen, are provided with references. The six methods are scored on how accurately they can identify the correct references. To avoid political bias, the results are also analyzed on a per-party basis.

In this thesis, the aim is to address the following research questions:

1. Does adding references to AI generated summaries increase the number of people who interact with the sources of these summaries?
2. Does adding references to AI generated summaries increase the number of interactions with the sources of these summaries?
3. What methodology is best for generating references used in AI generated summaries?

The remainder of this thesis is organized as follows. Chapter 2 reviews related research in the fields of Large Language Models, Referencing, as well as the combination of Natural Language Processing and Politics. Chapter 3 provides insight in the workings of OpenVerkiezingen and goes into the decision-making process for how our RAG based enhancement should function. Chapter 4 details the implementation of this RAG based enhancement. Chapter 5 describes the experiments performed both for user behavior on OpenVerkiezingen, as well as for finding the best method for generating references. In Chapter 6 the results for these experiments will be shown, and they will be discussed further in Chapter 7 along with future avenues of research. Finally, Chapter 8 will give some closing remarks and conclusions.

2. Related Work

In this chapter, we discuss a number of research fields that are closely related to our research or are the basis for the methods used. This includes Large Language Models, methods for verifying information for references, and avenues that use Natural Language Processing in a political context.

2.1 Large Language Models

Modern large language models are reliant on Transformers [9]. The revolutionary addition to the transformer compared to traditional models are the self-attention layer. These self-attention layers compute the relation between a given word and all the other words in the entire sequence [9]. These relations can be used to better understand the importance of the combination of words, even when words are far apart in a sequence [9]. This Transformer is an encoder-decoder model [10]. As the name suggests, this model is split up into two parts, an encoder and a decoder. The encoder processes the input into a vector, after which the decoder takes this vector and creates a prediction of the desired output. An example of this is translating a sentence from Dutch to English.

Later, a decoder-only model was proposed, namely the Generative Pre-trained Transformer (GPT) [11]. A decoder only model has the ability to generate text given a prompt. The input can also be adapted to fit a specific task, which allows for fine-tuning with minimal changes to the architecture of the model [11]. When the generative models get larger, they become stronger and show emergent abilities [12, 13]. The first of these larger generative models is GPT-3, which is considered to be the first Large Language Model (LLM).

An issue that LLMs face is hallucinations. An hallucination is used to refer to generated content that is unfaithful to the input or non-sensical [14, 15]. These hallucinations can be further divided into two types: intrinsic hallucinations and extrinsic hallucinations [16]. Intrinsic hallucinations occur when the generated text conflicts with

the source information. Extrinsic hallucinations occur when outputs can't be verified through the provided information or external knowledge bases.

One way of dealing with hallucinations is to enhance the LLM with Retrieval Augmented Generation (RAG). In RAG, the pre-trained knowledge from the LLM model is supplemented with knowledge retrieved from an external source [2, 4]. Before prompting the LLM, the input provided by the user is used to retrieve documents relevant to the input. These documents are then combined with the input to supplement the information available to the LLM before generating an answer. This also ensures that any information used by the LLM is up to date.

Another added benefit of using RAG is that it ensures that the provided data is grounded [2]. This guarantees that the data used to generate by the LLM is all relevant [2]. This gives high confidence that the most up-to-date version of information is used instead of an older version that was also in the training data, and it allows for data that was not yet available before the training cutoff to be used.

2.2 Referencing

There are, however, a number of different tasks that can be used (sometimes with a small adaptation) to perform the referencing task. The tasks we will discuss here are Claim Verification [7], Attributed Question Answering [8], and Natural Language Inference [17]. Of these avenues, Claim Verification will be discussed in more detail, as the relevance is greater for this thesis.

In **Claim Verification**, a given claim is checked on a corpus of information in order to find if the claim is true or not [7]. This task is generally performed in the following three steps: document retrieval, sentence selection, and label prediction [7, 18–20]. In the document retrieval step, the whole set of all documents in the corpus is reduced to the ones that are most relevant to the given claim. Within these documents, sentence selection is performed to find the most relevant sentences in these documents. These sentences are then finally compared to the claim, predicting a truth stance, or label, about the claim.

One of the tasks that uses claim verification is on the SciFact dataset [7]. This dataset is a collection of scientific claims, accompanied by abstracts that either support or refute the claim. Pradeep et al. use the following setup [18]. For document retrieval they first use BM25, after which they rerank their retrieval using T5 [21]. T5 is a sequence-to-sequence language model. However, T5 is sometimes used in an encoder-only setting, in which case it functions similarly to BERT [22]. For sentence retrieval, they again use

T5, but now on a sentence level instead of a document level. This ranks the sentences from the documents in terms of relevance to the claim. Finally, for label prediction, they fine-tune a larger T5 model (T5-3B). Their proposed pipeline outperforms the state-of-the-art solutions [7].

Hanselowski et al. created a pipeline for claim verification on a different task [19]. They tested their pipeline on the Fact Extraction and VERification (FEVER) shared task [23]. FEVER consists of a large number of claims that are manually verified against the introductory sections of Wikipedia pages. The claims are then labeled as Supported, Refuted, or NotEnoughInfo [23]. For document retrieval, they propose an entity linking approach. In entity linking, entities (locations, organizations, persons) are extracted from the text in the claim, which are then linked to the documents in the corpus [24]. In this case, the entities are linked to the titles of Wikipedia articles. Since multiple documents might be retrieved for any entity, they filter the articles, discarding any documents that do not have sufficient overlap with the topic of the claim. For sentence selection, they use ESIM [25], which takes as input a claim and a sentence to generate a ranking score. The evidence set consists of the five highest-ranked sentences. In order to label the claim, Hanselowski et al. propose an extension of ESIM, which allows them to predict the entailment relation between multiple input sentences and the claim. Their results show that retrieving more documents and sentences has a positive effect on both accuracy and recall [19].

Another claim verification framework for the FEVER task is proposed by Soleimani et al. [20]. They have modeled their three-step pipeline as follows. For document retrieval, they follow the method proposed by UKP-Athene [19]. For sentence selection, they opt to use a BERT model [22]. They finetuned their model using two different approaches, one pointwise approach and one pairwise approach [20]. Additionally, they experiment with Hard Negative Mining, which is used to retrieve a limited number of negative samples, while not using trivial examples [20]. They do this because in the FEVER dataset the ratio of negative (non-evidence) and positive (evidence) sentences is high. Finally, for claim verification, Soleimani et al. train a new pre-trained BERT model as a three-class classifier. When comparing the performance of their proposed solution, we see that they achieve a better recall than the state-of-the-art but lack behind in both precision and F1 score [20]. They do note that recall is the most important statistic for this type of task, because the sentence retrieval predictions are the samples on which the verification system is trained. Their best performing solution also managed to rank second at the time on the blind test results from the official FEVER framework [20].

In **Attributed Question Answering (AQA)**, the focus is not only on verifying that a given answer is correct, but it combines the answering of questions with providing

a source for this answer [8]. The general use case for AQA takes as input a question, which is combined with the top search results from the web to form answer, attribution pairs [8, 26, 27]. Next these pairs are scored to find the best supported answers. For this task, Menick et al. use Reinforcement Learning to fine tune the Gopher LLM [28]. In their approach, the authors still supplement the self-supported question answering with human ratings to prevent the production of hallucinated facts. Deng et al. opt for an NLI based approach, using the mT5 model [29]. Their system also combines the work of human annotators and LLMs. The human annotators manually extract useful information which the LLM bases its candidate summaries on. These candidates are then refined by the human annotators again.

Natural Language Inference (NLI) is the problem of determining whether a hypothesis can reasonably be inferred from a premise [17]. This means that the NLI model used can determine whether a given hypothesis agrees, contradicts, or is neutral about the given premise. NLI can be used in different contexts. One of these contexts is finding fake news [30]. Sadeghi et al. test a large number of machine learning models to classify the truthfulness of claims made in news articles. For these models, they compare a simple version without supplemented information with an NLI version that is supplemented with information from trusted news sources [30]. They find that the NLI models outperform the simple ones in all but one case [30].

Laurer et al. instead use NLI as a classification task [31]. In their paper, they perform tests on eight different datasets with a political context. To convert the classification task into an NLI task, they replace the hypothesis with statements regarding the label, like 'It is about democracy' [31]. The NLI model of choice is BERT-NLI. The authors find that BERT-NLI outperforms classical models when there is little data available, while still achieving similar results to these models on larger datasets [31].

These three different avenues highlight different ways to conclude which source is relevant for a provided statement. For Claim Verification, the focus lies on finding information that confirms or contradicts the provided statement [7]. Attributed Question Answering is more focused on combining the correct source with the answer [8]. Natural Language Inference is used to determine whether a hypothesis can be inferred from a given premise [17].

2.3 NLP and Politics

Within Natural Language Processing, a number of avenues related to politics are already more widely researched. One of these topics is stance detection. The purpose of

stance detection is to identify the stance of the text author towards a target explicitly mentioned or implied within the text [32, 33]. This is applied to multiple different facets related to politics or politicians. Kuhn et al. apply stance detection to analyze political discourse [34]. Bergam et al. analyze language used by the US Supreme Court to investigate the extent to which the Court's public-facing language is political. It can also be used to help detect fake news [36, 37].

The party manifesto is a great source of information about a political party. It can tell something about the ideology of the party, as well as the topics that are most prevalent at the time of writing. Bielik makes an analysis of social democratic parties in Central Eastern European countries [38]. Here, they analyze both the most prevalent topics as well as the sentiment towards these topics. These topics and their sentiment are then compared to find if certain topics are more common in certain political situations (like the effect of incumbency). Orellana and Bisgin perform similar analysis to political parties in New Zealand [39]. They also compare the similarity of party manifesto's over time.

Some research has also been conducted that uses LLMs in a political context. In Belgium, UGent researchers created "KamerRaad", which uses RAG to summarize relevant parliamentary reports related to user queries [5]. For the European Parliament elections in 2024, Chalkidis researched whether the stances of different political parties can be predicted on the "EU and I"[1] voting assistance questionnaire [6]. These last two papers are the closest to the research we perform in this paper but still lack the attribution element that makes up the largest contribution of this research.

[1] "EU and I" voting guide for European Parliament elections: https://euandi.eu/

3. Methods

In this chapter, we discuss the inner workings of OpenVerkiezingen, as well as a high-level overview of the generation of references using RAG. The five alternative methods used for comparison will be discussed in Section 5.2.

3.1 OpenVerkiezingen

When visiting `openverkiezingen.nl`, end users are greeted with the view shown in Figure 3.1. On the top part, they are able to select up to three political parties to compare. In the middle, they are able to provide a statement of their own choosing on which they want to compare these parties. Finally, in the bottom, the summaries for the chosen parties are shown together with the three sources used to generate these summaries.

FIGURE 3.1: Overview of OpenVerkiezingen platform.

In order to get a better grasp of how the addition of reference generation works, an explanation of the inner workings is required. This process is split into two parts. First,

there is the processing of the party manifestos. Secondly, there is the generation of summaries based on the statements provided by the users.

When processing the manifestos, their original versions, which normally are in PDF format, are first converted to XML. This allows the text to be easily split into smaller fragments. This splitting occurs between the headings in the text. These smaller fragments are then turned into embeddings. These embeddings will return similar results for text fragments that have a high semantic similarity. The process of splitting and embedding the fragments is performed by OpenVerkiezingen. The details of which method and tooling is used for splitting/chunking and embedding is a black box to us. At this point, the system is ready to process user queries.

On OpenVerkiezingen, users are asked to provide a statement they want to know more about, as well as a number of different parties from which they want more information. Whenever a user provides a statement to OpenVerkiezingen, this statement is also converted to an embedding. During the retrieval part of RAG the text fragments that are semantically closest to the provided statement are found, keeping the three most relevant sources. Next, the LLM is provided with these sources and asked to summarize what the given party says about the topic in their manifesto. This result is returned and displayed to the user, along with the relevant sources. This process is repeated for all parties selected by the user.

3.2 Reference Generation

While using RAG helps provide some additional confidence in the overall information used to generate an LLM response, it does not provide any guarantees as to what information the response is actually based on. In the case of OpenVerkiezingen, while providing the most relevant sources gives more confidence in the overall sentiment of the summary, it does not give any direction on a sentence-to-sentence basis. One way to fill this knowledge gap is to add references to the summary text. These references can provide more information on a more in-depth level, both within the structure of the summary, as well as for the sources provided.

In order to generate the references, we explored a number of different methods to find the solution that fits best for the given context. Two of these avenues will be elaborated upon, namely, generating the summary with references in a single prompt, and splitting the task of generating the summary and the references into multiple separate prompts.

Single prompt

The first method consists of adapting the task of summarization to also include instructions for generating the references while the summary is being created. Intuitively, this has a number of advantages. The first advantage is that it saves on costs. Having to do multiple LLM calls can get expensive quickly, thus limiting the number of prompts eliminates this concern. Besides this, allowing the system to add the references at the time the summary is generated provides greater confidence with regards to if the reference is related to the generated text fragment.

However, this approach also comes with a number of notable downsides. Firstly, in order to have the method properly function, the sources have to be adapted before being passed to the LLM. In the original OpenVerkiezingen implementation, the sources are passed as is, without any adaptation. This lacks the information required for the LLM to add a reference, as it does not have a real index to reference to. Because of this, the source has to be split on a sentence level and then numbered. This allows the LLM to reference the number to put in the generated text. Secondly, compared to other methods, very few references were actually generated. Observed behavior generally showed one or two references being generated, and at most a single reference per summary sentence. This does not give enough feedback for the user to accurately assess where the provided information can be found in the sources, especially for sentences that take information from multiple places within the sources. Finally, this method makes it hard to retrieve the references after the fact, which is an obstacle when trying to visualize the information back to an end-user.

Multiple prompts

A different approach is to split up the generation of the summary and the generation of the references. This means that the summaries are generated as is, and an extra prompt is created to find the references the summary is based on. This is done on a per-sentence basis, providing the references for each sentence if they are found. The main advantage of this approach is that it gives a lot of control. The new prompt can be constructed that provides the new information in any way that is desired. The LLM can, for instance, provide a JSON object with the references as full sentences, which allows for an easy way to visualize the results. This also means that the entire process does not interfere with the generation of the sentences. However, the main disadvantage has to do with the cost of the approach. Adding an extra prompt per sentence greatly increases LLM usage, also taking into account that all sources have to be fully provided for every prompt.

Decoupling the reference generation might also decrease the confidence in the relation between the reference and the text fragment.

In the end, together with OpenVerkiezingen, a decision was made to further research the approach with multiple prompts. This decision was made mainly because this approach does not interfere with the summarization process.

4. Implementation

In this chapter, we will explain in more detail the approach we took to implement the generation of references for the provided summaries.

4.1 Generating references

After the summary is provided by the system, a number of steps are taken to prepare for the generation of references. Firstly, the most relevant sources (that were also used by the LLM to generate the summary) are collected, as these are required to let the LLM know what pool of sentences the references could be based upon. Next, the summary gets split up into sentences, since we are interested in the references on a sentence-by-sentence basis. These two elements are then combined into a single prompt, which gives instructions on what is expected from the LLM. The chosen LLM is GPT3.5 Turbo from OpenAI.

This prompt is split up into two messages: one with the system role, and one with the user role. The full prompt can be found in Appendix A. The system role provides the system with information on what is expected from the response (Appendix A line 2-42). These expectations can be seen mostly as limitations for what a reference should look like. One important aspect is the fact that a reference should consist of a single sentence within the source material. This way the LLM does not mark an entire piece of the text as a reference, which would defeat the purpose of generating the references. It is however, possible for a sentence to have multiple references, thereby not limiting the information that can be linked to a generated sentence. Next, the system message is pivotal in ensuring that the response provided by the LLM is in a correct and easy to use format. The decision is made to have the data returned as JSON data, as this allows for easy use in the steps taken afterwards, with regards to processing the responses. In order to ensure that the response is in JSON format, both a sentence is provided asking for JSON, as well as providing multiple examples of what the response should look like.

```
Voorbeeld:
Zin: In het verkiezingsprogramma wordt voorgesteld om vermogen eerlijker te belasten door een aparte vermogensbelasting in te voeren op vermogens in box 2 en box 3 boven een miljoen, met een hoger tarief boven de twee miljoen.

Antwoord:
{
    "zin": "In het verkiezingsprogramma wordt voorgesteld om vermogen eerlijker te belasten door een aparte vermogensbelasting in te voeren op vermogens in box 2 en box 3 boven een miljoen, met een hoger tarief boven de twee miljoen.",
    "references": [
        "We gaan vermogen eerlijker belasten.",
        "We voeren daarnaast een aparte vermogensbelasting in op vermogens in box 2 en box 3 boven een miljoen, met een hoger tarief boven de twee miljoen."
    ]
}
```

LISTING 4.1: Example of expected response from LLM when using prompt for generating references.

One of these examples is shown below in Listing 4.1. The provided example is in Dutch, as we are dealing with Dutch party manifestos and a mainly Dutch audience.

When looking at this example, we can see that multiple, single-sentence references are generated for the provided sentence. The response also returns the sentence the references are linked to, again to make processing the responses easier. While in this example multiple references are generated, it is also possible that the system determines that a sentence in the summary does not find its basis in one of the sources. This might for instance be the case if the sentence is introductory to the provided prompt. In this case, we don't want the LLM to return any references. To accomplish this, another example is added which does not have any references. This example can be found below in Listing 4.2.

Next, the second message of the prompt is with the user role (Appendix A line 44-51). This is the point where the relevant sources, as well as the sentence we are trying to generate references for are provided to the LLM. It does this by giving brief explanations as to what is being provided. It states that the sources are from a political manifesto, and each source is separated by some white space. It also states that the provided summary sentence is based on the provided sources. Finally, it gives a shortened instruction about the type of task the LLM should perform.

```
Voorbeeld :
Zin: Deze partij is tegen de stelling "Nederland moet uit de
   Europese Unie".

Antwoord:
{
    "zin": "Deze partij is tegen de stelling "Nederland moet
   uit de Europese Unie"",
    "references": []
}
```

LISTING 4.2: Example of expected response from LLM which does not have any references.

5. Experiments

In this chapter, we will explain the experiments that are performed for our research, both with regards to generating references and to finding out which approach yields the best references.

5.1 OpenVerkiezingen

Keeping in mind what the goals are for OpenVerkiezingen, adding references to generated summaries helps the user digest the party manifestos provided by the political parties. It does this by pointing out the most important passages from the most relevant sources and chapters. Although the information on which the summaries were based was already available for the users to read, it is possible that this information would consist of three very long chapters from the party manifesto. Because the user can get provided with a large wall of text, this might dissuade them from actually diving deeper into the content of the party manifesto.

When looking at research question 1, our hypothesis is that the addition of references can help increase the number of users that actively interact with the source material provided. we believe that this will both be the case because users click on the references to be redirected to the given passages, but it will also increase the number of users that open the provided source text drawers. This is because the marked passages provide indications as to what the most important passages related to the given statement are.

To test this hypothesis, we performed an AB test on openverkiezingen.nl. For this test, the control group are users running the already established version of the platform, without references. The test group are users running the version of the platform that includes references. An overview of how both groups look can be seen in Figure 5.1. In order to keep track of how the users interact with the platform, a number of events were created to count the interactions with the summaries and the sources. For both the control and test groups, the number of times a drawer with source material was opened was tracked. For the test group, the number of times a reference was clicked on within

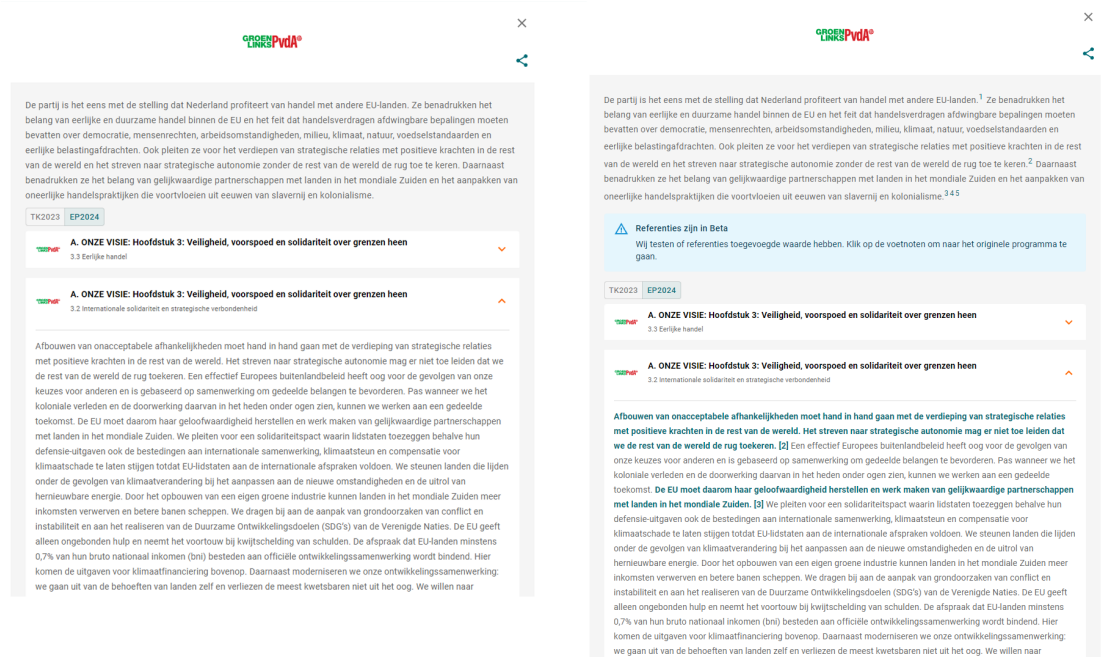

FIGURE 5.1: Comparison of the control group without references (left), and the test group with references (right) on OpenVerkiezingen.

the summary was also tracked. Clicking on a reference opens the corresponding drawer with source material and scrolls to the passage that the summary sentence refers to.

We use the following metrics to test the above hypothesis: Firstly, the number of users that interact with the system (clicking on a source drawer or reference) is compared between both groups. This provides insight into whether the references entice users to interact with the system. Next, we also address research question 2. To address this question, a comparison is made between the average number of interactions of the users that interact with the system. An increased number of interactions means that users are more likely to take a deeper dive into the source material. In order to check if the number of source drawers clicked also increases, both tests are also performed with the clicking on references excluded.

5.2 Best references

Besides finding out whether the references improve the overall experience for the users on the platform, it is also good to compare what is the best solution to generate these references. The initial solution was chosen to fit within the ecosystem provided by OpenVerkiezingen, but it might very well be the case that a different solution performs a lot better for this task.

Party	BBB	CDA	CU	D66	DENK	FVD	JA21	NSC
Amount	14	21	20	20	16	8	28	15
Party	PVDAGL	PVDD	PVV	SGP	SP	VVD	Volt	
Amount	28	23	14	12	11	20	21	

TABLE 5.1: Distribution of data based on what party the statement is from.

Because there is no data available to test this type of reference generating on, we have created a set of curated data based on the summaries and sources provided by OpenVerkiezingen. This data can be found on the GitHub page of this thesis.[1] In order to eliminate bias as much as possible, the statements used to generate the summaries are chosen from an existing voting guide[2]. The voting guide consists of 30 statements, tailored for the Dutch parliamentary election in 2023. The summaries for these statements are generated for every statement, for each of the 15 political parties currently in parliament. To further eliminate bias, 150 party and statement combinations were chosen at random to select references for. From any given summary, one or multiple of the summary sentences are selected, and linked to sentences from the sources that best represent what the summary is based on. This results in a total of 271 data points to compare methodologies with. The party distribution of these statements can be found in Table 5.1.

The following methods were chosen to compare. The implementation and paramaters used for each of the methods can be found on the GitHub page of this thesis.[1]

1. Deployed LLM solution as used in AB-test on OpenVerkiezingen. This test uses GPT-3.5-turbo-0125.

2. The same LLM solution, but using GPT-4-turbo-2024-04-09.

3. A Natural Language Inference based solution. Since NLI is a task that is adjacent to generating references, it might be applicable to it as well. When checking if a sentence can be inferred, it is important to note that, in general for our use case, the summary sentence is built up of of multiple sentences. Because of this, we should check if the source sentence can be inferred from the summary sentence, and not the other way around. Another noteworthy thing for NLI, is the fact that the model returns three scores per pair, one for contradiction, one for entailment, and one for neutral. Since we are only interested in whether or not there is entailment present for the pair, we discard the other values. The chosen NLI model is a cross-encoder from SBERT [40], namely nli-roberta-base.

[1] https://github.com/bramvankooten/Master-Thesis
[2] https://tweedekamer2023.stemwijzer.nl/

4. A passage retrieval cross-encoder. Since another adjacent field, namely Claim Verification, uses passage retrieval to find the passages that are most relevant to the given claim, we can adapt this approach to reference generating. Because we don't expect a negative truth stance to be found by our source text and the generated text based on this source text, we can reason that the most relevant passages can be used as references for our summary sentence. Since we are only interested in the most relevant sentences, and not for instance the three most relevant, a relatively high threshold is set. This avoids scenarios where relatively general sentences get assigned references. The model used is another SBERT cross-encoder [40], this time pre-trained on MS MARCO passage retrieval. The specific model is ms-marco-MiniLM-L-2-v2.

5. A BM25 retrieval approach. Since passage/document retrieval is a well established field, it is good to compare to one of the most established model in the field. In order to allow this approach to work, we denote the summary sentence as the search query, and denote each individual paragraph sentence as a document. Just like with the passage retrieval cross-encoder, we set a threshold to only denote the most relevant sentences as references. To run BM25, the bm25s Python package was chosen.

6. An embeddings based retrieval bi-encoder. In order to better bridge the gap between the more established approaches and the LLM approach, an embeddings based approach is adopted. In this approach, all the summary sentence and all source sentences are converted to embeddings, which are then compared based on similarity. A threshold is set to select which sentences are relevant for the given passage. The model used is e5-large-v2.

6. Results

In this chapter, we present the results from our experiments. We will break down the user interaction that occurred on OpenVerkiezingen and what method is best at generating references.

6.1 OpenVerkiezingen

The AB-test was run in the period right before the European Parliament election of 2024, from the 25th of May till the 6th of June. In this period, a total of 4292 unique users visited the website. Of this group, 3651 users were put in the control group, while 641 users were put in the test group. In order to test the hypothesis both when references are included and excluded, the results for the test group are also calculated when looking only at the clicks on the source panels, thus excluding the clicks on references. The full results can be found in Table 6.1.

Group	**Size**	**Interact**	**Ratio (p)**	σ_p	μ	σ
No references	3651	682	0.18680	0.01492	2.50147	2.30128
References (excl)	641	109	0.17005	0.03598	2.72477	2.37998
References (incl)	641	122	0.19033	0.03554	2.97541	2.62683

TABLE 6.1: Results gathered from AB-test performed on openverkiezingen.nl. This includes 3 rows: the control group (No references), the test group with references excluded (References (excl)), and the test group with references included (References (incl)). The different columns mean the following: Size is the size of the group, Interact is the number of unique users that interacted with the system, Ratio (p) is the ratio of user that interact with the system, σ_p is the standard deviation of p, μ is the average number of interactions with the system, and σ is the standard deviation of μ.

When looking at the results, we can see a number of different things. Firstly, when looking at the ratio of users that interact with the system (Ratio (p)), we see that this ratio decreases when the references are excluded, but increases when they are included. For both test groups, we can also see that the average number of interactions (μ) for the users that interact with the system increases in both cases.

In order to calculate if the results are significant, we perform a t-test. In order to do this, we need a standard deviation (σ). For the number of interactions this is just a simple calculation given the results. For the ratio of users, we use the formula for the standard deviation of a proportion in a population $\sigma_p = \sqrt{\frac{p(1-p)}{n}}$. The resulting values can also be found in the results table.

First, we look at the number of users that interact with the system. All the relevant values for these t-tests can be found in Table 6.2. For both the groups we find p-values close to 0. In the case that we exclude the references, as we found a drop in the interaction ratio, we can confidently conclude that overall less users click on source panels when references are shown on the screen. However, when the references are included, since we found an increase in the interaction ratio, we can confidently conclude that more users interact with the system when references are included.

	$\bar{x}_1$	σ_1	n_1	$\bar{x}_2$	σ_2	n_2	t-value	df	p-value
References excl	0.170	0.036	641	0.187	0.015	3651	-19.996	4290	0.000
References incl	0.190	0.036	641	0.187	0.015	3651	4.239	4290	0.000

TABLE 6.2: Values for t-tests regarding the number of users that interact with the system. $\bar{x}_i$ denotes the mean of sample i, σ_i denotes the standard deviation of sample i, and n_i denotes the size of sample i.

Next, we will look at the number of interactions. All the relevant values for these t-tests can be found in Table 6.3. When looking at the calculations excluding clicking on references, we find a slight increase in the number of interactions. However, with a p-value of 0.175, this increase is not large enough to conclude that the number of interactions increases. When we look at the calculations including clicking on references, we find a higher increase in the number of interactions. This results in a p-value of 0.020, meaning that we can say with reasonable confidence that the observed increase in interactions is significant.

	$\bar{x}_1$	σ_1	n_1	$\bar{x}_2$	σ_2	n_2	t-value	df	p-value
References excl	2.715	2.380	109	2.501	2.301	682	0.939	789	0.175
References incl	2.975	2.627	122	2.501	2.301	682	2.049	802	0.020

TABLE 6.3: Values for t-tests regarding the number of interactions with the system. $\bar{x}_i$ denotes the mean of sample i, σ_i denotes the standard deviation of sample i, and n_i denotes the size of sample i.

6.2 Best References

Next, we look at the results from the experiment for comparing different methods for generating references. The results can be found in Table 6.4. Overall, we find that the

Model	Precision	Recall	F1
BERT	0.625	0.631	0.590
BM25	**0.672**	0.762	**0.679**
NLI	0.372	0.319	0.327
E5	0.393	**0.830**	0.468
GPT	0.424	0.661	0.487
GPT-4	0.649	0.655	0.641

TABLE 6.4: Average precision, recall, and F1-score for different methods of generating references across all instances.

baseline solution of using BM25 scores best, yielding the highest precision and performs near the top result for recall. This is also reflected in the F1 score being the highest. We also clearly find that the NLI approach performs the worst by far, placing last in all three metrics.

The largest discrepancy between precision and recall can be found using the E5 model. While it does manage to score a very high recall, this is at the cost of its precision. In order to check if this discrepancy can be reduced, a precision-recall curve is created across the range of similarity scores for the embeddings. The resulting plot can be found in Figure 6.1. The range of thresholds tested is between 69.11 and 95.85, which are the lowest and highest similarity score observed.

The curve looks different than what one might suspect in a normal precision-recall curve, showing a drop in the precision after the recall drops below a certain value. This is due to the fact that the precision calculations generated a lot of NaN values. This happens for precision when none of the source sentences where marked as being a reference, and therefore giving a division by 0 error. When the threshold increases above a certain point, this happens for most instances of the data. To still give a proper representation of the performance in these cases, the decision was made to include these instances in the average precision calculation, setting their value to 0.

When looking at the graph, we find that the precision maximizes at 0.651, at which point the recall is 0.604. Thus, the threshold value that yields the best precision still does not improve upon the precision of the baseline, while also losing a large amount of recall.

To check if there is any further discrepancy in the data we created, an analysis is made of the performance for the different models when the data is split according to the different parties. Below, a selection of the results will be shown and discussed. Full results for every party can be found in Appendix B.

First we take a look at the results from NSC (Table 6.5). Here we find that all methods produce a result that outperforms the average. This the case for both the precision and

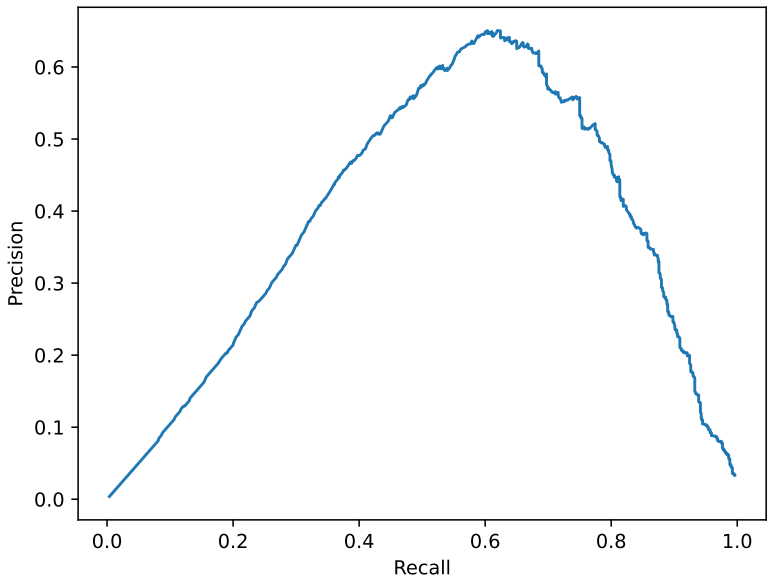

FIGURE 6.1: Precision-recall curve for the similarity score of the embeddings produced by the E5 model.

Model	NSC	PVV	VVD	VOLT
BERT	**0.817**	0.381	0.688	0.717
BM25	0.800	**0.476**	**0.767**	**0.900**
NLI	0.556	0.355	0.535	0.198
E5	0.507	0.353	0.375	0.520
GPT	0.590	0.194	0.504	0.643
GPT-4	0.814	0.347	0.624	0.829

TABLE 6.5: Comparison of average F1-score for different methods of generating references when looking at data points for NSC, PVV, VVD and VOLT. In total, these parties contained **15**, **14**, **20** and **21** out of 271 data points respectively.

recall, thus also yielding a greater F1-score. We also find that, while the results are close, BM25 gets outperformed by both BERT and GPT-4. This is mainly due to the lower precision achieved by BM25, though this is compensated by having a higher recall.

Next, we take a look at the results from the PVV (Table 6.5). Here we see the opposite effect of what we found for NSC, since all the methods perform significantly worse than the average. The only notable exception is NLI, which performs similarly to the average. Again, we find that this observation is the case for both the precision and recall. Both GPT models show the biggest drop in performance, becoming the two worst performing models.

When looking at the results of the VVD (Table 6.5), we see that there is a relatively large jump in performance for NLI, which is not observed in all methods. There is a slight increase in the performance for the retrieval based methods, but not as large as

found for NLI. We also see a slight drop in performance for E5, which is due to a decrease in precision.

Finally, when looking at VOLT, we find that their results outperform the average for all models, except NLI where it performs a lot worse. For both GPT models and BM25 we see a very large increase in performance, both in terms of recall and precision. This also includes the only model where the F1-score surpasses the value of 0.9.

7. Discussion

In this chapter, we will further discuss the results we found in the Results section. We will also discuss possible future avenues of research.

7.1 OpenVerkiezingen

In terms of the ratio of users that interact with the system, the finding that less panels are clicked when the reference are present goes against our hypothesis. The expectation was made that, even excluding the references, this number would increase, but there actually was a decrease. This could suggest that the users interact only with the panels or only with the references. However, it could also mean that users gain an increased confidence in the system when they see the references are there, and are therefore less enticed to interact. Whether this effect is actually there requires further experimentation, preferably on a larger scale than used in this thesis.

The results from the number of interactions per user are as expected, having increased both the number of clicks on the source panels, as well as the overall interactions when including the references. However, when the references are excluded, the increase is not large enough to be significant.

7.2 Best References

When looking at the overall results across all parties, we found that BM25 outperforms the other models. This ranking was mainly based on the F1-score of the different models. While BM25 also had the best precision, it's recall was not the highest, in which E5 outperformed. One factor that should be taken into account when looking at these results, is whether precision or recall should be prioritized for the given task. Do we want to make sure that all the generated references are accurate, or do we want to ensure that all the possible references are shown. Looking at the differing results might influence the choice of a different method than the overall best performing one.

One caveat to these results, is that for BERT, BM25, and NLI, only a single threshold value was used. The results could be further optimized by changing the thresholds using the method performed on the E5 model. This might further increase the performance of these models, or allows them to perform better on either recall or precision as the product requires. This methodology can't be applied to the GPT models, as these do not make use of a threshold value.

The results show clearly that the method chosen in the AB-test, namely using GPT-3.5 Turbo, was far from optimal. However, the upgrade to the better performing GPT-4 is not as straightforward as it looks. This is mainly caused by the sharp increase in computation cost that using GPT-4 brings. So while this would instantly increase the performance of the system, it is a trade off that has to be made.

Looking at the results from the individual parties, we can find a number of interesting discrepancies. We find that the results for different parties can differ greatly. Also, different models seem to perform well for some parties, while they perform badly for others. There might be multiple reasons why this is the case.

One possible reason might be that the data is biased, due to the fact that my own political beliefs influence my ability to properly label the test data. While it seems like the more left leaning parties (which better align with my own political beliefs) seem to score at least on par with the average, this is also the case for most other parties. The main outlier in this situation is the PVV, which scores the worst on average.

Another possible explanation for the differing result has to do with the ways which the manifestos have been written. Looking back at the example of the PVV, we find that their manifesto is relatively short. This might suggest that there is less information present to base a statement on. This in turn means that, when asking OpenVerkiezingen to summarize a statement, the summary will be filled with less definitive statements. This also means that the semantic relevance of any given sentence in the sources will have a weaker connection to the summary sentences.

However, this would not explain the performance of the NLI model on the PVV data, which performed similarly to the average. For this, we can compare the writing style of the manifesto to that of VOLT, which performed poorly on NLI specifically. The main difference we can observe between the two manifesto's is that for the PVV, the text consists mainly of longer paragraphs, which convey more of a feeling towards the stated problem. This also means that multiple (similar) topics can be woven together in a single paragraph. This is in contrast to VOLT, which has a more distinct and concise writing style for individual issues. We find single topics/goals divided into bullet points that contain multiple sentences about this single topic. These results give the impression

that NLI models are better at predicting correct references when provided with text that is more emotionally charged or value or principles based.

This finding is further confirmed by looking at the manifesto of the VVD and NSC, both of which score well using the NLI method. While in both manifesto's we find a structure that divides the content up into more distinct topics, we also find a lot of sentences explaining or justifying why something is important or included. This contrasts what we find in the VOLT manifesto, which has little of this language.

One reason why this effect is magnified could be caused by the way the summaries from OpenVerkiezingen are generated. A RAG model might use either Extractive or Abstractive Summarization based on the provided information. Given a more distinct and concise writing style, like in the manifesto of VOLT, the answer is more easily extracted from the text directly. In contrast, the text from the manifesto of the PVV might require more abstraction to properly answer a given question. This discrepancy further explains why keyword based methods like BM25 score better on this task, especially given a more concise writing style.

For the other parties, the main relation between the text and performance seems to be connected to information density. For both VOLT and NSC, we find that each subtopic has a clear division. Within these sub-topics, the information consists mainly of clear statements about his topic. This makes it easier to identify the relevant references used in the summary. This lack of noise in text means that sentences are often copied partially and combined with a different sentence. Little restructuring is needed to make a coherent sentence from the provided sources. In the case of the VVD, there is more noise in the sentences, thus resulting in worse results.

One thing that might have an effect on the information density is how well OpenVerkiezingen fragments manifesto into chunks. As this is a mostly automated process, one manifesto structure might provide a better division of information than others. This in turn affects the sources that are provided with a given statement, and therefore the accuracy of the information available. For example, the fragments of the VVD manifesto are significantly shorter than those of NSC or VOLT.

Training language concerns

While the results are promising, concerns arise about the ability of the chosen models to actually understand the task at hand. This is because the training data for these models are almost exclusively English, while our data and prompts are all in Dutch. To

Model	Precision	Recall	F1
NLI Eng	0.372	0.319	0.327
NLI Multi	0.351	0.293	0.311
E5 Eng	0.393	0.830	0.468
E5 Multi	0.175	0.964	0.270

TABLE 7.1: Comparison of English (Eng) and multilingual (Multi) models of NLI and E5 methods.

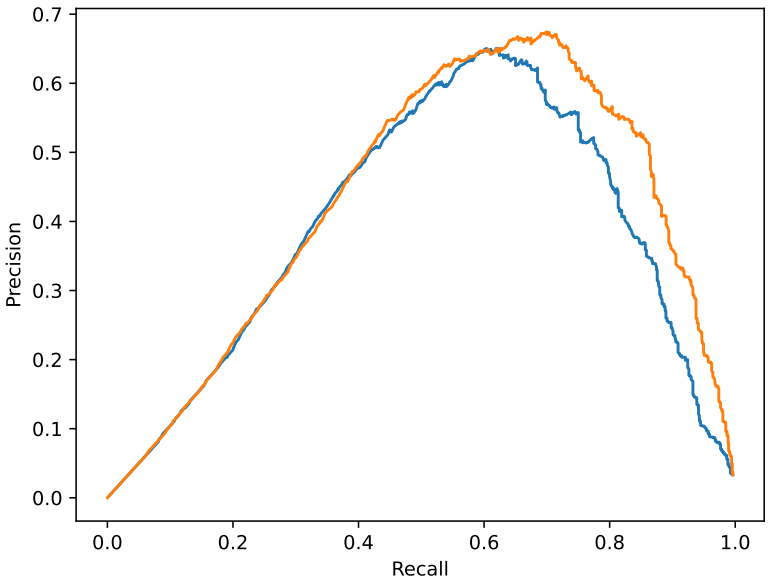

FIGURE 7.1: Comparison of precision-recall curves for the similarity score of the embeddings produced by the English (blue) and multilingual (orange) E5 models.

mitigate these concerns, a comparison is made between similar models that are trained on multilingual data.

In order to make a fair comparison between the English and multilingual models, I have searched for multilingual models that can be substituted in without making any changes. Models were found for E5 (multilingual-e5-large) and NLI (xlm-roberta-large-xnli). Using these models yielded the results shown in Table 7.1.

For NLI we see that the results are very similar, whereas the results for E5 have dropped compared to the English model. However, the results of E5 seem to be caused by the choice of threshold. This is made clear when we create a precision-recall curve for this result, as seen in Figure 7.1. The curves produced are very similar, giving good confidence in the ability of the chosen models to understand the provided task.

7.3 Future Work

Given these results, there still is room for improvement in certain parts of the research, which could be studied in the future. The first huge obstacle in the research is dealing with situations where no references are generated, or no references are expected. Currently, the dataset largely consists of snippets of text that have a reference associated with it, but this is not always the case. Cases where no references are expected cause issues with the calculations of the recall. Also, as was briefly stated in the results section, when a method does not find any references (even though they were expected), this will return NaN for the precision. Finding a better way to represent these situations will greatly improve the accuracy of the statistics, and provides greater confidence in which model generates the best references.

Besides this, it would be interesting to look at the effect of different LLMs that were designed for different languages. As this whole thesis was performed using Dutch data, it would stand to reason that it could increase the performance when using a model fine tuned for the Dutch language. Unfortunately, due to some technical difficulties, we were unable to apply that in this thesis.

8. Conclusion

In this thesis, our aim was to enhance an already existing AI powered voting guide using references. The RAG system used by OpenVerkiezingen was enhanced using an LLM, which was asked to provide references for text that was generated based on fragments of Dutch political party manifestos. This task was performed separate from the existing summarization in OpenVerkiezingen, in order to not interfere with the established method.

In order to test if references improved the user experience on OpenVerkiezingen, an AB-test was performed around the 2024 European Parliament elections. This test allows us to answer our first and second research question. Firstly, we found that the addition of references to AI-generated summaries increased the number of users who interact with the sources of these summaries. Secondly, we also found that the addition of references also increased the number of interactions with these sources.

Because the original method for generating references, using GPT-3.5, was chosen based on what would fit well within the ecosystem of OpenVerkiezingen, further tests were performed to answer our final research question; what methodology is best for generating references. Because there was no data available that was a good fit for this specific task, the data was generated by hand. Using this data, we found that the selected baseline of BM25 outperformed the other methods. It was also found that the larger GPT-4 model significantly outperformed GPT-3.5, but this also came with a sharp increase in cost.

Further investigation of the results of individual parties provided more insight in which of the chosen methods work better given certain writing styles. In general, it seems that references are more easily identified correctly if the manifesto text is divided into clear (sub-)topics, and has a higher information density.

A. Full Reference Prompt

```
[{
    "role": "system",
    "content": '''
        Je bent een assistent die bepaald op welke zinnen een gegenereerde zin gebaseerd is. Je doet dit gebaseerd op gegeven bronnen. Een referentie bestaat altijd uit 1 zin, en dus niet meerdere zinnen. Als je twijfelt of een zin een referentie is voor de gegeven zin, voeg hem dan niet toe. Een zin kan ook geen referenties hebben. Elke referentie bestaat uit maximaal 1 zin, en dus niet uit een hele paragraaf. Een referentie is dus echt een enkele zin. Eentje maar, niet meer.

        Geef jouw antwoorden terug als een json object. Zorg ervoor dat het antwoord niet te lang wordt. Het is erg belangrijk dat het antwoord uit maximaal 150 tokens bestaat!

        Voorbeeld 1:
        Zin: Ze benadrukken het belang van samenwerking binnen de EU om gezamenlijke uitdagingen aan te gaan, zoals klimaatverandering, migratie en veiligheid.

        Antwoord:
        {
            "zin": "Ze benadrukken het belang van samenwerking binnen de EU om gezamenlijke uitdagingen aan te gaan, zoals klimaatverandering, migratie en veiligheid",
            "references": [
```

```
15              "Alleen samen vinden we de oplossingen voor de grootste, grenzeloze uitdagingen van deze eeuw, zoals klimaatverandering, digitalisering, migratie en groeiende ongelijkheid.",
16              "We werken aan een volwaardige Europese defensie-industrie en vergroten de onderlinge samenwerking tussen defensie-eenheden.",
17              "Door samen te werken, samen keuzes te maken over ons klimaat, ons welzijn en onze welvaart, en onze veiligheid, blijft dat ook in de toekomst zo."
18          ]
19      }
20
21      Voorbeeld 2:
22      Zin: Deze partij is tegen de stelling "Nederland moet uit de Europese Unie".
23
24      Antwoord:
25      {
26          "zin": "Deze partij is tegen de stelling "Nederland moet uit de Europese Unie"",
27          "references": []
28      }
29
30      Voorbeeld 3:
31      Zin: In het verkiezingsprogramma wordt voorgesteld om vermogen eerlijker te belasten door een aparte vermogensbelasting in te voeren op vermogens in box 2 en box 3 boven een miljoen, met een hoger tarief boven de twee miljoen.
32
33      Antwoord:
34      {
35          "zin": "In het verkiezingsprogramma wordt voorgesteld om vermogen eerlijker te belasten door een aparte vermogensbelasting in te voeren op vermogens in box 2 en box 3 boven een miljoen, met een hoger tarief boven de twee miljoen.",
36          "references": [
```

```
37                  "We gaan vermogen eerlijker belasten.",
38                  "We voeren daarnaast een aparte
    vermogensbelasting in op vermogens in box 2 en box 3 boven
    een miljoen, met een hoger tarief boven de twee miljoen."
39              ]
40          }
41      '''
42      },
43      {
44          "role": "user",
45          "content": '''
46              "De volgende bronnen komen uit het
    verkiezingsprogramma van een politieke partij.\n\n"
47              f"{sources}\n\n"
48              f"\nDe volgende zin is gebaseerd op de gegeven
    bronnen: "
49              f"\"{summary_sentence}\". Geef aan welke zin of
    zinnen zijn gebruikt om de volgende zin te maken. Kopieer in
     je antwoord de volledige zinnen die gebruikt zijn. De
    referenties bestaan uit maximaal 1 zin. Geef maximaal drie
    referenties per antwoord. Een referentie is dus echt een
    enkele zin. Eentje maar, niet meer."
50          '''
51      }
52  ]
```

LISTING A.1: Full prompt used for reference generation. On line 47, 'sources' referes to the most relevant sources for the user provided statement. On line 49, 'summary_sentence' referes to the sentence in the summary we are generating references for.

B. Party Results

BBB 14 CDA 21 CU 20 D66 20 DENK 16 FVD 8 JA21 28 NSC 15 PVDAGL 28 PVDD 23 PVV 14 SGP 12 SP 11 VVD 20 Volt 21

Model	Precision	Recall	F1
MSMARCO	0.625	0.631	0.59
BM25	0.672	0.762	0.679
NLI	0.372	0.319	0.327
E5	0.393	0.83	0.468
GPT	0.424	0.661	0.487
GPT-4	0.649	0.655	0.641

Table B.1: Average precision, recall, and F1-score for different methods of generating references across all instances.

Model	Precision	Recall	F1
MSMARCO	0.417	0.464	0.393
BM25	0.586	0.798	0.644
NLI	0.107	0.143	0.119
E5	0.604	0.81	0.617
GPT	0.538	0.786	0.6
GPT-4	0.643	0.679	0.648

Table B.2: Average precision, recall, and F1-score for different methods of generating references when looking at data points for BBB. In total, **14** out of 271 data points are from BBB.

Model	Precision	Recall	F1
MSMARCO	0.647	0.603	0.597
BM25	0.487	0.659	0.519
NLI	0.19	0.159	0.159
E5	0.389	0.786	0.485
GPT	0.304	0.492	0.357
GPT-4	0.425	0.532	0.464

Table B.3: Average precision, recall, and F1-score for different methods of generating references when looking at data points for CDA. In total, **21** out of 271 data points are from CDA.

Model	Precision	Recall	F1
MSMARCO	0.671	0.675	0.615
BM25	0.699	0.792	0.725
NLI	0.525	0.408	0.437
E5	0.225	0.725	0.298
GPT	0.435	0.592	0.479
GPT-4	0.638	0.6	0.608

TABLE B.4: Average precision, recall, and F1-score for different methods of generating references when looking at data points for CU. In total, **20** out of 271 data points are from CU.

Model	Precision	Recall	F1
MSMARCO	0.596	0.683	0.606
BM25	0.621	0.775	0.658
NLI	0.275	0.275	0.275
E5	0.313	0.858	0.397
GPT	0.474	0.683	0.523
GPT-4	0.775	0.808	0.787

TABLE B.5: Average precision, recall, and F1-score for different methods of generating references when looking at data points for D66. In total, **20** out of 271 data points are from D66.

Model	Precision	Recall	F1
MSMARCO	0.266	0.219	0.212
BM25	0.578	0.438	0.467
NLI	0.5	0.292	0.354
E5	0.565	0.75	0.602
GPT	0.422	0.667	0.495
GPT-4	0.698	0.656	0.671

TABLE B.6: Average precision, recall, and F1-score for different methods of generating references when looking at data points for DENK. In total, **16** out of 271 data points are from DENK.

Model	Precision	Recall	F1
MSMARCO	0.438	0.438	0.417
BM25	0.625	0.688	0.646
NLI	0.438	0.438	0.417
E5	0.505	0.875	0.599
GPT	0.247	0.688	0.34
GPT-4	0.417	0.562	0.458

TABLE B.7: Average precision, recall, and F1-score for different methods of generating references when looking at data points for FVD. In total, **8** out of 271 data points are from FVD.

Model	Precision	Recall	F1
MSMARCO	0.738	0.661	0.663
BM25	0.697	0.705	0.669
NLI	0.25	0.134	0.169
E5	0.467	0.902	0.538
GPT	0.414	0.607	0.471
GPT-4	0.56	0.562	0.549

TABLE B.8: Average precision, recall, and F1-score for different methods of generating references when looking at data points for JA21. In total, **28** out of 271 data points are from JA21.

Model	Precision	Recall	F1
MSMARCO	0.853	0.817	0.817
BM25	0.767	0.933	0.8
NLI	0.567	0.567	0.556
E5	0.429	0.883	0.507
GPT	0.589	0.617	0.59
GPT-4	0.833	0.8	0.814

TABLE B.9: Average precision, recall, and F1-score for different methods of generating references when looking at data points for NSC. In total, **15** out of 271 data points are from NSC.

Model	Precision	Recall	F1
MSMARCO	0.599	0.756	0.644
BM25	0.62	0.833	0.661
NLI	0.42	0.381	0.371
E5	0.312	0.887	0.411
GPT	0.416	0.661	0.486
GPT-4	0.625	0.595	0.6

TABLE B.10: Average precision, recall, and F1-score for different methods of generating references when looking at data points for PVDAGL. In total, **28** out of 271 data points are from PVDAGL.

Model	Precision	Recall	F1
MSMARCO	0.519	0.674	0.523
BM25	0.624	0.826	0.677
NLI	0.457	0.435	0.435
E5	0.238	0.826	0.334
GPT	0.47	0.739	0.547
GPT-4	0.819	0.783	0.78

TABLE B.11: Average precision, recall, and F1-score for different methods of generating references when looking at data points for PVDD. In total, **23** out of 271 data points are from PVDD.

Model	Precision	Recall	F1
MSMARCO	0.345	0.464	0.381
BM25	0.435	0.571	0.476
NLI	0.369	0.357	0.355
E5	0.309	0.571	0.353
GPT	0.177	0.274	0.194
GPT-4	0.324	0.405	0.347

TABLE B.12: Average precision, recall, and F1-score for different methods of generating references when looking at data points for PVV. In total, **14** out of 271 data points are from PVV.

Model	Precision	Recall	F1
MSMARCO	0.708	0.667	0.65
BM25	0.847	0.792	0.803
NLI	0.417	0.292	0.333
E5	0.589	0.833	0.64
GPT	0.353	0.583	0.423
GPT-4	0.736	0.708	0.714

TABLE B.13: Average precision, recall, and F1-score for different methods of generating references when looking at data points for SGP. In total, **12** out of 271 data points are from SGP.

Model	Precision	Recall	F1
MSMARCO	0.879	0.652	0.709
BM25	0.924	0.697	0.761
NLI	0.364	0.258	0.288
E5	0.638	0.795	0.619
GPT	0.43	0.758	0.523
GPT-4	0.67	0.614	0.631

TABLE B.14: Average precision, recall, and F1-score for different methods of generating references when looking at data points for SP. In total, **11** out of 271 data points are from SP.

Model	Precision	Recall	F1
MSMARCO	0.767	0.667	0.688
BM25	0.755	0.825	0.767
NLI	0.567	0.542	0.535
E5	0.272	0.892	0.375
GPT	0.413	0.775	0.504
GPT-4	0.625	0.633	0.624

TABLE B.15: Average precision, recall, and F1-score for different methods of generating references when looking at data points for VVD. In total, **20** out of 271 data points are from VVD.

Model	Precision	Recall	F1
MSMARCO	0.782	0.73	0.717
BM25	0.901	0.944	0.9
NLI	0.214	0.206	0.198
E5	0.402	0.929	0.52
GPT	0.554	0.937	0.643
GPT-4	0.833	0.841	0.829

TABLE B.16: Average precision, recall, and F1-score for different methods of generating references when looking at data points for VOLT. In total, **21** out of 271 data points are from VOLT.

Bibliography

[1] Hans Schmeets. Politieke betrokkenheid in nederland. *Statistische Trends, december*, 2017.

[2] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[3] Yizheng Huang and Jimmy Huang. The survey on retrieval-augmented text generation for large language models. *arXiv preprint arXiv:2404.10981*, 2024.

[4] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[5] UGent-onderzoekers ontwikkelen politieke ChatGPT: "We willen met 'KamerRaad' de democratie een duwtje in de rug geven." — ugent.be. https://www.ugent.be/nl/actueel/ugent-onderzoekers-ontwikkelen-politieke-chatgpt-201cwe-willen-met-2018kamerraad2019-de-democratie-een-duwtje-in-de-rug-geven.201d, 2024. [Accessed 04-02-2025].

[6] Ilias Chalkidis. Investigating llms as voting assistants via contextual augmentation: A case study on the european parliament elections 2024. *arXiv preprint arXiv:2407.08495*, 2024.

[7] David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. Fact or fiction: Verifying scientific claims. *arXiv preprint arXiv:2004.14974*, 2020.

[8] Bernd Bohnet, Vinh Q Tran, Pat Verga, Roee Aharoni, Daniel Andor, Livio Baldini Soares, Massimiliano Ciaramita, Jacob Eisenstein, Kuzman Ganchev, Jonathan Herzig, et al. Attributed question answering: Evaluation and modeling for attributed large language models. *arXiv preprint arXiv:2212.08037*, 2022.

[9] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[10] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[11] Alec Radford. Improving language understanding by generative pre-training. 2018.

[12] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[13] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[14] Katja Filippova. Controlled hallucinations: Learning to generate faithfully from noisy data. *arXiv preprint arXiv:2010.05873*, 2020.

[15] Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. On faithfulness and factuality in abstractive summarization. *arXiv preprint arXiv:2005.00661*, 2020.

[16] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 2023.

[17] Bill MacCartney. *Natural language inference*. Stanford University, 2009.

[18] Ronak Pradeep, Xueguang Ma, Rodrigo Nogueira, and Jimmy Lin. Scientific claim verification with vert5erini. *arXiv preprint arXiv:2010.11930*, 2020.

[19] Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych. Ukp-athene: Multi-sentence textual entailment for claim verification. *arXiv preprint arXiv:1809.01479*, 2018.

[20] Amir Soleimani, Christof Monz, and Marcel Worring. Bert for evidence retrieval and claim verification. In *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II 42*, pages 359–366. Springer, 2020.

[21] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of

transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[22] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[23] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*, 2018.

[24] Silviu Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 708–716, 2007.

[25] Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*, 2016.

[26] Jacob Menick, Maja Trebacz, Vladimir Mikulik, John Aslanides, Francis Song, Martin Chadwick, Mia Glaese, Susannah Young, Lucy Campbell-Gillingham, Geoffrey Irving, et al. Teaching language models to support answers with verified quotes. *arXiv preprint arXiv:2203.11147*, 2022.

[27] Haolin Deng, Chang Wang, Xin Li, Dezhang Yuan, Junlang Zhan, Tianhua Zhou, Jin Ma, Jun Gao, and Ruifeng Xu. Webcites: Attributed query-focused summarization on chinese web search results with citations. *arXiv preprint arXiv:2403.01774*, 2024.

[28] Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

[29] L Xue. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020.

[30] Fariba Sadeghi, Amir Jalaly Bidgoly, and Hossein Amirkhani. Fake news detection on social media using a natural language inference approach. *Multimedia Tools and Applications*, 81(23):33801–33821, 2022.

[31] Moritz Laurer, Wouter Van Atteveldt, Andreu Casas, and Kasper Welbers. Less annotating, more classifying: Addressing the data scarcity issue of supervised machine learning with deep transfer learning and bert-nli. *Political Analysis*, 32(1): 84–100, 2024.

[32] Dilek Küçük and Fazli Can. Stance detection: A survey. *ACM Computing Surveys (CSUR)*, 53(1):1–37, 2020.

[33] Michael Burnham. *Natural Language Politics*. PhD thesis, The Pennsylvania State University, 2024.

[34] Jonas Kuhn, Gabriella Lapesa, Sebastian Padó, Sean Papay, and Patricia F Zauchner. Automatic analysis of political debates and manifestos: Successes and challenges. In *Robust Argumentation Machines: First International Conference, RATIO 2024, Bielefeld, Germany, June 5-7, 2024, Proceedings*, volume 14638, page 71. Springer Nature, 2024.

[35] Noah Bergam, Emily Allaway, and Kathleen Mckeown. Legal and political stance detection of scotus language. *arXiv preprint arXiv:2211.11724*, 2022.

[36] Abeer AlDayel and Walid Magdy. Stance detection on social media: State of the art and trends. *Information Processing & Management*, 58(4):102597, 2021.

[37] Estela Saquete, David Tomás, Paloma Moreda, Patricio Martínez-Barco, and Manuel Palomar. Fighting post-truth using natural language processing: A review and open challenges. *Expert systems with applications*, 141:112943, 2020.

[38] Ivan Bielik. Application of natural language processing to the electoral manifestos of social democratic parties in central eastern european countries. *Politics in Central Europe*, 16(1):259–282, 2020.

[39] Salomon Orellana and Halil Bisgin. Using natural language processing to analyze political party manifestos from new zealand. *Information*, 14(3):152, 2023.

[40] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.