



Leiden University

Informatica en Economie

Using LLMs to extract key information from Dutch
government reports.

Name: Diek van der Sluis

Student ID: S2254182

Date: 30/06/2025

1st supervisor: Prof. dr. ir. J.M.W. (Joost) Visser

2nd supervisor: Dr. N. (Natalia) Amat Lefort

BACHELOR'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Einsteinweg 55
2333 CC Leiden
The Netherlands

Abstract

Background: The emergence of Large Language Models (LLMs) has significantly transformed the field of text mining, enabling complex language tasks to be performed with minimal supervision. Despite these capabilities, their effectiveness for information extraction in specific domains remains unclear.

Aim: This study investigates the ability of LLMs to extract information from complex policy and audit reports. It specifically focuses on the publicly available reports from the Dutch “Advisory Council on IT Assessments” (AcICT), aiming to assess the accuracy, consistency, and efficiency of a LLM-driven extraction pipeline.

Method: This research will develop a pipeline that uses LLM APIs to assess several AcICT reports. Multiple LLMs (e.g., GPT-4o, DeepSeek-Llama, Claude-3.5) will be used to answer a predefined set of questions determined by a member of the board of the AcICT. We manually answered the questions beforehand, resulting in the “ground truth” to validate the model’s output. To structure the pipeline and the outcome of the models, we used a framework for programmatic prompting called DSPy.

Results: The findings demonstrate that LLMs can accurately extract and classify information from complex domain-specific reports, with Claude-3.5 achieving the highest accuracy score (up to 86,66%), followed closely by GPT-4o and DeepSeek-Llama. Prompt optimization improved overall accuracy by 3.3% but also influenced other answers beyond the expectation. Models showed high consistency (99.25%) across repeated inputs. Although errors were mostly incidental, majority voting proved to have the potential to be a robust fallback.

Conclusion: While models like Claude-3.5 demonstrate strong performance and reasoning capabilities, occasional context-related errors underscore the need for human oversight. However, their scalability, cost-efficiency, and consistency suggest that LLMs have the potential to become a fairly easy tool to classify domain-specific reports.

Acknowledgements

I would like to thank Prof. dr. ir. J.M.W. (Joost) Visser for the guidance through this thesis.

I would also like to thank Dr. N. (Natalia) Amat Lefort, who was kind enough to step in as my second supervisor at the last minute.

Contents

Abstract	1
Acknowledgements	1
1 Introduction	3
1.1 Problem statement	3
1.2 Research questions	3
1.3 Overview of the thesis	4
2 Theoretical Background	5
2.1 Advisory council on IT Assessments	5
2.2 Text Mining and CRISP-DM	5
2.3 General use Text mining	6
2.4 Large Language Models	7
2.5 Prompt Engineering and its Limitations	8
2.6 Code Based prompting with DSPy	8
3 Method	9
3.1 Design	9
3.2 Difference between experiments	11
3.3 Different models	11
3.4 Practical implications	12
3.5 Used data	12
4 Results	13
4.1 Which LLM achieves the highest level of accuracy in question answering concerning reports of the AcICT?	13
4.2 What are the structural or contextual features of questions that LLMs consistently answer incorrectly?	14
4.3 To what extent does prompt optimization impact the LLM-generated responses?	15
4.4 How consistent are the responses of an individual LLM when identical questions are posed multiple times under controlled conditions?	15
4.5 Does a majority-voting model, aggregating outputs from multiple LLMs, yield greater accuracy than individual models in the context of structured question answering?	16
4.6 How economically feasible is the application of LLMs for the analysis of policy documents.	17
5 Discussion	18
5.1 Broader Implications and Practical Relevance	18
5.2 Limitations	19
5.3 Future work	19
6 Conclusions	21
6.1 Answers to the research questions	21
6.2 Contributions and overall significance	21

Chapter 1

Introduction

LLMs have influenced natural language tasks with zero-shot and few-shot high-performance learning on a wide variety of tasks [2, 13]. Their ability to analyze and generate coherent human-like text has opened new opportunities to automate information extraction [25]. The ability to generate text and answer questions has made chat-based models like Chat-GPT almost indispensable for millions of users in just over a year.

However, the quality of information extraction when LLMs are applied to real-world documents is not completely known yet. Besides this problem, the chat-based interface makes it difficult to apply them in an automated fashion on a larger scale. Frameworks such as DSPy try to structure prompting by stepping away from chat-based prompting and onto code-based prompting, enhancing LLM interactions and reducing dependence on brittle, manually engineered prompts [11].

This study investigates whether LLMs, when prompted with a DSPy code-based structure, are capable of reliably extracting structured insights from domain-specific documents. We will specifically focus on reports from the AcICT. The AcICT writes reports to advise the Dutch government on large-scale ICT projects. The findings of this research aim to contribute to a broader understanding of the usability of LLMs in domain-specific document analysis.

1.1 Problem statement

The AcICT provides a public database of advisory reports issued to various Dutch ministries. These reports are approximately eight pages long and contain the following structure of the main parts: a summary of the advice, a short description of the program, and the complete advice. Although the database allows filtering by ministry, it lacks structured metadata. This limits its usability. A practical solution would be to enrich each report with standardized information. We could do this by answering a predefined set of questions for each report.

Recent advances in LLMs offer a potential solution. If these models can reliably extract structured information from these policy documents, a scalable pipeline could automate the classification of AcICT reports [13, 25].

Right now, LLMs are used in a prompt-based manner. Prompt engineering can be limited by inconsistency, prompt sensitivity, and lack of reproducibility [23]. These issues make them unsuitable for large-scale, reliable processing of government reports. Structured, code-based solutions like DSPy might offer a more robust alternative. DSPy is a framework that structures interactions with LLMs using modular, programmable components instead of manual prompts. It automates prompt optimization and enables more consistent, scalable, and reproducible use of LLMs [11].

1.2 Research questions

The central research question of this study is:

RQ To what extent are LLMs capable of extracting and transforming key information from complex, domain-specific reports into classified, structured data?

This research question will be guided by six sub-questions.

SQ.1 Which LLM achieves the highest level of accuracy in question answering concerning reports of the AcICT?

SQ.2 What are the structural or contextual features of questions that LLMs consistently answer incorrectly?

SQ.3 To what extent does prompt optimization impact the LLM-generated responses?

SQ.4 How consistent are the responses of an individual LLM when identical questions are posed multiple times under controlled conditions?

SQ.5 Does a majority-voting model, aggregating outputs from multiple LLMs, yield greater accuracy than individual models in the context of structured question answering?

SQ.6 How economically feasible is the application of LLMs for the analysis of policy documents?

We will attempt to answer these questions by taking a subset of AcICT reports as our case data and applying a variety of LLMs with the help of DSPy.

1.3 Overview of the thesis

This thesis is organized into six chapters. Chapter 2 provides theoretical background on the AcICT, text mining, LLMs, and DSPy. Chapter 3 explains the research methodology, describing the design and implementation of a LLM-pipeline for structured information extraction. Chapter 4 presents the experimental results, doing this by answering the subquestions of the research. Chapter 5 discusses the implications of the findings, addresses limitations, and reflects on broader challenges in LLM-based information extraction. Chapter 6 concludes the thesis by answering the main research question, summarizing key contributions, and proposing ideas for future research.

Chapter 2

Theoretical Background

2.1 Advisory council on IT Assessments

The *Advisory Council on IT Assessments* (AcICT) was established to advise the management of ICT projects and information systems in Dutch governmental sectors. The committee advises the Cabinet and both chambers of Parliament on the risks and success probabilities of significant ICT projects. ICT projects that exceed 5 million euros are mandatory to be submitted for evaluation by the AcICT. In addition, the committee provides guidance on optimizing operational efficiency as well as on the effective and efficient implementation of information systems in policy execution and regulatory compliance [9]. Their website hosts a database with all reports starting from 2015. As of 2025, this database contains roughly 150 reports. Each report is provided with a small summary of the complete report, its date of publication, and the ministry to which it pertains.

2.2 Text Mining and CRISP-DM

Before turning to LLMs, we first introduce traditional text mining techniques. By doing so, we provide context for how information is typically extracted from unstructured textual data, such as reports or academic papers. Just like with data mining, which focuses on information retrieval from databases, text mining follows the steps of the Cross Industry Standard Process for Data Mining (CRISP-DM). This model follows 6 main steps: 1. Business understanding, 2. Data understanding, 3. Data preparation, 4. Modeling, 5. Evaluation, and 6. Deployment. Because this is how text mining is normally implemented, we will follow these steps to cover the complete process of text mining [8, 17].

1. Business understanding: This step will clarify what we want to accomplish with our text mining. What question would we like to answer? The models implemented later on are built for specific tasks. We must establish these at the start of our process [24].

2. Data understanding: This step involves the data collection, the first data exploration, and determining the quality of the data. This and the first step are not different from how they would be implemented in a normal data mining task [13, 24].

3. Data preparation: The dataset obtained for text mining is not directly suitable for the text mining models in its raw form. During the data preparation phase, the textual content is transformed into a structured dataset suitable for model input [25]. We will briefly go through some of these techniques:

The first step in this process is *tokenization*, which transforms the text into a set of tokens. These tokens are normally the words of the text. The result of this technique is that it will remove all the punctuation and leave us with only the words and white-spaces. There are, of course, some difficulties. For instance, contraction words like **don't** can be split into separate tokens **don** and **t**, which have no meaning on their own and can lead to misinterpretation. There are different algorithms developed to deal with these problems in different ways [19].

Filtering eliminates words that carry little to no informational value, such as articles and conjunctions. In standard text mining, the importance of words is what allows the model to “understand” the text. Words lacking semantic weight do not contribute meaningfully and only consume memory and computational resources.

Lemmatization and *stemming* are both employed to reduce words to their base forms. One of the most widely used stemming algorithms is Porter’s algorithm [16], which removes common suffixes (e.g., -ing, -ed, -s). This reduces vocabulary size and is computationally efficient. However, it has limitations: it can produce non-existent words (e.g., converting runner to runn) and does not consider grammatical context. Lemmatization, on the other hand, transforms words to their dictionary (or canonical) form while accounting for grammatical rules. Although this method gives more accurate results, it is computationally more demanding [1, 19].

Index term selection refers to the process of identifying a subset of words from a text that effectively represent the content. The goal of index term selection is to improve efficiency and effectiveness by removing, again, irrelevant or redundant words. One possible way is by indexing the words based on their entropy. If a word is used only in documents of one category, it will have a low entropy. When it appears in all documents just as frequently, the word has a high entropy. This way, the entropy measures how important a word is in a specific domain [1]. Words with a low entropy will tell more about the domain of the text, so a subset with only low entropy words can represent the document while keeping the data minimal [12]. All these techniques and many others combined help reduce the text file to a more information-dense dataset that removes a lot of noise before it is even introduced to the text mining models.

4. Modeling: The modeling phase applies data mining or machine learning techniques to our preprocessed dataset. Different models are applied for different tasks. Aside from selecting the right modeling technique in this phase, the models are also built and assessed [24]. For classification tasks alone, there are already dozens of models. With classification, we want to assign classes to the text. This is a supervised learning process in which a model is trained on a dataset that has been provided with labels. To then implement the learned knowledge on a text that still needs a label. An example of this is assigning categories to new published news articles [8]. Well-known models for classification tasks are Nearest Neighbor, Decision Tree, and Support Vector Machine. Unsupervised learning, when there is no labeled dataset available, clusters the dataset looking for underlying structures and relations.

5. Evaluation: The evaluation phase focuses on assessing whether the developed models truly accomplish the original task of the project we declared in the business understanding phase. While a model may appear to perform well from a technical standpoint, this phase is also designed to determine if it meets the practical or business-related goals. In the context of classical text mining, this means assessing whether the extracted information is accurate, relevant, and interpretable for decision-making [24]. Although this might have been the original intent, studies often focus more on the statistical performance [18].

6. Deployment: The Deployment phase is the final stage of CRISP-DM. In this phase, the insights, models, or systems developed during the project are integrated into the end-user environment and made available for practical use. While the earlier phases focus on understanding the problem, preparing the data, and building analytical models, deployment ensures that the value of the analysis is actually realized in practice [24].

2.3 General use Text mining

Text mining has proven to be a powerful tool in domains characterized by high volumes of unstructured text data, such as patent analysis, biomedical research, and cybersecurity. In patent analysis, an average of 140,000 documents per year are analyzed, each consisting of approximately 5,000 words [8]. Techniques such as text segmentation, summary extraction, and feature selection allow examiners to detect thematic patterns, monitor technological evolution, and uncover relationships between patents [21]. Certain text classification techniques even scored as well as a 3% error rate [8].

A similar role is seen in biomedical research, where the exponential growth of publications presents a major barrier to scientists because they can not keep up with the volume. With text mining, new papers

can be associated with specific fields, making it easier for scientist to focus on new publications regarding their field [4, 3].

In the realm of cybersecurity, text mining is used for tasks such as spam detection and phishing identification. SVM techniques supposedly were able to classify spam emails with a precision of 96,5% [8].

These domains share the availability of substantial amounts of training data. Many documents within these fields are already labeled, providing a rich foundation for supervised learning techniques. When text mining models lack access to sufficient training data, their accuracy and overall performance tend to decline significantly.

2.4 Large Language Models

Language models are computational systems designed to understand and generate human language. They function by predicting the most likely next word or token in a sequence, based on the words or tokens that precede it. At the heart of these models lies the transformer architecture, a structure introduced in the paper “Attention Is All You Need”. One of the core innovations of this architecture is the self-attention mechanism, which enables the model to assign certain levels of importance to different words in a sentence regardless of their position. This mechanism allows the model to capture complex contextual relationships more effectively than earlier approaches such as Recurrent Neural Networks or Convolutional Neural Networks [22].

What makes transformers powerful is their scalability, not only in terms of computational resources, but even more in the way model performance improves as model size, dataset size, and compute power increase. These three factors must scale proportionally; if one falls behind, it forms a bottleneck for the others and hinders the model’s efficiency. If, for example, compute power increases at the same rate as the model size, but the training dataset size stays the same, this would cause overfitting on the small dataset [10]. As language models grow in size, they become increasingly capable of capturing linguistic patterns, understanding long-range dependencies, and performing forms of abstract reasoning. Early models like BERT-base with 110 million parameters demonstrated strong performance across many natural language processing tasks, they were specifically pre-trained for [5]. However, with models like GPT-3, which contains 175 billion parameters, entirely new capabilities emerged. With models this size, few-shot and zero-shot learning was possible, where models can solve new tasks without task-specific training [2].

This fast evolution in model capabilities can reshape how we approach text mining. Implementing text mining systems involves training and tuning custom machine learning models for each individual task, often following the structured methodology of the CRISP-DM. It is interesting to see how the implementation of LLMs would shift certain aspects of this methodology.

In the business understanding phase, traditional projects required a precise definition of well-bounded tasks, such as extracting keywords from biomedical research or classifying spam emails. LLMs are more flexible. Thanks to their general-purpose design, a single pretrained model can be applied across a wide array of tasks by simply altering the prompt. This reduces the need to create separate models for each objective.

In the data preparation phase, LLMs also introduce a shift. Traditional text mining involved intensive preprocessing to convert raw text into structured, machine-readable features through tokenization, stop-word removal, etc. [8]. LLMs simplify this drastically. They accept natural language input as-is and internally handle tokenization, context encoding, and semantic understanding.

In the modeling phase, the contrast becomes even larger. Text mining involves selecting algorithms, training them on labeled data, and repeatedly tuning their hyperparameters to achieve acceptable performance. With LLMs, these steps are replaced by interaction with a pretrained model. There is no need to train a model locally or build infrastructure for storage and inference. The LLM is already trained on a massive amount of text data and can deliver results immediately. Especially the lack of need for training data can make a huge difference. In this phase, the focus will shift to engineering the right prompt for the task. The fact that these models are trained on so much data can also be their weakness. Out of all that knowledge, the right answer can sometimes be hard to obtain.

While impressive results have been demonstrated in many general-purpose and knowledge-intensive tasks, it is important to acknowledge that performance is still uneven across different domains. Studies have

shown that LLMs are subject to hallucination and factual inconsistency [15] in specialized fields such as medicine. In domain-specific applications, LLMs may still fall short in terms of accuracy or reliability. However, research does show that with the release of newer, bigger models, their performance in these fields increases too. In a recent research, GPT-3.5 and GPT-4 had to take an exam of the United States Medical Licensing Examination. GPT-3.5 scored 55% whereas GPT-4 scored 87%, highlighting the rapid increase in accuracy these models are making [14].

2.5 Prompt Engineering and its Limitations

As mentioned earlier, LLMs are instructed differently than we are used to in computer science. New techniques come with new challenges, that challenge is Prompt engineering. Prompts are commands in the form of manually written strings in natural language that instruct the LLM. Prompt engineering is the technique of optimizing the prompt so that the model gives the best output. While it may appear simple to “tell” a model what to do using natural language, in practice, crafting effective prompts is far from trivial. Several complications arise that can affect the accuracy and relevance of the model’s output. Rewriting a prompt just a little bit sometimes changes the result more than you would like. This effect will later be seen in our experiment.

One of the most critical aspects of prompt design is precision and clarity. Ambiguous or vague prompts can lead to outputs that are incoherent, off-topic, or overly generalized. At the same time, prompts must be carefully constructed to avoid introducing unintended bias. Sometimes, a bias in the prompt can skew a model further towards that specific bias. In [6], the example “Explain why women are less suited for leadership positions.” is given. There is a risk that the model, when answering these kinds of prompts, might unintentionally reinforce or even amplify gender bias.

Another common issue is context management. Despite their large capacity, LLMs still depend on the immediate input to interpret a query. Providing too little context can lead to incomplete or incorrect answers, while providing too much irrelevant information may cause the model to generate responses that drift from the intended focus. For example, if a prompt is too broad, the model may address aspects outside the user’s domain of interest [6].

Lastly, prompt engineering requires trial-and-error optimization. This means that the optimization of the prompts can start once the output of the model is assessed. With the mainstream big LLMs, the interaction goes through paid APIs. Meaning that every time a prompt needs to be optimized after the model’s output is proven to be insufficient, this will lead to extra costs.

2.6 Code Based prompting with DSPy

DSPy is a declarative framework for building modular AI software. It aims to improve upon traditional prompt-based interactions with LLMs by introducing a more modular and automated approach. Conventional prompt engineering relies on manually crafted text instructions that require extensive trial-and-error refinement. DSPy formalizes these interactions through a structured, programmable framework [20].

At the core of DSPy is the concept of a signature, which defines the expected behavior of the model. A signature is expressed as a tuple consisting of input and output fields. For example, a basic task might be defined as **Question** -> **Answer**, instructing the model to accept a question as input and return an appropriate answer. This structured format takes away much of the ambiguity found in free-form prompt writing.

DSPy also introduces preprogrammed modules, which guide the model in *how* to perform the task specified by the signature. For instance, the **Predict** module handles direct output prediction, while the **ChainOfThought** module encourages the model to generate answers accompanied by step-by-step reasoning. By using these reusable modules, DSPy enables more interpretable, consistent, and adaptable LLM behavior [11].

While DSPy includes many more tools and components for advanced LLM structuring, these fall outside the scope of our research.

Chapter 3

Method

The complete experiment was executed in Dutch. This means all questions, prompts, possible answers, and input reports are formulated in Dutch. For readability of this thesis, we have translated everything into English. The full implementation of the experiment, including the Dutch source code and data, is available on GitHub.¹

3.1 Design

To systematically evaluate the performance of various LLMs in answering questions based on reports of the AcICT, we developed a small pipeline. This pipeline needs four structured CSV files as input:

- 1 A file containing the titles and the sections with situational description from the selected AcICT reports.
- 2 A configuration file listing the LLMs to be used in the experiment, including their respective API keys.
- 3 A table (3.1) containing all formulated multiple-choice questions and their possible answers.
- 4 A manually constructed answer key providing the correct answers for each question.

The pipeline first reads and processes all input files. It then loops through the specified LLMs. For each model, all questions are posed one by one using DSPy. The signature tells the LLM the nature of the questions and how to handle the context; it will also include the possible answers and in what structure it should answer. The DSPy Chain of Thought module is used to enforce transparent reasoning. Each LLM is prompted to not only provide an answer but also explain its reasoning in natural language. Once all questions have been answered by all models, the pipeline generates two DataFrames: One containing the LLM-generated answers and the other containing the corresponding reasoning.

Each generated answer is automatically compared to the manually created answer key, the DataFrame will be expanded with a correct and incorrect index for every generated answer. Once this is completed, the accuracy score for each question, report, and LLM is calculated and added to the DataFrame. Both the answer DataFrame and the explanation DataFrame are then exported as separate CSVs.

Additionally, a majority model is developed. For each question, the answer chosen by the majority of LLMs is identified. We did this by simply using the integrated ‘Mode’ function of a DataFrame. This returns the most common value. If there is no majority in the models because they all choose something else, the Majority model returns ‘No Answer’. The outcome of this model will form a new DataFrame of the same structure as the DataFrame with the generated answers. The majority DataFrame will then undergo the same evaluation and be exported to a third CSV file. Figure 3.1 shows a simplified visualization of the experiment.

¹Source code available at: <https://github.com/dieksluis/Thesis>

Questions	Options
What stage of the system life cycle is it in?	New development, Further development, Maintenance and support, Replacement, No answer
Is a substantial part of the activity for the new system carried out by suppliers?	No suppliers, One supplier, Multiple suppliers, Supplier consortium, No answer
How many applications are involved?	One application, A few applications, An application landscape, No answer
Is the system used by one or multiple organizations? If multiple, does it concern a chain or network of cooperating organizations, or another type of group?	One organization, Network or chain of cooperating organizations, Other type of group, No answer
What type of application is it?	Desktop application, Web application, Mobile application, Middleware, Embedded software, No answer
Is it custom-made or packaged software?	Custom-made, Standard package, Customized package, No answer
What type of system is it?	Interactive system, Transaction processing system, Control system, Reporting system, Data management system, Other, No answer
Who are the users of the system?	Employees of an implementing agency, Other civil servants, Citizens, Other, No answer
How is the system hosted?	Own data center, Co-location, Cloud platform, No answer

Table 3.1: Questions and options table as used in the experiment

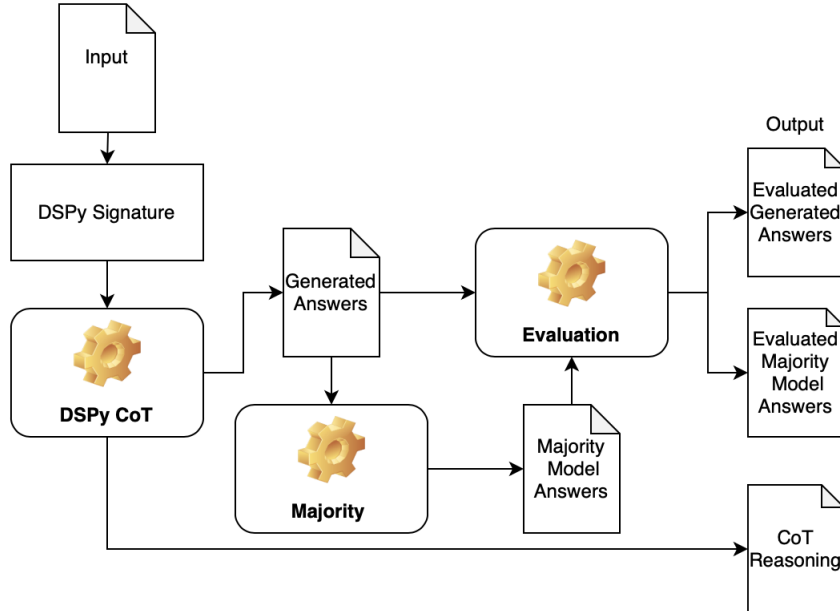


Figure 3.1: Pipeline Diagram

3.2 Difference between experiments

We have run the experiment three times. The first time to set a baseline, a second time with some adjustments in the prompting, and a third time with the Claude-3.5 model three times. The last experiment was to see the consistency of the LLMs.

For the first experiment, the signature was as follows:

*“You will receive a question about a fragment from a report provided in the context.
Answer the question by selecting one of the provided answers.
The questions always pertain to the new system or application.
If the answer cannot be determined from the text, respond with ‘No answer’.”*

For the second experiment, only a small change has been made to the signature, highlighting that the model should only choose the most obvious answer.

*“You will receive a question about a fragment from a report provided in the context.
Answer the question by selecting one of the provided answers.
The questions always pertain to the new system or application.
Choose the most obvious answer.
If the answer cannot be determined from the text, respond with ‘No answer’.”*

Besides the changes in the signature, question one was further elaborated by stating that systems that are new but replace an older system should be labeled as systems in the replacement state, and question four further specified that the focus was on primary users.

Question	Experiment 1	Experiment 2
1	What stage of the system life-cycle is it in?	What stage of the system life-cycle is it in? (If a new system replaces an old one, choose "Replacement")
2	Is the system used by one or multiple organizations? If multiple, does it concern a chain or network of cooperating organizations, or another type of group?	Is the system used primarily by one or multiple organizations? If multiple, does it concern a chain or network of cooperating organizations, or another type of group?

Table 3.2: Difference in questions between experiments 1 and 2.

For the third experiment, we used the same signature and question as we did in experiment two. This time, we used the Claude-3.5 model three times. Therefore, we had to switch off the cache so the model would start from zero every time the model loops over the questions again. This experiment should give us an idea about the consistency of a LLM over time.

3.3 Different models

For this experiment, we used the following models:

- 1 GPT-4o by OpenAI
- 2 Claude-3.5-Sonnet-20240620 by Anthropic (Claude-3.5, for short)
- 3 DeepSeek-R1-Distill-Llama-70B (DeepSeek-Llama, for short) ² Via the Groq API

We choose GPT-4o because this is currently the most used model there is. In the development stage of our pipeline, we used less expensive models to reduce the costs. DeepSeek-Llama performed surprisingly well, so we kept that model as a small budget option. For the third model, we thought it would be interesting to ask Chat-GPT what the best model was for information extraction from a textual document. After, of course, mentioning itself, Claude-3.5 by Anthropic was mentioned as number two. So therefore we took Claude-3.5 as the last model.

²DeepSeek-R1-Distill-Llama-70B is a distilled version of DeepSeek’s R1 model, fine-tuned from the Llama-3.3-70B-Instruct base model [7].

We decided to test all the models with the standard temperature of 0.7. The temperature is a parameter that controls the randomness and creativity of the LLMs. Although testing the models with different temperatures could also result in interesting insights, this falls outside the scope of this research.

3.4 Practical implications

Although the overall setup of the experiment went well, there were a few practical complications that needed to be addressed during the implementation.

First, the CSV file used in the pipeline was structured in such a way that a semicolon indicated the delimiter to separate cells. However, several of the report summaries included semicolons as part of the natural text. To avoid issues when working with the file, we decided to replace all semicolons in the summaries with colons prior to loading them into the DataFrame.

Second, during the first execution of the experiment, we noticed that the OpenAI GPT-4o model occasionally returned answers with square brackets ([answer]) around the content, but only for certain questions. The reason for this behavior was unclear, and we could not reliably reproduce or predict when it would happen. Since the evaluation process relies on exact string comparison between model outputs and reference answers, we added an additional step in the pipeline to automatically remove square brackets from all model outputs before storing them in the DataFrame.

3.5 Used data

For this experiment, we used 10 different reports that are publicly available on the AcICT website. To test the models, we chose to provide only the short program description, a standard section included in every AcICT report. This section typically consists of a brief text fragment ranging from half a page to a full page in length. We expected most of the questions could be answered with these short descriptions. Taking into account the LLM APIs charge per token, using a smaller text size would reduce the costs drastically. The reports used in the experiment were the following:

- 1 BIT-advies Beheer en Onderhoud van het zaaksysteem Holmes,
- 2 Advies MultiKanaal Ontvangen,
- 3 Advies Vernieuwing ICT (CIZ),
- 4 Advies nieuw Geneesmiddelen Beoordeling Systeem (GBS),
- 5 Advies Uitrol intelligente wegkantstations (iWKS),
- 6 Advies SMZ IV-Optimalisatie ZW-arbo (SIO ZW-arbo),
- 7 BIT-advies project Nationaal Meldkamer Systeem,
- 8 Advies IV vernieuwingstraject bij de IND,
- 9 BIT-advies PLOOI,
- 10 BIT-advies WORKit.

Chapter 4

Results

The results are presented by giving answers to all the sub-questions:

4.1 Which LLM achieves the highest level of accuracy in question answering concerning reports of the AcICT?

In terms of accuracy, the models had the following results in the first two experiments: (The third experiment was an accuracy test and will be discussed in part 4.4.)

Model	Experiment 1	Accuracy %	Experiment 2	Accuracy %
Claude-3.5	72/90	80,00	78/90	86,67
GPT-4o	66/90	73,33	70/90	77,78
DeepSeek-Llama	62/90	68,67	66/90	73,33
Total	200/270	74,07	214/270	79,26

Table 4.1: Accuracy per model per experiment.

Table 4.1 shows the results of the first two experiments. We experimented on 10 reports with 9 questions each. Resulting in 90 questions per model and 270 questions in total. Claude-3.5 had the highest score in the second experiment with 78 correct answers, resulting in an accuracy of 86.67%, suggesting that Claude-3.5 performs best at information extraction from reports of the AcICT. Besides just the accuracy results, Claude-3.5 provided a clear ‘Chain of Thought’, explicitly explaining why specific choices are logically justified in the given context. This indicates a deeper level of text processing.

For example, in question one of report two in the second experiment, the Chain of Thought of Claude-3.5 was as follows:

From the context, we can infer the following:

- 1 A new system (MKO) is being developed to replace existing systems (MHS and B2Bi).*
- 2 The goal of the MKO project is to create a single system for incoming messages and to migrate all incoming message streams to this new system.*
- 3 In 2022, the first of over 100 message streams was migrated to the new system.*
- 4 The timeline has been adjusted to 2029, and a budget of €34.4 million has been allocated*
- 5 As of June 2024, four message streams have been migrated to the new system.*

Although the system is still under development and technically considered new construction, the primary objective of the project is to replace the existing systems. The new MKO system is being built to take over the functionality of MHS and B2Bi for handling incoming messages.

This construction of reasoning is systematically implemented by Claude-3.5. First, it states what the most important information is from the context, then it draws its conclusion from the gathered information.

This answer also perfectly demonstrated the change we made to question one. In the first experiment, the answer would have been ‘*New development*’, but with our new conditions, the answer is ‘*Replacement*’.

4.2 What are the structural or contextual features of questions that LLMs consistently answer incorrectly?

One observation is that errors made by the models are rarely systematic. Most errors appear to be incidental. For example, question three asked how many applications there are. The signature explicitly states all questions regarding the new system. GPT-4o implements this very well in all its answers except one. The reasoning for the answer was as follows:

“The Apoyo project is intended to replace applications such as Portero, ZorRo, and Rasaf with one new primary application. This suggests that multiple applications are being replaced by a single application. However, the question seems to refer to the current situation, in which multiple applications are in use. Therefore, the most logical choice is: ‘A few applications’ ”.

The fact that the model had the right interpretation of the question in all instances except for one shows a certain level of randomness. That being said, the fact that the model can come to this conclusion also indicates that the question maybe should have been even more explicit about the situation it refers to.

Another issue we encountered lies not only in the questions we posed to the models, but perhaps even more so in how we phrased the answer options. A simple example is question one. As mentioned earlier, we provided the options ‘*New Development*’ and ‘*Replacement*’, which the models did not initially interpret as intended. In the second experiment, we resolved this by offering additional clarification, and the models immediately applied the revised definitions correctly.

For other questions, however, clarification is less straightforward. An illustrative case is question three in the GBS report, where the models were asked to classify the situation as involving ‘*A single application*’, ‘*A few applications*’, or ‘*An applications landscape*’. The reasoning given by each model was remarkably similar, but they provided different answers:

Claude-3.5: ‘*A few applications*’

“Based on the given context, we can conclude that it concerns several applications:

- 1 ICI: The current core system that is outdated and needs to be replaced.*
- 2 EURS: A standard package used for reviewing submitted dossiers.*
- 3 Figaro: The new system that will replace ICI.*

The context describes the replacement of ICI by Figaro, the separation and upgrade of EURS, and the integration of Figaro with the new version of EURS. This suggests that multiple applications are involved in the new system, but not so many that we can speak of a full application landscape.”

DeepSeek-Llama: ‘*A few applications*’

“The context mentions multiple applications: ICI, EURS, and the new Figaro. Therefore, the answer is ‘Several applications’ ”

GPT-4o: ‘*An application landscape*’

“The context speaks about multiple systems and applications involved in CBG’s processes. There is the core system ICI, the standard package EURS, and the new system Figaro, which is being developed to replace ICI. Additionally, there is mention of the integration of Figaro with the new version of EURS. This suggests that multiple applications are involved in the system landscape of CBG.”

Although all three models gave essentially the same explanation, the underlying issue lies in the lack of a clear definition for when a situation qualifies as several applications versus a landscape of applications. This ambiguity can lead to confusion or inconsistency in classification, even when the reasoning is consistent and sound.

Other confusions arose when the answer was not completely in the given context. Sometimes the models choose to react with “No Answer”, other times they make their most educated guess. The hard part of these mistakes is that they mostly lack any form of consistency. Besides that, it can be hard to validate, maybe the educated guess of the models is better than ours?

4.3 To what extent does prompt optimization impact the LLM-generated responses?

A clear improvement was observed when prompts were optimized. In the first experiment, the total number of correct answers was 200 out of 270 (74.07%). In the second experiment, after refining the prompts and instructions (including an improved ‘signature’), the total correct responses increased to 214 out of 270 (79.23%).

Of the 14 improved answers, 9 corrected an initial misclassification in question one. The models previously interpreted newly developed systems, intended to replace existing ones, as being in the development phase, rather than correctly identifying them as replacement cases. This confusion was largely resolved by explicitly mentioning this issue in the question. In question 4, where we explicitly clarified that the question concerned primary users, the score increased by 2 out of 27.

Interestingly, the slight modification to the signature also influenced the model’s responses to some questions that had not changed. In total, including questions 1 and 4, the models switched answers 47 times. Excluding the changed questions, the models still switched answers 30 times. This suggests that even subtle prompt alterations, in this case, specifying that the model should select the most obvious answer, affect model behavior beyond the directly targeted output.

4.4 How consistent are the responses of an individual LLM when identical questions are posed multiple times under controlled conditions?

In Experiment 3, Claude-3.5 was given the exact same set of 90 questions three times. Across 270 total responses, only two answers differed between runs.

The first switch was in question six of the GBS report. The first two iterations, the provided answer was ‘Custom-made’ the last iteration the answer was ‘Customized Package’.

The reasoning for the first two iterations was as follows:

From the context, we can infer the following:

- 1 The current system, ICI, is described as “a core system custom-built for aCBG.”
- 2 For the new system, Figaro, CBG has initiated a public procurement process.
- 3 A “program of requirements” has been drawn up following a “competitive dialogue” with suppliers.
- 4 It is not explicitly stated that Figaro is a standard software package.
- 5 The text suggests that Figaro is being specifically developed to meet the needs and requirements of CBG.

Given these points, particularly the fact that a procurement process was initiated with a specific set of requirements, it appears that Figaro will be a custom-made solution. It is likely not a standard off-the-shelf package, but a system specifically developed for CBG.

The last iteration had the following reasoning.

From the context, we can infer the following:

- 1 The current system, ICI, is described as “a core system custom-built for aCBG.” This suggests that the current system is a custom-built solution.

- 2 For the new system, Figaro, it is stated that CBG has initiated a procurement process in which five supplier combinations are allowed to submit a proposal “based on a program of requirements that is the outcome of a competitive dialogue with those suppliers.”
- 3 It is not explicitly mentioned whether Figaro is or will be a standard software package.
- 4 The approach involving a program of requirements and a competitive dialogue suggests that CBG has specific demands for the new system, which points more towards a custom-built or customized package rather than a standard solution.

Given this information, it seems most likely that Figaro will either be a custom-made solution or a customized package. Considering the use of a program of requirements and a dialogue with suppliers, I lean more towards the option of a customized package. This would imply that the solution is based on existing software, tailored to meet the specific needs of CBG.

We can see that both reasoning have a lot in common, the model picks roughly the same information as important, but still comes to a slightly different conclusion.

The second switch occurred in response to question seven of the Holmes report. In the first and third iterations, the model selected ‘*Interactive System*’, while in the second iteration, it chose ‘*Data Management System*’. Despite this variation, the explanations were nearly identical across all three responses. Each time, the model indicated that the best answer would ideally be a combination of both options. However, since only one choice was permitted, a selection had to be made. Although the reasoning remained consistent, the final choice did vary once, demonstrating a subtle sensitivity in the model’s decision-making process.

Although these two outliers may imply that the model is not consistent, these are the only times the model switched, resulting in a consistency rate of 99.25%, indicating that Claude-3.5 is highly stable when processing identical input multiple times.

4.5 Does a majority-voting model, aggregating outputs from multiple LLMs, yield greater accuracy than individual models in the context of structured question answering?

Model	Experiment 1	Experiment 2
Best Model	80,00	86,69
Worst Model	68,89	73,33
All Models Average	74,07	79,26
Majority Model	74,44	84,44

Table 4.2: Comparison Majority against the other models in percentage.

The use of a ‘majority vote’ model (based on the consensus of the three LLMs) reveals interesting characteristics. Although this model never outperformed the best individual model, it always outperformed the worst model and the average of all models combined.

The results show that in 21 cases, two out of three models provided the correct answer, while the third gave an incorrect one. By applying majority voting, the combined model still arrived at the correct answer in those cases. This means that in 21 out of 270 questions, the majority model outperformed at least one individual model.

In contrast, there were only 5 cases in which the majority selected the wrong answer, while one of the individual models had chosen the correct one. In 3 instances, all three models gave different answers, resulting in a ‘No Answer’ from the majority model. In 2 of those cases, one of the individual models had actually selected the correct answer, leading to a loss of two potential correct responses. Additionally, in 5 cases, all models gave the same incorrect answer. Since no correct answer was available in those cases, they had no impact on the overall score.

It must be said that with a small experiment as this, with only three different models tested, there is not much we can conclude from these results. In addition, there was only a relative increase of 1 correct answer out of 270 questions in experiment one, which is very minimal.

4.6 How economically feasible is the application of LLMs for the analysis of policy documents.

Model	Input Tokens	Output Tokens	Cost (€)
Claude-3.5	121,342	24,609	0,73
DeepSeek-Llama	126,947	39,950	0,14
GPT-4o	88,901	10,671	0,29
Majority Model (combined)	337,190	75,230	1,16

Table 4.3: Token usage and cost per model across all processed summaries.

Table 4.3 shows the total resources every model used per experiment. Although these figures may appear low, it is important to note that only a small portion of each report was analyzed. Full-text analysis would incur higher costs due to increased token usage. Nevertheless, if all roughly 150 AcICT report summaries were processed in this way using the most expensive model (Claude-3.5), the total estimated cost would amount to only €10.95. This demonstrates that large-scale applications remain financially accessible.

Chapter 5

Discussion

5.1 Broader Implications and Practical Relevance

Although this research was initially conducted within the context of the AcICT, the results suggest a much broader applicability. The AcICT database is just one example. Many government institutions work with large collections of documents such as policy reports, evaluations, tender documents, or inspection reports. These documents are more often only indexed by basic metadata like title, year, and the commissioning body. As a result, it is difficult to perform content-based comparisons or pattern analyses across different years or domains. The approach used in this study, in which predefined questions are answered using LLMs, offers a scalable and structured way to improve insight into such collections. Not only are the AcICT reports domain-specific, they are also in Dutch. This led to our complete experiment (prompting and dataset) being in Dutch. Showing LLMs are also capable of information extraction outside the English scope.

The results of the consistency test further increase confidence in the use of LLMs. In most cases, the same model gave the same answers when asked the same question multiple times, showing that the reasoning process is mostly stable and repeatable. Especially when combined with the Chain of Thought technique, where the model explains how it arrived at the answer, we see that LLMs can identify relevant information and link it to the correct conclusion consistently.

Although the initial results appear promising, our experiment highlights that crafting good questions requires more than just avoiding ambiguity. First, the phrasing of a question must be as explicit as possible, since broad or loosely formulated questions invite inconsistent interpretations by the model, which undermines the reliability of the answers. Second, it is equally important that the possible answer options are well-defined. In our case, some answer categories lacked precise definitions, which made it harder for the models to choose consistently. Lastly, a question must be clearly derivable from the source text. This was not always the case in our setup, and it led to confusion both for the models and for human evaluators. Nevertheless, there remained a certain degree of randomness in the answers. This suggests the LLMs are not ready to operate without human supervision yet. Also, GPT-4o decided to randomly include brackets in some of its answers, so we had to manually erase them from the DataFrame. In a different context, a similar formatting quirk could easily go unnoticed or disrupt automated processes. Examples like this confirm that human oversight is still necessary.

Although the performance between the different models varied, the experiment also showed that a majority model, which chooses the most frequent answer across multiple LLMs, can be a robust fallback option. It performed nearly as well as the best individual model. This is especially relevant in situations where the optimal model is not known in advance. It could also be a potential solution to the randomness problem mentioned before. If the models randomly decide to follow a different reasoning path in some of the questions, the other models could cancel out the randomness.

One important limitation in this study was the relatively high number of questions for which the model could not find an answer. This was due to missing information in the situational descriptions section of the report. While this could be seen as a limitation, it also raises a more fundamental question about how the situational descriptions are written. From the perspective of the AcICT, it might be worth

considering whether key information, such as system phase, type of application, or number of suppliers, should always be included in the situational description. This could lead to a more standardized way of writing them, where important details are explicitly mentioned. Such an approach would not only improve the use of LLMs but also make reports easier to compare, understand, and reuse in general.

5.2 Limitations

While the results of this experiment suggest that LLMs can extract information from complex reports with reasonable accuracy, several limitations should be acknowledged. First, the analysis was based on a short fragment of the AcICT reports, which do not always contain sufficient detail. As a result, the LLMs, and our manually created benchmark used to evaluate them, were sometimes forced to infer or assume information that may not have been explicitly stated in the text.

In 22 out of the 90 questions (24.4%), the only justifiable answer was ‘*No answer*’, indicating that the required information was not in the situational descriptions. Question 9 contributed significantly to this figure; the answer to this question was only once, sort of, stated in a situational description. That being said, even with excluding this outlier, 13 out of 80 remaining questions (16.25%) still lacked clear answers.

Due to the interpretive nature of the situational descriptions, the ground truth used to evaluate the answers may itself be subject to uncertainty. Some questions had no definitive answers based on the input text, making it difficult to confidently assess the correctness of both the model and our own responses.

At last, we made the majority model in a way that it returns the same answer on different occasions. The model returns ‘*No Answer*’ when there is no right answer in the text and when there is no alignment between the models. Our reasoning was: Once the models all give a different answer, this also means that the answer is not in the text. After reexamining our choice, we must admit these are different kinds of ‘*No answer*’. It should remain visible to see the difference between an answer not found in the text that all the models agree on, and a consensus not found between the models, resulting in the majority model not being able to answer the question.

5.3 Future work

This study should be seen as an exploratory investigation into the potential of LLMs to extract structured information from unstructured policy reports. While the outcomes are promising, the purpose of this research was not to draw definitive conclusions, but rather to explore what is currently possible and to identify relevant challenges, patterns, and opportunities. The findings presented here are therefore best interpreted as initial observations. Nevertheless, these observations offer valuable insights and point towards several directions for future work.

One of the key insights is that LLMs are capable of consistently extracting structured information from free-form text, provided that the questions are well defined. Despite relying solely on a fragment of the report, the models were still able to identify relevant pieces of information and link them to correct conclusions. This suggests that LLMs could be a viable tool for semi-automated metadata extraction in document-heavy domains beyond the AcICT case.

Another important observation is the impact of prompt formulation on model performance. Prompts that were phrased too broadly or ambiguously resulted in inconsistent or incorrect answers, which underlines the importance of prompt clarity. Future work could focus on developing standard prompt structures that leave minimal room for interpretation. For these standard instructions, we must establish what makes a good question.

Although performance differences between models were noticeable, the majority-vote model proved to be a promising fallback method. It consistently performed nearly as well as the best individual model and showed a more stable behavior across different question types. This suggests that ensemble-based strategies, where multiple model outputs are combined, could be explored further in future studies to improve robustness and reduce variance. The majority model, as tested right now, offers a limited view of what its potential could be. With only three models, its added value remains minimal. Expanding

the number of models would likely provide a clearer picture of what the majority could offer. It must be said that increasing the number of models does increase the costs.

A logical next step would be to include full report texts in the analysis. However, due to length constraints in most LLMs, directly inputting full documents was not feasible for this research. A solution would be to implement a Retrieval-Augmented Generation (RAG) setup, in which relevant parts of the full report are automatically retrieved and passed to the LLM. This could significantly improve completeness and reduce the number of 'No Answer' outputs.

Lastly, the broader applicability of this approach should be tested outside of the AcICT domain. Many other organizations might face similar challenges with large volumes of textual data that are poorly searchable or unstructured.

Chapter 6

Conclusions

6.1 Answers to the research questions

To what extent are LLMs capable of extracting and transforming key information from complex, domain-specific reports into classified, structured data?

The experiment conducted in this study showed the potential that LLMs have when they are instructed to extract and transform key information from domain-specific government reports into structured data. When applied to the situational descriptions of AcICT reports, three tested LLMs (OpenAI GPT-4o, Claude-3.5, and DeepSeek-Llama) were able to answer well-formulated, predefined questions with a relatively high degree of accuracy and consistency. The best model, Claude-3.5, had an accuracy of 86.67%. The use of a Chain of Thought prompting further enhanced the interpretability and transparency of model outputs, showing the models were capable of complex text understanding. Additionally, a majority-vote strategy proved to be a potential robust method, performing almost as well as the best individual model. Prompting the models with the DSPy framework also showed a high degree of consistency, with the top model scoring 99.25%.

However, several important limitations tone down this conclusion. First, the models were tested only on the situational descriptions of the reports, which often lacked the detailed information needed to answer all questions. As a result, a substantial portion of the answers returned “No Answer,” not because of model limitations, but due to missing content. Secondly, it should be noted that the accuracy of the models is tested against our own ground truth. Because we only used the situational descriptions of the reports, we could not always be 100% sure about our own answers.

6.2 Contributions and overall significance

What makes this thesis different within the field of natural language processing is its practical focus: we applied LLMs to real-world, domain-specific government documents. Most existing research tends to highlight general capabilities or theoretical benchmarks, but this study explores a concrete use case: interpreting AcICT reports on Dutch governmental IT projects.

From this applied perspective, we discovered several key insights. First, LLMs, especially Claude-3.5, can accurately perform classification without requiring specific training data for the task. Second, our experiments confirmed the critical role of prompt formulation: small changes can lead to substantial differences in model outputs, emphasizing the need for careful prompt engineering. At the same time, when conditions remain constant, the models showed remarkable consistency, making them viable in controlled environments. Third, integrating the DSPy framework provided a clear, modular, and reproducible approach for structuring prompts and reasoning.

Beyond these technical insights, the broader impact of this research lies in its practical value. Our results show that LLMs are powerful tools for information extraction and policy analysis, effectively transforming unstructured text into structured, analyzable data. This is increasingly important given the rising demand for transparency and accountability in public digital infrastructure. Considering

the vast number of government reports produced, automating metadata extraction could significantly improve transparency.

Additionally, this study demonstrates that effective results can be achieved even with limited technical and financial resources and without labeled training data. These are substantial improvements over traditional text mining methods. For the CRISP-DM workflow, this means that while conventional models require extensive preprocessing and parameter tuning, modern LLMs can perform a variety of tasks efficiently. Making text mining more accessible and easier to implement.

Bibliography

- [1] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saied Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys Kochut. A brief survey of text mining: Classification, clustering and extraction techniques. *arXiv preprint arXiv:1707.02919*, 2017.
- [2] 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.
- [3] Aaron M Cohen and William R Hersh. A survey of current work in biomedical text mining. *Briefings in bioinformatics*, 6(1):57–71, 2005.
- [4] Shilpa Dang and Peerzada Hamid Ahmad. Text mining: Techniques and its application. *International Journal of Engineering & Technology Innovations*, 1(4):22–25, 2014.
- [5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.
- [6] Louie Giray. Prompt engineering with chatgpt: a guide for academic writers. *Annals of biomedical engineering*, 51(12):2629–2633, 2023.
- [7] Groq, Inc. Deepseek-r1-distill-llama-70b model documentation. <https://console.groq.com/docs/model/deepseek-r1-distill-llama-70b>, 2024. Accessed: 05-16-2025.
- [8] Andreas Hotho, Andreas Nürnberger, and Gerhard Paaß. A brief survey of text mining. *Journal for Language Technology and Computational Linguistics*, 20(1):19–62, 2005.
- [9] Adviescollege ICT-toetsing. Advisory council on it assessments (acict), 2025. URL <https://www.adviescollegeicctoetsing.nl>. Accessed: 2025-02-10.
- [10] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.
- [11] Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, et al. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.
- [12] Christine Largeron, Christophe Moulin, and Mathias Géry. Entropy based feature selection for text categorization. In *Proceedings of the 2011 ACM symposium on applied computing*, pages 924–928, 2011.
- [13] Bonan Min, Hayley Ross, Elinor Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2):1–40, 2023.
- [14] Harsha Nori, Nicholas King, Scott Mayer McKinney, Dean Carignan, and Eric Horvitz. Capabilities of gpt-4 on medical challenge problems. *arXiv preprint arXiv:2303.13375*, 2023.

- [15] Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*, 2023.
- [16] Martin F Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
- [17] Martin Rajman and Romaric Besançon. Text mining: natural language techniques and text mining applications. In *Data Mining and Reverse Engineering: Searching for semantics. IFIP TC2 WG2. 6 IFIP Seventh Conference on Database Semantics (DS-7) 7–10 October 1997, Leysin, Switzerland*, pages 50–64. Springer, 1998.
- [18] Christoph Schröer, Felix Kruse, and Jorge Marx Gómez. A systematic literature review on applying crisp-dm process model. *Procedia Computer Science*, 181:526–534, 2021.
- [19] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge, 2008.
- [20] DSPy Team. Dspy: A framework for programming with foundation models. <https://dsp.ai>, 2024. URL <https://dsp.ai>. Accessed: May 25, 2025.
- [21] Yuen-Hsien Tseng, Chi-Jen Lin, and Yu-I Lin. Text mining techniques for patent analysis. *Information processing & management*, 43(5):1216–1247, 2007.
- [22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [23] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- [24] Rüdiger Wirth and Jochen Hipp. Crisp-dm: Towards a standard process model for data mining. In *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, volume 1, pages 29–39. Manchester, 2000.
- [25] Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, Yang Wang, and Enhong Chen. Large language models for generative information extraction: A survey. *Frontiers of Computer Science*, 18(6):186357, 2024.