ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

Máster en Big Data: Tecnología y Analítica Avanzada

Development of an NLP Toolbox for Enhancing Productivity in Strategy Consulting

Autor
Carlota Monedero Herranz

Dirigido por
Mónica de Santos Rodríguez

Madrid
June 2023

Carlota Monedero, declara bajo su responsabilidad, que el Proyecto con título **Development of an NLP Toolbox for Enhancing Productivity in Strategy Consulting** presentado en la ETS de Ingeniería (ICAI) de la Universidad Pontificia Comillas en el curso académico 2022/23 es de su autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: .... Fecha: ..11.. / ..06.. / ..2023..

Autoriza la entrega:

EL DIRECTOR DEL PROYECTO

Mónica de Santos Rodríguez

Fdo.: .... Fecha: ..11.. / ..6.. / ..2023

V. B. DEL COORDINADOR DE PROYECTOS

Carlos Morrás Ruiz-Falcó

Fdo.: .... Fecha: ...... / ...... / ....

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

Development of an NLP Toolbox for Enhancing Productivity in Strategy Consulting

1º. Declaración de la autoría y acreditación de la misma.

El autor D. Carlota Monedero Herranz **DECLARA** ser el titular de los derechos de propiedad intelectual de la obra: Dev. of NLP Toolbox for Enhancing Prod. in Strategy Consult., que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, los derechos de digitalización, de archivo, de reproducción, de distribución y de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

(a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.

(b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.

(c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.

(d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.

(e) Asignar por defecto a estos trabajos una licencia Creative Commons.

(f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente*).

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

(a) Que la Universidad identifique claramente su nombre como autor de la misma

(b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.

(c) Solicitar la retirada de la obra del repositorio por causa justificada.

(d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

(a) El autor se compromete a:

(b) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.

(c) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.

(d) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

(e) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a ..11.. de ......junio........ de 2023.

ACEPTA

Fdo.: ....

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
(ICAI)

Máster en Big Data: Tecnología y Analítica Avanzada

Development of an NLP Toolbox for Enhancing Productivity in Strategy Consulting

Autor
Carlota Monedero Herranz

Dirigido por
Mónica de Santos Rodríguez

Madrid
June 2023

Resumen

Los últimos avances en el campo del Procesamiento de Lenguaje Natural (NLP) han demostrado tener el potencial de revolucionar la forma en la cual las compañías operan. Actualmente, numerosas industrias están cambiando el foco de su trabajo hacia la integración de tecnologías basadas en Modelos de Lenguaje (LLMs) como forma de aliviar la carga de trabajo de sus empleados y aumentar su productividad.

Sin embargo, a día de hoy existen ciertas preocupaciones que dificultan la adopción cpopleta de estas tecnologías. Algunas de las más importantes incluyen la falta de transparencia en el contenido empleado para generar la respuesta, la posibilidad de que el modelo cometa *halucinaciones* o el uso de información no actualizada por parte del mismo. Adicionalmente, muchas de las aplicaciones desarrolladas en la actualidad carecen de un marco de integración unificado para las herramientas desarrollaadas, lo que dificulta su adopción práctica en un entorno empresarial.

EL presente trabajo está enfocado hacia el desarrollo de una serie de herramientas basadas en el uso de tecnologías de Procesamiento de Lenguaje Natural con aplicación en el campo de la consultoría estratégica, así como el desarrollo de un marco unificado de utilización de las mismas. El potencial uso de dichas herramientas se muestra a través de cuatro casos de uso diferentes, en los cuales se demuestra la versalidad, flexibilidad y potencia tanto de las herramientas descritas como de sus integraciones conjuntas.

Abstract

Recent advancements in the field of Natural Language Processing (NLP) have the potential to revolutionize the way companies work. Many industries are now shifting towards relying on Large Language Models (LLMs) as a means for alleviating the workload of employees and enhance overall productivity.

However, there are still certain concerns that hinder the full adoption of such technologies, such as lack of transparency on the content used, *hallucinations* or outdated information used by the models. Moreover, existing solutions lack comprehensive frameworks that integrate multiple NLP technologies to harness their collective capabilities effectively.

The present work is aimed at developing a set of NLP-based tools to be used in the field of strategy consulting, as well as a unified framework to work with them. We showcase the potential usage of these tools in up to four different real world use cases, proving the versatility and power of the described tools and pipelines.

By addressing the limitations of current approaches, this dissertation aims to enable companies to leverage the full potential of NLP advancements in their strategic decision-making processes, ultimately boosting productivity and achieving better outcomes.

To those who believed,
I'm here because of you.

I ponder of something terrifying,
because this time there is no sound to hide behind.
I find over the course of our human existence,
that one thing consists of consistency,
and it's that we're all battling fear.
Oh dear, I don't know if we know why we're here,
oh my, too deep, please stop thinking,
I liked it better when my car had sound.

Twenty One Pilots

Agradecimientos

Esta sección es muy personal para mí. Se trata de una carta de amor a todas las personas que me acompañan en el día a día en mi viaje y dan pequeños pasitos a mi lado. Te ruego que si no eres parte de esas personas no leas esta carta. No es para ti.

En primer lugar, a mi madre, por siempre estar allí aunque sea en la distancia. Por ser mi estrella, mi faro en la oscuridad, la luz de mi vida y la persona que siempre me ha guiado. No hay palabras en el mundo que sean capaces de expresar lo que siento por ti. Eres el amor de mi vida y por eso solo puedo decirte... Gracias, mamá.

En segundo lugar, a mi padre, por mostrarme que el mundo es más que dinero y que la felicidad no se encuentra en los números, sino en las pequeñas cosas. Por enseñarme a no compararme, a seguir mi camino y, en última instancia, a mantenerme fiel a mi misme... Gracias, papá.

En tercer lugar, agradecer a mi abuela, por su paciencia y sus ganas de vivir. Sé que no soy la persona más fácil de tratar y que en ocasiones mis emociones sacan lo peor de mí, pero ella siempre está ahí, animando desde el fondo, aún cuando el partido ha acabado y yo haya perdido. Por estar orgullosa de mí, incluso cuando no hay que estarlo... Gracias, abuela.

En cuarto lugar, a mi hermano. Por ser el payaso que es y por hacerme reír todos los días. Por convivir conmigo, por compartir espacios y por mostrarme que no siempre tienes que ser perfecto para ser feliz; tan sólo tienes que ser tú mismo. Por ser único y por querer siempre estar a mi lado... Gracias, Gonzalo.

En quinto lugar, agradecer a Andrea. Pocas veces se tiene la oportunidad de conocer a una persona así y espero que Gonzalo sea consciente de ello. Por mostrarle a la adversidad tu mejor cara. Por decir siempre la verdad aunque duela. Por apoyarme de verdad y no por asociación... Gracias, Andrea.

Finalmente, agradecer a mis amigues por todo lo que han hecho por mí. Elles saben mejor que nadie lo duros que han sido estos últimos meses. A Tivi, por ser una loca perdida y sacarme siempre una sonrisa. Porque aunque todo vaya mal ella sigue adelante y me empuja a seguir con ella. Porque se merece lo mejor y todo lo bueno que le pase. A Javi, por quererme tanto. Por animarme con sus cotilleos y sus dramas. Por querer estar siempre a mi lado, aunque duela. Por demostrarme que el odio y el amor son dos caras de una moneda que me alegro de haber podido voltear. A Diego, por ser una luz de bondad en mi camino. Por ser una persona tierna y amable, que se ríe de la vida aunque la vida se ría de nosotros. Por pasarnos horas al teléfono hablando de absolutamente nada. A Carmen, porque aunque no siempre nos veamos siempre es como si nunca nos hubiéramos separado. Por ser tan ella y ponerse el mundo por montera. Por lo mucho que la admiro y lo mucho que la quiero. A Amparo, por entenderme y querer ser un poicito conmigo. Por ser un espacio seguro y por enseñarme que la vulnerabilidad es también hermosa. Por los lazos que nos unen y lo mucho que significa para mí poder llamarle amigue. A Marce, por ser una mujer tan sumamemente increíble y que me inspira tanto. Por ser una luchadora nata y por hablar siempre con honestidad, amor y valentía. Por su pasión, su bondad y su fuerza. A todes, solo puedo deciros...

Gracias.

Carlota Monedero
Madrid
June 11, 2023

Contents

List of Figures

List of Tables

Code Listings

Acrónimos

AI	Artificial Inteligence
ANN	Artificial Neural Network
BS4	Beatiful Soup
LLM	Large Language Model
LSTM	Long Short-Term Memory Neural Network
MECE	Mutually Exclusive Collectively Exhaustive
ML	Machine Learning
MSMARCO	MicroSoft MAchine Reading COmprehension
NER	Named Entity Recognition
NLI	Natural Language Inference
NLP	Natural Language Processing
NWMD	Negative Word Movers Distance
POS	Part of Speech
RNN	Recurrent Neural Network
STBS	Semantic Textual Similarity Benchmark
USE	Universal Sentence Encoder

Chapter 1

Introduction

1.1 Problem description and motivation

In an era marked by the rapid advancement of technology and the increasing reliance on data-driven decision-making, many fields have witnessed a transformative shift. The availability of vast amounts of textual information calls for the development of innovative tools and techniques to extract valuable insights efficiently. This is particularly true when it comes to professionals that have to work with this type of information on a daily basis, such as strategy consultants.

Natural Language Processing (NLP), a subfield of artificial intelligence (AI), has emerged as a promising field that could help address the challenges faced by these firms in handling large volumes of unstructured data. Since the development of transformer-based models, the rise of more and more powerful technologies is increasingly allowing businesses to automate tasks and increase productivity of their employees.

However, problems such as the lack of transparency over the sources used for extracting information or the possibility of getting false or inaccurate results inadvertently have significantly hindered their adoption. This calls for the development of innovative tools and techniques that allow businesses to use these new technologies to their fullest potential, allowing them to extract valuable insights efficiently whilst having transparency over the means used for extracting them.

The current work is aimed at contributing to solve this problem by designing and implementing a series of NLP-based tools with the objective of enhancing the capabilities that LLM models have to offer. Through the application of cutting-edge techniques and algorithms, this research has sought to leverage the power of NLP in enabling analysts to extract meaningful information from textual sources, such as reports and internet articles. This was done by integrating different pipelines of both information retrieval, content analysis and quality testing.

1.2 Objectives

1. Apply state-of-art NLP techniques to alleviate the potential drawbacks of LLM models.
2. Design a comprehensive framework that can integrate classic NLP and LLMs in a wide variety of applications.
3. Apply said tools to a series of real world problems, in order to prove their potential use in a business setting.

1.3 Document structure

- **Chapter 1: Introduction:** Brief introduction to the topic at hand, including problem description, motivation and objectives, as well as the context of the company.
- **Chapter 2: State of the Art:** Description of the state of the art in the field of NLP and, specifically, Large Language Models (LLMs), to provide a deeper context on the problem and need for the presented solution.
- **Chapter 3: Work Description:** Exhaustive description of the work done, including all of the tools and technologies developed as part of the project.
- **Chapter 3: Experimentation:** Summary of some of the most important experiments done during the project, including the relevant insights and conclusions extracted from each of them.
- **Chapter 4: Use cases:** Description of real world implementations of the work done during a collaborative project with other management consulting agencies.
- **Chapter 5: Future Prospects:** Analysis on future improvements and lines of investigation that are currently open for development.
- **Bibliography.**
- **Appendix A:** Appendix containing further details on experimentation results.

1.4 Context of the company

Bayesian$_x$ is an innovative start-up who works at the intersection of analytical and management consulting. Some of the clients we work with include tel-co, public sector, banking and insurance, among others.

At Bayesian$_x$, we work towards integrating the latest advancements in Machine Learning & Data Science to develop tools that can help clients gain meaningful insights on their company and make conscious, data-driven decisions. Our focus is not on advancing research on the field, but rather integrating state-of-the-art technologies to provide comprehensive tools that are usable by analysts.

Chapter 2

State of the art

The project is developed in the context of Natural Language Processing (NLP), which refers to a branch in Artificial Inteligence (AI) that focuses on the processing of unstructured text data.

Early work on the field on NLP was primarily focused on programatically encoding rules and patterns to provide machines with a rudimental understanding of language. Examples of these include early chatbots such as ELIZA [1] and SHRDLU [2]. However, during the early 2000's research shifted towards adapting statistical models based on Machine Learning (ML) to process text input. This was made possible thanks to the widespread use of the internet, which allowed for an increase in the availability of raw text data that could be processed and modelled in an often unsupervised way [3, 4].

However, during the last decade, increases in computational capacity and parallelization, as well the existence of large corpus of text data have contributed to foster the development of new techniques based on Artifical Neural Networks (ANNs). These models are adjusted using billions of trainable parameters and their architectures are much more sophisticated than traditional Machine Learning models, which allows them to better capture the sense and meaning of human-written text [3, 5, 6]. This approach is known as *Neural Language Modelling* and it involves constructing *a posteriori* probability distributions of the ocurrance of a particular word given the previous n words [7]. This is typically done through a encoder-decoder architecture, including some variation of an attention mechanism that enables the network to take previously seen elements into account when making its predictions [8, 9]. Some of the most well-known architectures include Recurrent Neural Networks (RNN) [10], Long Short-Term Memory Networks (LSTM) [11] and Gated Recurrent Networks [12].

The above-described approaches still lacked refinement, as they relied on a sequential modelling of language rather than a full understanding of its internal workings. Furthermore, the need for words to be processed sequentially limited the ability of the models to leverage parallel computing and process long texts, which in turn limited their practical applications to simple and short text processing. To solve these problems, a new model architecture called Transformer was developed. The model abandoned the concept of recurrence and focused on stacking multiple encoders and decoder units which used both self and cross-attention in parallel to model the relationship between words [3, 13]. Their flexible architecture and their open-source nature allows Transformer-based models to be task-agnostic, which gives researches and developers the power to adapt it to a wide variety of problems by means of fine-tuning in specific datasets, in an approach commonly known as *transfer learning* [14, 15].

However, transfer learning still requires usage of relatively large text corpora in order to perform fine-tuning of the pre-trained models, so further efforts were made in order to develop general purpose models that were already trained on a large amount of text data. These models are commonly known as *large-language models* (LLMs) [16]. Some famous examples of LLMs include OpenAI's GPT-2 [17], GPT-3 [18] and GPT-4 [19], Google's T5 [14] and XLNet [20] and DeepMind's Chinchilla [21], among others.

Even though scientific research is still developing ways to improve upon these models, significant efforts are being made towards developing applications that can leverage the power of LLMs to their fullest potential. Examples of such recent applications include OpenAI's ChatGPT[1] and GitHub's CoPilot[2]. LLM-based technologies have the potential of making an impact in a variety of industries, such as medicine, education, e-commerce or construction [22, 23, 24, 25]. However, there are still challenges that limit their implementation in real-case scenarios [16]. Some of the most prominent worries surrounding the adoption of LLMs include:

1. **Hallucinations**: LLMs tend to *hallucinate* (i.e., produce an erroneous answer without any warning.) This can happen as a result of multiples situations, such as limited training dataset and outdated or conflicting information with their internal representations of the world. Hallucinations are unpredictable and appear to be completely reasonable and grammatically correct responses, so they are oftentimes difficult to detect. This is of special concern to businesses, who are reluctant to base important decisions on the output of a black-box model.

[1] https://openai.com/blog/chatgpt
[2] https://github.com/features/copilot/

2. **Lack of visibility**: Since LLMs are trained on vastly large corpus of data from a wide variety of sources, there is no visibility on the sources the model is using to produce a certain output.
3. **Outdated information**: LLMs do not have access to the internet. Although some platforms are trying to integrate live internet access through means of plug-ins[3], there is no real guarantee that the model uses the plug-in unless specifically instructed to. Moreover, this integrations can only be used from the UI platform and cannot be accessed from outside through an API.

The above limitations imply that any output produced by a LLM would need a fact-checking that, at the present moment, can only be done by a human. While keeping a human-in-the-loop approach may seem beneficial, the task done by the analyst in this case goes beyond supervision of the content, as it requires active checking of a wide range of internet resources in an attempt to review each of the model's output. The task thus becomes slow and inefficient, reducing the automation of the process and thus diminishing the margin by which productivity is increased.

The aim of the current project is to address these issues by creating a set of applications that leverage of both classic NLP techniques with LLMs in order to provide companies with a powerful yet reliable toolbox that suits their specific needs.

[3]https://openai.com/blog/chatgpt-plugins

Chapter 3

Work description

This section is dedicated towards explaining in detail all of the tools developed during the project. Additionally, the main pipeline in which the different tools are meant to be used is explained, providing diagrams aimed at facilitating its understanding.

3.1 Overview

A summary of the pieces of the project can be visualized in **Figure 3.1**.

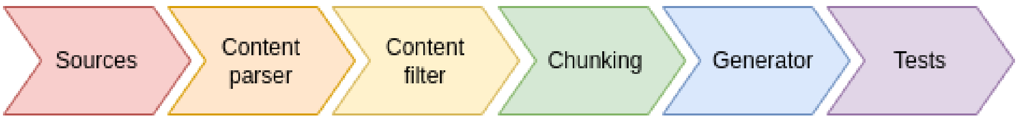

Figure 3.1: Overview of main pieces of the project

The main pipeline in which the project was designed to be used is outlined as follows:

1. Extract information from source.
2. Parse information to text.
3. Filter relevant information from text.
4. Reduce the size of the information pieces (*chunking*).
5. Generate novel content and insights.
6. Test quality of output text.

A global overview of the above described pipeline can be visualized in **Figure 3.2**. Note that the steps indicated using dashed lines are optional and not always required.

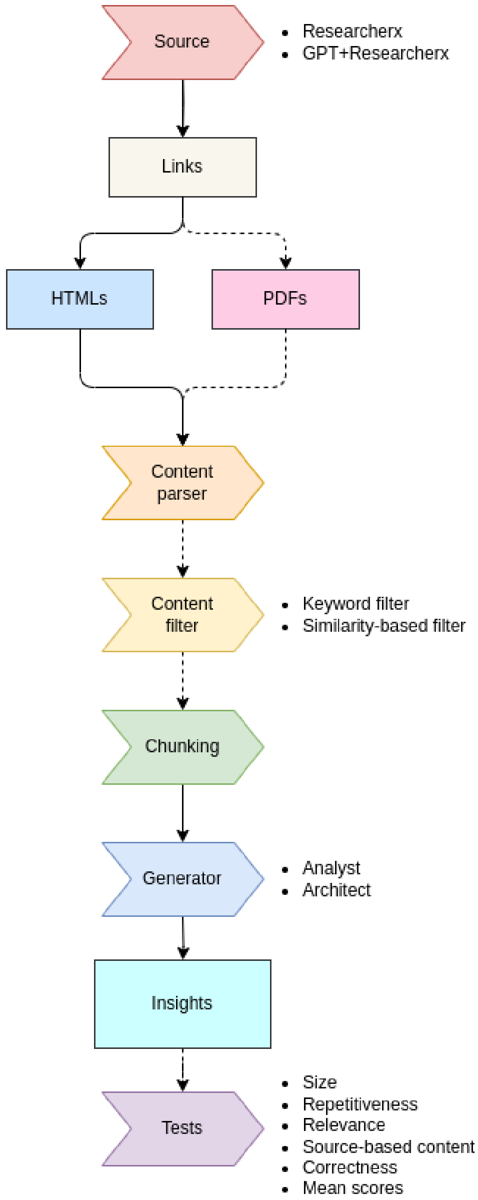

Figure 3.2: Overview of main project pipeline

3.2 Sources

The first step in our pipeline is information extraction. All of the technologies used for extracting information were developed by another team within the company and so the present work will not provide a detailed description of them. However, a brief overview of each of them is provided in the following sections.

3.2.1 Researcher$_x$

As commented before, one of the main limitations of LLMs is the fact that they are unable to provide clear insights on the data used to generate the answer to a particular question. Furthermore, information directly obtained through means of using them requires both clear definition of the questions to be asked as well as a constant interaction with the models in order to obtain more information on a topic.

Thus, for use cases when a large corpus of information from the internet is to be retrieved, the company developed a tool known as *Researcher$_x$*, which semantically searches Google based on a series of keywords in order to retrieve valuable information related to a particular topic. This tools allows us to retrieve a series of links leading to both PDF and HTML documents that contain relevant information that can be later parsed using the **Content parser** technology.

This is the most commonly used source in the project and so most of the pipeline descriptions will start by using it.

3.2.2 GPT + Researcher$_x$

As previously described, the **Researcher$_x$** technology works by using a set of keywords to search content on the internet. This implies that an *a priori* knowledge of the field and the questions to be asked is needed in advance. Furthermore, the quality of the keywords can significantly impact output quality. In order to reduce the margin for error in the first selection of keywords a previous layer of processing was introduced by using GPT-3 [18].

In this pipeline, the user first asks a broad question to a GPT-3 model, which then outputs an answer based on its own knowledge. We then use the content of that answer to extract relevant information (such as keywords or subfacts to be checked) that can be passed on to *Researcherx* to perform an internet search based on them.

Extraction of keywords is mainly done through traditional Natural Language Processing (NLP) techniques, such as Named-Entity Recognition (NER) for extracting relevant dates and places and Part of Speech (POS) Tagging for identifying nouns and verbs of interest. With respect to the subfact extraction, it is mainly done using subsequent calls to a GPT-3 model using a custom-made prompt optimized for this particular task.

3.3 Text extraction

One of the main innovations provided by the current work is the ability to provide users with tools that allow them to get an increased amount of content available for their use, as well as increased visibility over the sources where that content was found. Thus, we designed a series of content parsing, filtering and chunking.

The text extraction steps described in this section are normally applied after an internet search process has been carried out using either one of the technologies described in the **Sources** section, which output a list of links that lead to sources of information that need to be processed.

3.3.1 Content parser

Once the target links have been identified by a source, the next step is to extract the relevant content present in those documents. Our primary goal is to extract as much information as possible from each of the provided sources that we can later refine to keep only the most valuable information.

The first step in this process is to identify and separate links referring to websites (HTML-formatted documents) from those referring to PDF content, which we will do by a simple filtering logic (i.e., identifying whether the link includes a ".pdf" extension on its URI). After this classification step, retrieval of content is done using the *requests* library, which sends a request to the web server for the raw content of the website. Validation tests were implemented to account for different errors that could happen during content retrieval.

Once the content has been successfully retrieved, we proceed with a separate approach to parsing of contents depending on the type of URL, as they both require the use of different technologies. Furthermore, since parsing PDFs is computationally more expensive and time-consuming (due to the unstructured nature of the format, as well as their increased length compared to HTMLs), the extraction of content from this type of document is defined as optional and would require specific instruction from the user to be carried out. Separating processing of both types of documents has the additional advantage of allowing for more control over the amount and type of information desired, as well as the response time.

HTML

In the case of HTML web pages (such as articles, blogs, technical websites, etc.), we will use *Beautiful Soup*, a Python library that allows for a smooth and clean parsing of the raw HTML content. Depending on the amount of information required for the use case, different levels of content can be retrieved:

1. Parse all of the text from the website (including headers and footers).

2. Parse only specific HTML elements (such as `<div/>`, `<li/>`, `<p/>`, etc.).

3. Parse only text that is formatted as paragraph in the HTML (`<p/>`)

The ability to customize the level of information extracted from each HTML allows us to tailor the parsing process to the specific use case and quality of retrieved links. As an example, if we were to parse a relatively low amount of HTMLs in which we are sure of where relevant information is located, we could consider using a more strict parser to extract text in a target fashion and thus reduce the amount of tokens we will later pass on to the LLM. On the other hand, when dealing with a large database of diverse origins where no regular structure can be identified, using a more relaxed parser is highly advised, as it will prevent loss of data during the extraction process. In this case, using the parser in combination with content filtering is strongly recommended, but will once again depend on the specific use case.

Once content is parsed and stored, some post-processing tasks are done to allow for a more natural presentation of the text at hand. Post-processing tasks include removing *unicode* characters and extra spaces, as well as ensuring a uniform encoding of the raw text. Since LLMs are designed to interact with human-readable content and optimized for human interaction, ensuring that the output text is similar to a human-written output is desirable in order to make sure that we are leveraging the technology to its fullest potential.

PDF

In the case of PDF parsing, the selected library to be used is *PyPDF*, an open-source library that allows for clean parsing of PDFs without the need for previous download of files. Similar to HTML documents, after the PDF document is parsed some simple content formatting (such as eliminating *unicode* characters and extra spaces) to allow for a cleaner, more readable format.

in contrast with HTML document parsing, where selection of elements based on the HTML structure could be done to adjust the level of content to be retrieved, PDFs are parsed fully. This, combined with the longer nature of the documents, means that parsing PDFs is more computationally expensive and time-consuming than parsing HTMLs, so this content parsing solution is only implemented when specifically instructed by the user.

3.3.2 Content filtering

The above described approach to text retrieval and content mining allows us to extract a significant amount of text from various sources. However, most of the time the majority of the content of interest will be related to a specific field, so filtering steps become useful for locating information related to a particular topic. Two different approaches to content filtering were developed: **Keyword filtering** and **Similarity-based filtering**. The approach used for both of these methods is described below.

The user is able to choose to apply either of these methods (alone or in combination) to filter text content on a sentence basis. Once the filter(s) are applied over the source, we obtain a subset of sentences either containing the keywords of interest or sentences with a similarity-score above an specific cutoff-value. As a default, the cutoff value is set to select the sentences with a similarity score within the upper 5% of the distribution, although this value can be changed by the user depending on whether they need a more relaxed filtering. If both filters are applied, we proceed to merge the sentences obtained from both steps, eliminating the duplicate entries. This results in a significantly smaller source of information derived from each corpus of text, which we can join to create a cohesive small paragraph.

Although the joining of sentences create a somewhat haphazard paragraph to the human reader, this does not pose a problem for our pipeline, as the main goal is to pass this information to a LLM that can process it to produce meaningful insights.

Keyword filtering

As suggested by the name, keyword filtering is based on the use of a series of words specified by the user to reduce the amount of content parsed while still preserving relevant information. For this approach, a list of single-word keywords is provided and sentences that do not contain any of those keywords are filtered out of the corpus.

Although simple, this approach is computationally inexpensive and can prove to be powerful when users have a clear understanding of the information they are searching for.

Similarity-based filtering

One of the main limitations of keyword filtering is that it requires an *a priori* list of well-defined keywords to perform filtering with, which can lead to loss of information if the keywords selected are not exactly matching the ones present on the text. As an example, if we were to filter content using *job* as a keyword, text containing other words such as *position*, *occupation* or *profession* would be filtered out.

Additionally, the keywords used for filtering need to be one-word, which means it is not a suitable method for keywords comprising multiple words (e.g.: *Artificial Intelligence*, *metropolitan area*, *semantic analysis*, etc.).

Thus, we developed a second approach to filtering based on similarity scores using cross-encoders. Cross-encoders work by directly classifying a data pair based on their similarity without the need for calculating their respective embeddings. Their architecture is transformer-based, and it is comprised of two siamese networks. When presented with a pair of sentences, each of the networks will compute the embedding representation of one sentence. Once the embeddings are produced, an additional classification layer is used to directly calculate the similarity score between them.

Although comparatively slower than bi-encoders [26], cross-encoders are significantly more accurate in their assignment of a similarity score, which yields better results when applied to text filtering.

The model used as cross-encoder is STBS-RoBERTa, which was specifically trained in the Semantic Textual Similarity Benchmark Dataset [27]. The model works by computing the similarity score between each keyword and each of the sentences of the selected source. Once all the scores are calculated, we build a distribution and select only the top 5% from it. The main reason for using this specific approach is that the distribution of scores is highly dependent on the document and hand. Thus, setting an strict boundary (e.g., scores over 0.65) would significantly decrease the amount of information retrieved from some documents, even if they did have valuable information. Adapting the boundary for each of the documents parsed thus allows us to reduce the amount of content retrieved from each of them while ensuring that there is still information extracted at the end of the filtering process.

We designed this approach as a compliment for keyword-based filtering, to be used primarily for multi-word keywords. However, the same technique can also be applied with single-word keywords if the user wanted to relax the constraints on filtering.

3.3.3 Chunking

LLMs tend to have a limitation in the number of tokens users are allowed to pass in each call. In the case of models where prompting is allowed, size calculation takes into account not only the raw content (in this case, the extracted text from the relevant source at hand), but also the size of the prompt template where the text is to be inserted. Furthermore, we have confirmed through different experiments that the global size of the prompt can significantly impact the way the model responds (see **Section 4.1.5** for more information). Prompts that are either too large or too small tend to result in anomalies in model behavior (such as ignoring input text or breaking the formatting laws enforced on the model within the prompt).

Thus, we devised some validation tests used to automatically check the size of the text and chunk it in order to fit it within our pre-established limits.

3.4 Generators

One of the main reasons for the development of this project was to provide users with the ability to automatically extract insights that were relevant to their specific needs. In order to do this, we not only created capabilities that allowed users to extend the amount of content retrieved from different sources, but also designed others that could help them generate new insights by leveraging LLMs. For this purpose, we created a series of LLM-based applications that we collectively grouped under the name of *Generators*

The goal of these Generator pieces is to integrate LLMs in order to generate new content based on some pre-existing content passed to them. In contrast with traditional use of LLMs, in which the user asks a question to the model and blindly trusts the sources that the model is using to answer it, we integrate LLMs as a means of intelligently extracting insights upon pre-existing information. This approach presents a series of advantages when compared to traditional approaches to LLM usage:

1. Increased **visibility** over content used.
2. Increased **control** over response produced.
3. Ability to **test** and check generated responses (since the content from which they were derived is available for inspection).

We developed two different Generators, each of which is geared towards performing a different task:

1. ***Analyst***: The role of the Analyst generator is to extract insights based on pre-existing content that is passed to it as part of a complex prompt.

2. ***Architect***: This generator is in charge of structuring content passed to it into a coherent and cohesive structure.

Each of the Generators use *gpt-3.5-turbo* as LLM model, which is the model used by ChatGPT. We interact with the model directly using API calls through a function that is integrated into the pipeline using LangChain [28], a state-of-art framework that provides a wide range of capabilities for developing LLM-based applications.

LangChain has recently become the preferred library for working with LLMs, as it provides an intuitive API with a high level of abstraction form underlying processes. In addition, it is highly flexible and versatile, allowing users to easily build complex processing pipelines in just a few lines of codes. Finally, it provides additional capabilities, such as access to DocStores and integration with information retrieval tools that may prove to be useful in future steps of project development.

The approach we followed for using Generator capabilities in our project is through means of prompting. In contrast with supervised learning, in which the model is trained to predict an output variable based on an input through means of a probability function $P(y|x)$, prompting-based language models are trained to predict the probability function of the text directly. When asked a question, the input passed by the user is inserted into a pre-defined template into which the model fills the "gaps" in information from which the desired output is obtained. This allows the models to be able to perform *few-shots* and/or *zero-shot* predictions without the need for fine-tuning on a specific corpus data [29].

Few-shot predictions are the ones in which the model is provided with some context or examples on the desired input-output relationship, which it can then use to better learn what the user expects the model to do when presented with new content. For an example of this type of prompt, refer to **Figure 3.3**.

As evidenced in the presented prompt, the model is specifically provided with clear instructions on the desired output, as well as with an example of the type of information to search for. This allows for better control over both the information that the model will provide the user with, as well as the shape in which this information is to be presented. The use case for this type of prompting is situations in which consistency of response is crucial and for which the information to be retrieved is specific and oftentimes mixed with other types of information. It works specially well in cases where non-useful information is similar to useful content, so that clear instructions on how to differentiate both are required.

In spite of its many advantages, using few-shot prompting techniques requires

Given the following content, help us summarize in distinct bullet points the different services offered in this Innovation Center.

This is an example of the ideal output, obtained from a different source:

1. *Brand Vision & New Business Experience with innovative business solutions and smart electronics to elevate businesses to a new level.*
2. *Smart Hospitality Solution featuring SMART Signages and SMART Hospitality Displays to provide guests with a luxurious in-room experience.*
3. *Retails, Food & Beverage Experience with interactive displays to create a personalized shopping experience, and mobile devices to improve flexibility and boost sales.*
4. *Finance & Corporate new vision with video walls and other financial information to attract and retain customers, and enhance employee collaboration and communication.*
5. *School & Office Projector-less solution with interactive displays like Samsung Flip, which streamlines productivity and enables efficient collaboration, and Touch Board solutions to enable business professionals to stay connected at all times.*
6. *Smart Air Con Experience with the 360 Cassette, which creates a comfortable office environment with its innovative circular design that delivers air evenly throughout every corner in any space.*
7. *Technology & Gallery Integrated Experience with "The Frame," an innovative lifestyle TV that transforms into an in-home artwork, and Outdoor waterproof smart signage that enhances living spaces.*

Here is the content I want you to analyze as described above:

{ content }

Figure 3.3: Example of few-shot prompt

Extract the main idea from this text:

{ text }

Summary:

Figure 3.4: Example of zero-shot prompt

a high level of expertise in the field, so that the user is able to clearly specify what they are looking for in an ideal response from the model. In addition, the process of designing a custom made prompt is time consuming, as it implies several iterations on a basic prompt for correcting model response and guiding it towards the desired outcome. Finally, this type of prompting is not reusable, as the level of customization arrived to at the end of the process is high enough so that changing the examples will most likely result in a different response from the model when applied to a different use case.

Thus, for cases in which the user does not have prior knowledge of the matter at hand and they need to extract information to get a general idea of the topic, using zero-shot prompting may be a more suitable approach. In contrast with the previously described approach to prompting, zero-shot prompting is done by giving the model more room to generate answers, without specifying any examples of the desired outcome the user is expecting. Prompts are generally shorter and more intuitive to craft, requiring little to no effort on the user's side.

An illustrative example of this prompting technique can be found in **Figure 3.4**. In this prompt, the model is not given instructions on how to use the provided content to generate content. This results in a much more creative effort by the model, that can come up with a wide range of different responses to the question posed. Moreover, due to the general nature of the prompt, it can be reused in different contexts with little to no modification, allowing for the designed of pre-made prompt templates that can be easily integrated into a content generating pipeline.

The risk of using this approach is that the consistency of answers is at odds with the freedom bestowed upon the model, which is not forced to keep a consistent format across different answers. Thus, the process of generating content in this particular scenario becomes rather stochastic, increasing the risk of hallucinations and disregard of provided content. We suggest using this prompting strategy with moderation and always accompanying it with both automatic and manual correctness and/or fact-checking tests, keeping a critical eye on results.

It is also highly advisable to run the prompt multiple times, to check consistency over subsequent responses from the model to the question asked. This is a suggestion that is specially advisable in the case of zero-shot prompting, but the same can be said for few-shot prompts.

3.4.1 Analyst

As previously introduced, the main role of the Analyst generator is to analyze content passed to it as part of a prompt and revealing novel insights upon the information, much like a consultant or Data Analyst would do.

This specific implementation allows for the use of both few-shot and zero-shots prompts, which makes it highly flexible and adaptable to a wide variety of situations.

3.4.2 Architect

Another capability provided under the Generator section is what we called the Architect. As its name suggests, this piece is in charge of building a *content architecture* by taking relevant pieces of content and structuring them into a cohesive scheme.

The structure we propose is based on two levels of abstraction, which can be visualized in **Figure 3.5**.

1. **Level One**: Level one is comprised of singular statements that capture a key point to be included at the highest level of importance within the structure. It reflects a concept that encapsulates others.
2. **Level Two**: Level two is comprised of multiple statements that are encapsulated under one of the statements in level one, and are thus related to each other. Their role is to provide more information upon the global concept overview that level one is meant to provide.

In order to build this structure, the Architect must be specialized in performing summarization tasks, as well as understanding the relationships between contents and the differences in relative importance of each and everyone of them. In this case, the use of a LLM model proves to be insufficient for the level of refinement of this task, requiring a series of previous additional steps to be integrated into the algorithm, which is outlined in **Algorithm 1** and **Figure 3.6**.

The first step in the pipeline is to produce embeddings for each of the bullet points of extracted information, which is done with the help of a bi-encoder. As previously explained, bi-encoders differ from cross-encoder in both their speed of computation, which is much higher in the case of bi-encoders, making them a more suitable option when there is a large amount of content to be processed. In addition, cross-encoders can only be used in cases where there is a need for comparing two embeddings in terms of their similarity. Since in this case we only need to compute the embedded representation of the information provided by each of the bullet points of extracted information, the use of cross-encoders is not a valid option, so we decided to compute them using a bi-encoder algorithm. In particular, the all-MiniLM-L6-v2 model was selected for this task, which is specifically recommended for tasks in which there is a need for encoding either sentences or short paragraphs, the type of information we will be working with.

After the embedded representations of the information are computed, we applied a community detection algorithms in order to find regions where there is a high density of information. In contrast with classical unsupervised clustering algorithms, community detection algorithms are specifically designed for working with

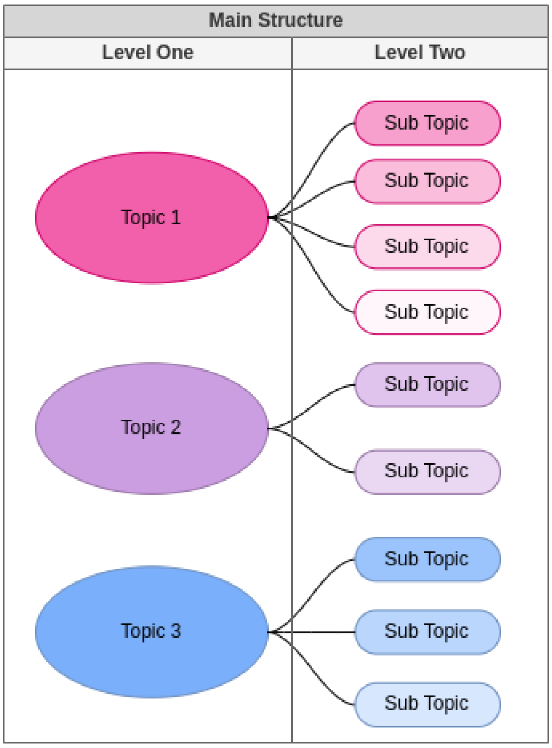

Figure 3.5: Two-level structure built by Architect.

Input : Extracted Information
Output: Cohesive Structure

Step 1: Encode the extracted information into embeddings using a bi-encoder;
embeddings ← encodeInformation(extractedInfo);

Step 2: Cluster the embeddings using a community detection algorithm;
clusters ← clusterEmbeddings(embeddings);

Step 3: Retrieve clusters and their centers;
clusterCenters ← retrieveClusterCenters(clusters);

Step 4: Use gpt-3.5-turbo model to create a cohesive structure based on the cluster centers;
cohesiveStructure ← generateStructure(clusterCenters);

Output: Cohesive Structure (cohesiveStructure);

Algorithm 1: Architect Algorithm

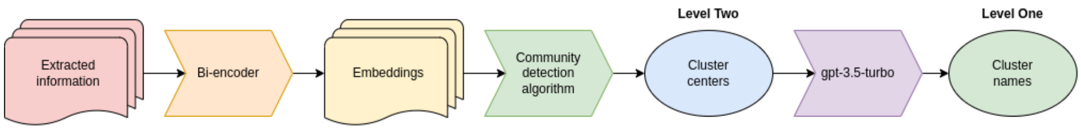

Figure 3.6: Diagram of Architect Algorithm

data that is related to each other in networks and graphs, making them a more suitable approach to use in this scenario, where we are trying to find communities of embeddings that are related to each other in some way. The community algorithm used finds regions with high density (i.e.: a cosine similarity higher than a set threshold) and it then returns the center points of said clusters in addition to all embeddings that lie within their proximity in the embedding space.

Key hyperparameters to be set in this case include the minimal community size (i.e.: the minimal number of embeddings that ought to be found together to consider it a cluster) and the similarity threshold. Both will depend on the use case and can be modified according to user needs. Larger values of the minimal community size will likely result in larger clusters that can group non-related concepts. However, values of this parameter that are too low can render a large number of clusters that fail to capture relationships between similar concepts, instead segregating them into different regions. The same can be said for the set threshold of similarity: too strict a threshold will likely result in the inability of the algorithm to capture related concepts, whereas a similarity threshold that is excessively low can render clusters with unrelated concepts grouped together. It is important to note that any sentence that lies outside of the set threshold will not only not be included into a cluster, but also excluded from entering another cluster if there are not enough sentences to create one, so a careful selection of both of these hyperparameters is advised to prevent loss of significant information when using this technique.

Once the embeddings are clustered together, the centers are retrieved to create the second level of the structure and mapped back to obtain the original sentences, which will be used as part of the prompt that is later passed to the gpt-3.5-turbo model.

The cluster centers are now used as part of a custom prompt that is passed to the generator, in order to create the first level of the structure. Obtaining this level requires the integration, reorganization and effective summarization of related key points (provided by the sentences) into both a list of bullet points and a high-level statement that encapsulates their meaning. In this case, since the task at hand is fairly straightforward and providing a swift response to each of the cases was of paramount importance, we used a zero-shot prompt in which we simply instructed the model on the task it needed to perform as well as the required output format.

3.5 Tests

As previously explained, one of the main limitations of using LLMs on a production setting is that their outputs have to be constantly monitored by humans, so as to ensure that they abide by quality standards. In an effort to alleviate the pressure on analysts and contribute to the automation of the revision process, we designed a series of tests that can be automatically applied over any LLM output to ensure that it complies with the expected quality standards. Since there is yet to be a standardized way of defining quality guidelines, we decided to draw inspiration from standards applied in the field of strategy consulting, including principles such as concision, mutual exclusivity of ideas and correctness of statements.

3.5.1 Size

One of the main characteristics of responses produced by LLMs is their relatively large size, which oftentimes include information that is not relevant to the question posed (such as introductory paragraphs or end summaries). The same can be said for individual statements (or *bullet-points*) within a response, where verbose words and repetitive information is included to provide a more formal response.

To help automatically flag verbose responses, the first approach was devising a simple test that checks whether the response produced by the model is within imposed limits. Specifically, the test is comprised of three metrics:

1. **Total number of characters**, including all bullet points within the response. We defined a *character* as any alphanumeric character (including underscores).
2. **Total number of characters** in each **bullet point**.
3. **Average size** of bullet point.

Since the flexibility of the limit will mostly depend on user preference and use case, it is a parameter to be determined by end users.

3.5.2 Repetitiveness

As stated before, one of the main sources of lengthier and/or bad quality output is the repetition of ideas across different sentences within the response. When presenting content to a client, output quality requires that ideas abide by a principle known as ***MECE*** (*Mutually Exclusive, Collectively Exhaustively*, so the use of LLMs that have not been tested for repetitiveness will require content revision and reformatting by the analyst.

To help diagnose output quality in advance, a repetitiveness test was designed. The test is optimized for screening *bullet-point* formatted output, ensuring that the level of similarity between each of them is below a certain threshold.

The first step in running the test is generating a grid that allows for the comparison of bullet points in a two-by-two fashion. Since the test has to be light in execution, it was considered that generating combinations of bullet points was the optimal approach, as it avoided repetitions between them.

Once generated, each pair is screened to test for the similarity between its respective bullet points by using a combination of two metrics: the USE_cos metric and the STBS-RoBERTa metric. The USE_cos metric is obtained by computing the embedding representations of each pair of bullet points using the Universal Sentence Encoder [30]. After the embeddings are computed, the cosine similarity between both of their vector representations is calculated in order to obtain a measurement of how similar both sentences are. On the other hand, the STBS-RoBERTa metric implies the use of a cross-encoder (see **Similarity-based filtering**) to calculate the similarity between both sentences. Once calculated, both metrics are compared to one another to assess the level of discrepancy between them (see **Section 4.1.3** for more information on the need for this assessment)

Table 3.1: Example of combinations of bullet points

Bullet 1	Bullet 2
Scalability	Efficiency
Scalability	Open-source
Scalability	Multiple data-types
Efficiency	Open-source
Efficiency	Multiple data-types
Open-source	Multiple data-types

and the average of both metrics is calculated as a final score for each pair of bullet points.

After all of the scores are computed, the average score by bullet point is calculated by taking the average of all the pair scores in which the bullet point was evaluated. This average repetitiveness score can then be compared against a user-set threshold in order to identify whether the bullet point is considered to be repetitive or not.

3.5.3 Relevance

Another prominent issue when dealing with content automatically produced by LLMs is the presence of information that does not relate to the question posed by the user. As explained earlier, this can be one of the main factors of a lengthier response, but it can also appear without prior warning when the model is producing a list of bullet points. In this scenario, non-relevant information will appear as an *outlier* bullet point, which either does not directly relate to the question asked (i.e., it is an hallucination) or partially relates to it (i.e., it relates to the topic at hand) but does not contribute to answering the question posed by the user.

An illustrative example can be inspected in **Figure 3.7**. In the example, it is clear that only answer A1 is a valid output to the question at hand. However, the reason why answers A2-A6 are wrong is significantly different. In the case of A2, the answer is related to the topic, but does not answer the question at hand, as it provides a definition of risk factors rather than a concrete example of a disease. Contrariwise, A3 conceptually answer the question (i.e., it provides an example), but it does not relate to the topic at hand. Both A4 and A5 are neither related to the topic nor answering the question. However, the level of "wrongness" of A4 is significantly lower than that of A5, since A4 is an answer related to the healthcare

Q: "Give me an example of cardiovascular associated disease"

A1: "One example of a cardiovascular associated disease is coronary artery disease (CAD)."

A2: "Risk factors for cardiovascular associated disease include high blood pressure, high cholesterol, smoking, obesity, diabetes and a family history of the disease."

A3: "One example of pulmonary associated disease is COPD."

A4: "Pulmonary-associated diseases are a group of respiratory conditions that affect the lungs and respiratory system."

A5: The capital of Australia is Canberra.

Figure 3.7: Example of question-answer pairs

domain.

To account for this inspection, we designed a relevance test to help provide an early detection of such cases. For any given question and a series of answers, the test performs the steps outlined in **Algorithm 2** and **Figure 3.8**.

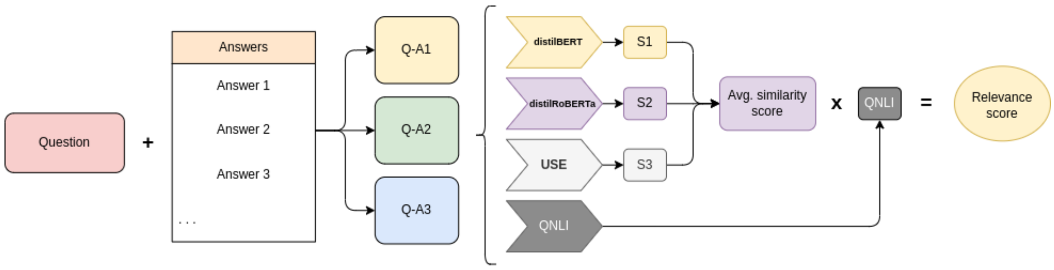

Figure 3.8: Diagram of Architect Algorithm

The first step is used to test whether the answers are relevant to the topic of the question at hand. In order to do this, a combination of three similarity models each used. Depending on the model used, a different approach is followed:

- The STBS-DistilRoBERTa model is a pre-trained Sentence-BERT model which has a Siamese DistilRoBERTa-Network architecture., with two identical DistilRoBERTa networks that share the same parameters. In this architecture, the sentences to be compared are passed through each of the respective DistilRoBERTa models independently to produce their sentence embeddings, which are later compared using a similarity metric. The model

Input : Question-Answer Pairs
Output: Global Similarity Score

Step 1: Use 3 different pre-trained similarity models (namely: distilBERT[31], SBERT-distilRoBERTa[32], USE[30]) to calculate similarity scores for each question-answer pair;

for *each question-answer pair* **do**
    distilBERT_score ← calculateSimilarity(distilBERT, question, answer);
    distilRoBERTa_score ← calculateSimilarity(SBERT_distilRoBERTa, question, answer);
    USE_score ← calculateSimilarity(USE, question, answer);
end

Step 2: Compute the average of the three scores to obtain a global similarity score;

global_similarity_score ← (distilBERT_score + distilRoBERTa_score + USE_score) / 3;

Step 3: Calculate QNLI thematic commonality score [33];

QNLI_score ← calculateQNLI_score();

Step 4: Multiply the average similarity score by the QNLI score;

global_similarity_score ← global_similarity_score × QNLI_score;

Output: Global Similarity Score (global_similarity_score);

Algorithm 2: Relevance test Algorithm was trained using the Semantic Textual Similarity Benchmark [27], which contains pairs of sentences with human-labeled similarity scores that range from 0 to 5, where 0 indicates no similarity and 5 indicates complete similarity. The model is thus trained to predict these labels and the similarity between the human prediction and the one from the model is assessed using the Pearson correlation coefficient. Due to its architecture, the model can be directly used to compute the similarity score between the question and answer pair, as it internally computes the embeddings and outputs the normalized cosine-similarity between them.

- On the other hand, the DistilBERT model is a pre-trained sentence-transformer that uses a transformer architecture model to perform sentence binary classification. The specific model used in this case was trained on the MS MARCO (Microsoft MAchine Reading COmprehension) dataset to perform sentence binary classification. In this case, the model is used for transforming the sentence pair into its respective embedding representations, which are later compared using cosine similarity as a separate step.
- Finally, Google's Universal Sentence Encoder (USE) is a general-purpose pre-trained model that implements slight modifications to the traditional transformer's architecture to allow for adaptation to both contextual and general content by incorporating two different output layer. In this case, the fixed output layer is used to produce embedding representations of both the input question and answer that are later compared using cosine similarity.

The choice of these models was based on both their state-of-art performance on similar tasks, as well as their slightly different approaches to the problem. Through experimentation we could assess that neither of these models performed at the level of precision and consistency required, as there was a clear level of discrepancy between the models (see **Section 4.1.4** for more information). Inspired by the principles of stacking, we decided to combine the output of each of the models to calculate an average similarity score, which allows us for a more conservative and robust approach to the problem, rather than relying on a single model.

However, the similarity score only provides a partial coverage for the problem at hand, since it allows discrimination of sentences based on whether or not they are related to the topic. As previously stated, models can also output answers that are related to the general topic of the question, but that do not specifically answer the question at hand. Thus, a second layer of evaluation is introduced by using a DistilRoBERTa based model to calculate the QNLI similarity score. The model architecture is that of a DistilRoBERTa based model combined with a classification layer which is specifically trained to solve Natural Language Inference

(NLI) problems, in which the task is to determine whether a given hypothesis is true or false based on a corresponding premise. The model first computes the corresponding embeddings of both the answer and the question at hand and then uses a classification layer to determine whether a given sentence answers the corresponding question.

The combination of both the similarity scores and the QNLI score allows us to effectively ponder the scores obtained in the previous step, so that answers that are related to the topic but do not answer the question get a proportionally lower similarity score.

3.5.4 Source-Based Content

Another pressing worry when dealing with LLMs is the fact that they are prone to ignoring the input context and add information based purely on their own internal knowledge, causing a phenomenon commonly known as *hallucination*. Hallucinations often take place when the provided information is conflicting with the internal representation of the world that the model has, a situation that may in turn cause the model to ignore input data and answer solely based on its specific knowledge.

The reason why this is problematic is double. First of all, there is limited to no visibility about the data used for training the model, especially when using private models. Secondly, the information on which the model was trained is likely to be outdated on some topics, as the models do not have the ability to access real-time information. Thus, it is of paramount importance to guarantee that the model is not incurring in hallucinations and that it is effectively using the provided context to extract insights.

In order to assess this, we designed a source-based content test that uses similarity scores to locate the sentences within the original text that are most similar to each of the bullet points that make up the response obtained from the LLM. If the sentences that are found to be most similar to the bullet point turn out to have a relatively low similarity score (below a certain threshold), we could conclude that the bullet point was most likely not produced using the information provided as context, and it is thus an hallucination.

The algorithm can be visualized in both **Algorithm 3** as well as **Figure 3.9**:

Input : Original Source, Bullet Points
Output: Similarity Scores

Step 1: Split both the original source and bullet points into separate sentences;

sourceSentences ← splitSentences(originalSource);
bulletPointSentences ← splitSentences(bulletPoints);

Step 2: Compute the embeddings of each sentence using a bi-encoder;

sourceEmbeddings ← computeEmbeddings(sourceSentences);
bulletPointEmbeddings ← computeEmbeddings(bulletPointSentences);

Step 3: for *each bullet point* **do**

Step 3a: Compute the cosine similarity score between the bullet point and the source sentences;

similarityScores ← computeCosineSimilarity(bulletPointEmbedding, sourceEmbeddings);

Step 3b: Use the cosine similarity score to select the top 5% most similar sentences from the source.;

topSimilarSentences ← selectTopSimilarSentences(similarityScores);

Step 3c: Refine the precision of the similarity score using a cross-encoder;

refinedSimilarityScores ← refineSimilarityScores(topSimilarSentences);

end

Output: Similarity Scores (refinedSimilarityScores);

Algorithm 3: Source-based content Algorithm

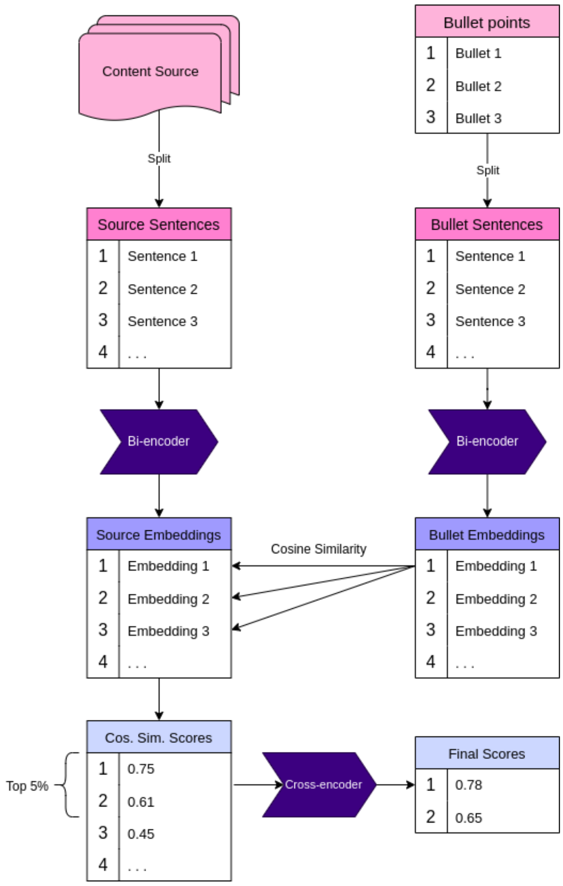

Figure 3.9: Diagram of Source-Based Content Test Algorithm

As presented in the diagrams, for this test we are using a combination of both bi-encoder and cross-encoder to compute the similarity scores. Due to their simple architecture, bi-encoders are able to process large amounts of data in a computationally efficient way, which is important when dealing with potentially long contexts that need to be processed fast so that they do not become a bottleneck within the processing pipeline. However, the simplicity of their design can also provide a limitation to their precision when used for predicting similarity scores between two input sentences. On the contrary, cross-encoder are computationally expensive, but much more precise when it comes to assigning similarity scores. Thus, it is a common practice to use bi-encoders to perform a pre-filtering step in which the assignment of exact similarity scores is not crucial. followed by a re-evaluation step using a cross-encoder, which helps better calibrate the scores on a reduced amount of data.

On the other hand, it is also important to note that the selection of the top scoring sentences is based purely on the similarity score distribution of each sentence of the original source with respect to the bullet point at hand and it is thus not a fixed threshold. This allows us to better adapt the test to the particularities of the response and the source text, so that we can always give at least one most similar sentence, even if the general scores for the bullet point are all relatively low. Our aim is to ultimately keep a human-in-the-loop approach, which allows the user to have control and visibility over the processes and, if desired, judge the output of the test by themselves.

3.5.5 Correctness

Compared to the previously described test, the tests for correctness are significantly more complex, as they require an integration of many of the previously described pieces working together in order to build a cohesive solution.

The motivation behind the use of the test is to provide qualitative and quantitative insights on the veracity of any given statement that is produced by a LLM. As previously explained, LLM are often obscure in the way they produce responses. This stems from two characteristics that are inherent to the way in which these models work:

1. LLMs are trained using a large corpus of information from a wide variety of sources. The size and heterogenicity of the corpora makes it impossible to perform exhaustive fact-checking on the content from the sources, so as to check whether or not it corresponds to reality.

2. Even if the above were not true, the access to the content used for training the above models is restricted, so there is currently no way for a user to check by themselves the information that the model is using to produce the given output. This poses a problem not only in terms of the level of trust the user is able to put on the answer (which is closely tied to the usability of the content), but also to the ability of the user to obtain more information on their own through custom research of the sources upon which the model based the answer.

The above explained facts call for a test that is able to contrast the information that is provided by the model. The objective is double:

1. Provide fact-checking capabilities that are able to distinguish whether each of the subfacts that compose any given statement produced by a LLM is true or not, retrieving evidence that either supports it or refutes it.

2. Retrieve and summarize sources that can expand upon the content provided by the model, so as to give the user an intuition of where the model might have been able to retrieve that information in the first place.

Depending on the level of granularity the user wishes to use when assessing content correctness, two different tests were developed:

1. ***Fact-checking test***
2. ***Subfact-checking test***

Fact-checking test

The most straightforward implementation of the correctness test is to test whether a bullet-point contained within the response of the model is supported by evidence found on the internet or not. Note that we are intentionally avoiding the use of the word "truth" for describing the objective of this test, as we recognize that the fact that any affirmation is is publicly available for consultation does not necessarily mean that the statement is either logically of factually true. However, we do consider that being able to provide evidence on where the statement is located is in itself valuable, as it provides the user with the tools required to perform more in-depth research to fact-check the source and decide whether or not it is worthy of their trust. The algorithm for this test is outlined in **Algorithm 4** and **Figure 3.10**.

The described pipeline can be interpreted as a more powerful extension of the **Source-Based Content** test by combining it with internet-search capabilities.

The first three steps of the pipeline are related to information retrieval. In order to do this, we integrated the test with LangChain. In this particular case, we used the Google Serper Wrapper, which allows for direct interaction with the Google Search API[1]. In order to perform the fastest search possible, the entire fact that is to be checked is passed to the LangChain function, which directly passes it as a query parameter to the Google Search API and handles the retrieval of results from the API. Once the links are located, the contents are classified as belonging to either HTML or PDF and parsed accordingly (see **Section 3.3.1**).

Note that, in this case, no further processing of information is required, meaning there is no filtering of content and/or chunking of information into smaller pieces, so as to preserve as much information as possible. Since we will already be filtering in the following steps of the pipelines, incorporating further filtering may prove to be counterproductive to our goals. On the other hand, reducing the size of the content is not required in this scenario, as we will not be depending on the use of any prompt-based LLM.

[1]https://serper.dev/

Input : Fact, Relevant Keywords
Output: Similarity Scores

Step 1: Extract relevant keywords that summarize the main points from the fact;

extractedKeywords ← extractKeywords(fact);

Step 2: Retrieve links containing relevant information using the previously extracted keywords;

relevantLinks ← retrieveLinks(extractedKeywords);

Step 3: Parse content from links;

parsedContent ← parseLinks(relevantLinks);

Step 4:

for *each source* **do**

Step 4a: Split source into separate sentences;

sourceSentences ← splitSentences(source);

Step 4b: Compute the embeddings of the fact and each of the source's sentences using a bi-encoder;

factEmbedding ← computeEmbedding(fact);
sourceEmbeddings ← computeEmbeddings(sourceSentences);

Step 4c: Compute the cosine similarity score between the fact and the source's sentences;

similarityScores ← computeCosineSimilarity(factEmbedding, sourceEmbeddings);

Step 4d: Use the cosine similarity score to select the top 1% most similar sentences from the source;

topSimilarSentences ← selectTopSimilarSentences(similarityScores);

Step 4e: Refine the precision of the similarity score using a cross-encoder;

refinedSimilarityScores ← refineSimilarityScores(topSimilarSentences);

end

Output: Similarity Scores (refinedSimilarityScores);

Algorithm 4: Fact-checking Algorithm

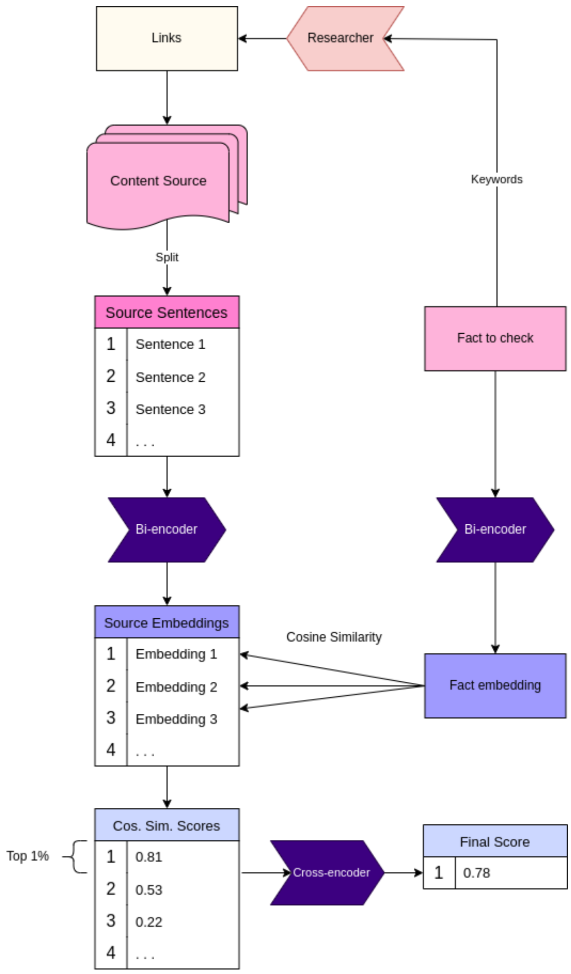

Figure 3.10: Diagram of Fact-Checking Test Algorithm.

After content is successfully retrieved and parsed, a modification of the previously described **Source-Based Content** pipeline is carried out in order to extract the most similar sentences from the source. The major modifications that are incorporated into the previous test's logic include:

1. The computation of scores is done on a **source-level** basis, rather than on a fact-based basis. This stems from the fact that the sources used for checking each of the subfacts are different, which makes it impossible for us to perform the test using the same sources for each of the facts.

2. The cosine similarity threshold is set to a much more strict level (compare 5% against 1%. Note that when performing the **Source-Based Content** we were already given a source of information that the model had to interpret and change, while keeping the meaning intact. Thus, a higher level of freedom

was allowed for the model, as it was understood that it had a certain margin for restructuring content and providing novel formulations on it. However, in the context of fact-checking, we aim to provide the exact sentence that was most likely to be used by the model to provide that response and/or the sentence that better supports the statement the model is giving the user. This calls for a higher level of severity on the way we judge sentences from each of the sources found, so as not to provide the user with a misleading sense of trust on statements that are not evidence-based. However, if the user wanted to reduce the level of automation of the test, they could manually reduce the threshold to match their specific needs and combine it with a manual inspection of the most similar sentences in the original text.

The final output of the test consists on a list of the most similar sentences retrieved for the particular fact that is being checked, as well as both their individual cosine similarity scores and the average cosine similarity score. Additionally, the links to the original sources in which the most similar sentences where found are provided to the user, so that they can further explore them and decide whether they are to be trusted or not.

The main disadvantage of this approach is that computation of the test is fairly slow, as it requires running a whole internet-search pipeline, combined with content parsing and cleaning, as well as computing the embeddings for all of the content that is retrieved from the source. This may not pose a significant problem, as users are aware that the test is time consuming and that the results they will be obtaining will probably not be in real-time. Furthermore, we expect this to be a test that is only run on specific situations, where the user needs to be confident on the quality of the output it produces. Thus, it may not be an intermediate test, but rather a test that is performed as a final check on output that is already processed and ready to use. However, if the user wanted to speed up the process, they may attend to some of the following guidelines:

1. **Avoid PDF-parsing** if it is not strictly required. PDFs are often longer in size and are thus more computationally expensive to process, both due to their parsing but also because of the computation of a larger number of embeddings. We recommend a conservative approach, in which the test is first performed using only information from HTMLs. If the user is not happy with the results or believe that valuable information can be extracted from PDFs, they may re-run the test enabling PDF-search.

2. **Limit the number of sources** used. Since the initial internet-search that is carried out at the beginning of the pipeline has an extensive focus, relevant information is likely to be contained on only the first couple of links retrieved.

Arunachal Pradesh Hydroelectric Development Corporation, India (2009): Land acquisition challenges delayed the 600-MW Kameng hydropower project by three years, causing cost overruns and financial difficulties

Figure 3.11: Example of subfact to be checked

Subfact-checking test

Oftentimes, the level of complexity of the statement to be assessed requires a higher level of granularity in its assessment. An illustrative example of a fact produced by GPT-3 when asked to provide examples on Land Acquisition related issues is presented in **Figure 3.11**.

In order to consider the statement presented in **Figure 3.11** as true, the following conditions ought to be true as well:

1. *Arunachal Pradesh Hydroelectric Development Corporation* is a real company.
2. *Arunachal Pradesh Hydroelectric Development Corporation* is located in India.
3. *Arunachal Pradesh Hydroelectric Development Corporation* faced land-acquisition challenges.
4. The land-acquisition challenged faced by the company resulted in a delay of the *600-MW Kameng hydropower project.*
5. The project was delayed by three years.
6. The delay caused both overruns and financial difficulties.
7. The above-mentioned issue took place in 2009.

In this case, the statement is so complex and nuanced that the information is unlikely to be present in the same source, let alone the same sentence. Thus, if the user wanted to check the subfact they would be in need of a test that would be able to both decompose the fact into subfacts to be checked and test them individually to assess whether there is supporting evidence that can back them up.

Given a sample fact generated by any LLM, the subfact-checking test performs the steps outlined in **Algorithm 5** and **Figure 3.12**.

Input : Fact
Output: Insights on Subfacts

Step 1: Extract subfacts that can be derived from the fact;

subfacts ← extractSubfacts(fact);

Step 2: Extract relevant keywords that summarize the main points from the fact;

extractedKeywords ← extractKeywords(fact);

Step 3: Retrieve links containing relevant information using the previously extracted keywords;

relevantLinks ← retrieveLinks(extractedKeywords);

Step 4: Parse content from links;

parsedContent ← parseLinks(relevantLinks);

Step 5: Filter relevant information from each source;

filteredContent ← filterContent(parsedContent);

Step 6: Chunk each source into pieces of adequate size;

```
for each source do
    chunkedContent ← chunkContent(filteredContent);
end
```

Step 7: Prompt gpt-3.5-turbo to provide insights on whether each subfact is true or not, based on the content from each source;

```
for each subfact do
    for each source do
        insights ← promptGPT(subfact, chunkedContent);
    end
end
```

Output: Insights on Subfacts (insights);

Algorithm 5: Subfact-checking Algorithm

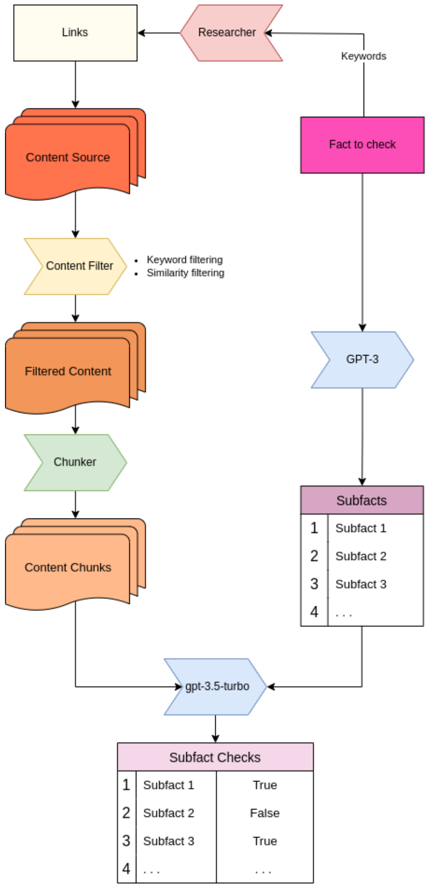

Figure 3.12: Diagram of Subfact-Checking Test Algorithm.

Compared to the previously explained **Fact-checking test**, in this case there is an additional step, which requires us to automatically extract the subfacts that can be derived from the provided fact. In order to do this, GPT-3 was specifically instructed through custom prompting to perform subfact extraction according to a desired output format. The resulting subfacts are not to be used during the information retrieval process, but rather during the model-prompting phase.

Steps two and three of this pipeline are analogous to those previously described for the **Fact-checking test** pipeline. However, there is an important difference in the way they are implemented. Instead of using a lightweight internet search engine, the pipeline starts by performing a mroe exhaustive internet research in order to retrieve potentially useful links that can map to relevant information that supports the fact at hand. In order to do this, a slight modification of the **GPT + Researcher$_x$** source is used, so that the user is not required to ask any question to the model. Rather, we use NLP techniques that allow for the extraction of keywords based on the fact that the user wants to test for correctness. Once these keywords are extracted, *Researcherx* is used to perform an internet search and retrieving relevant links that can potentially contain information related to it.

In contrast with the previous case, in which we used LangChain capabilities to automatically perform internet search, this approach allows us to have more control over the internet search process, as we can always inspect and check the keywords that are being extracted and used for performing the internet search. However, in the future we would like to better address how using LangChain for this particularly step performs when compared to our custom solution, as there are potentially interesting features that it can provide, such as:

1. Retrieval of metadata from each of the links obtained, which can be useful for later selection of the best potential links for the use case.

2. Granular control over the amount of links we want to retrieve, that can be customized for each specific use case.

3. Relatively high searching speed.

4. Clean API, that allows for a high level of abstraction over the internet search process and can be integrated with LLM capabilities using the same library.

Once the desired links are retrieved, the content is classified as belonging to either HTML or PDFs and parsed accordingly (see **Section 3.3.1**). After extracting and adequately parsing information from the sources, the content is filtered using both keyword-based and similarity-based filters on a sentence level. In contrast to the previous implementation of the test, in which we wished to retrieve all the information, the limitations imposed by working with a LLM down the line implies that the content being passed to it needs to be limited in size, both because of the token limitations that the model has as part of its API interaction and also due to the monetary cost of using the model. The filtered sentences from each source are then pieced together into a single text that is chunked to match the appropriate size range of 500-700 tokens prior to continuing the process.

The use of these pre-filtering steps implies that the output content that is to be fed to the LLM may prove to be incoherent, mixing phrases that although relevant are not directly related to one another and/or are presented in separate parts of the document. This is a known limitation of using the pipeline, and opens the door for improvement of the pipeline in the future, assessing the real need for the pre-filtering step and reevaluating how to better improve the chunking process through means of overlapping. However, due to the limited amount of time and resources that we could allocate at the time of development, we chose this alternative, as it provided a first approach to the problem that was coherent and solid.

After the content is processed, we proceed to pass the information to gpt-3.5-turbo model from OpenAI through an API call. The prompt used for performing the API call to the model can be inspected in **Listing 3.1**. This prompt is optimized to be both consistent in the output format as well as being transferable to be used in a wide variety of context and examples.

Code Listing 3.1: Code for interacting with gpt-3.5-turbo model using OpenAI API.

```
def check_subfacts_llm(source, facts):
    response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
    temperature=0,
    messages=[
            {"role": "user", "content": "I will give you a paragraph and a list of
                subfacts. Your job is to check if each subfact is mentioned or
               not in the paragraph"},
            {"role": "user", "content": "Use only the paragraph. Cite the sentence
                in the paragraph where the information is located."},
            {"role": "user", "content": "Answer each subfact separately using this
                format: {subfact}: {True/False}"},
            {"role": "user", "content": "Paragraph: "+source},
            {"role": "user", "content": "List of subfacts to check: " +facts}
            ]
    )

    answer = response['choices'][0]['message']['content']
    tokens = response['usage']['total_tokens']

    return answer, tokens
```

After the model is called on each of the content sources, a global score is calculated to assess the global level of support assigned to the fact the user wanted to check in the first place. The score is calculated as explained in **Algorithm 6**.

Input : Fact, Sources
Output: Final Score

Step 1: Calculate the number of true subfacts for each chunk of each source;

```
for each source do
    for each chunk do
        trueSubfacts ← countTrueSubfacts(chunk);
    end
end
```

Step 2: Calculate the average number of true subfacts by source;

```
for each source do
    averageTrueSubfacts ← calculateAverageTrueSubfacts(source);
end
```

Step 3: Calculate the average support for each fact by source;

```
for each source do
    averageSupport ← calculateAverageSupport(source);
end
```

Step 4: Calculate the global score by averaging all the sources' scores for the complete fact;

globalScore ← calculateGlobalScore(sources);

Step 5: Ponder the final score by multiplying it by a "penalty factor" to account for imprecision in model predictions;

finalScore ← globalScore × penaltyFactor;

Output: Final Score (finalScore);

Algorithm 6: Subfact-checking Score Calculation Algorithm

In order to facilitate the computation of the score, we consider a true assignment by the model to be equivalent to 1, whereas if the model assigns the subfact as "false" the score for the subfact is equivalent to 0.

Note that the last step in the algorithm consists in applying a correction factor to the final score provided. This is due to the fact that, although both the prompt and the chunk size were optimized to maximize consistency in output format, the problem of dealing with LLM is still prevalent in this case. This implies that the risk of the model hallucinating and/or ignoring context input is still relevant in this scenario, so we decided to conduct a series of tests that could assess whether the model is using the provided context to produce insights (see **Section 4.1.5** for more information on experimentation).

Results indicated that accuracy of the model was close to 80% when predicting whether a subfact is true based on its appearance on provided context. However, the level of specificity is low (about 32%), with a False Positive Rate (FPR) of 68%. Combined, these two metrics clearly evidence that the model is prone to either ignore the content provided when providing a response or overestimating the level of relationship that is established between a provided subfact and the context, thus assuming that a subfact is true even if the support provided by the content is minimal. We hypothesize that this could be due to the model being optimized to always try to please humans and make as much effort as possible to provide help when prompted to do so. This could lead to the model being somewhat reluctant to express negatives or being strict and critic in its analysis of content, as it believes that the user would not be pleased with such response and is thus prone to "force" positive answers even if there is little to no support on them.

Keeping this information in mind, we designed a correction factor that could ponder the correctness score and thus put the assumptions of the model used to test, in an effort of keeping the process as honest as possible and not providing the user with a false sense of blind trust, which is precisely the issue we are tackling in this section. The score is set to the level of accuracy observed for the model (0.8), although further experimentation upon the performance of the model would be required to fine tune this number and allow us to adjust it according to the observed results.

Mean scores

As previously mentioned, the correctness tests are advisable in cases where the user wants to be sure that the response from a LLM is indeed based either on the content provided to it (**Source-Based Content**) or content from external sources that are publicly accessible (**Correctness**). However, there is a third case scenario in which none of the above mentioned tests prove to be useful.

There are cases in which the desired output of the model is not strictly a text-based answer, but rather a score or a series of scores that the model is instructed to assign to content. This score can be based on criteria explained to the model through means of few-shot prompting. However, defining a score is difficult in itself, as it requires considerable effort from the human part on clearly defining an evaluation rubric that assigns a certain amount of points to each of the described items. Moreover, there is still no guarantee that the model will follow the instructions on the rubric. If the evaluation criteria end up being too long, the model is likely to either forget them or ignore them when passed an input text as part of the prompt (see **Section 4.1.5** for further details).

Thus, the only suitable option for this approach is to trust the model on its criteria for assigning scores to each of the answers provided. Two issues arise from this approach. Firstly, there is no visibility on the criteria the model is using for assigning the scores. The solution for this is double-checking results provided by the model, which impairs fully automation of the pipeline. Secondly, since the model is not given clear criteria on how to assign the scores, it is not uncommon that it hallucinates and thus produces outputs that are not based on the facts but are a clear invention of the model.

In order to alleviate these issues, we propose a double approach. For the first one, we simply suggest that the user is aware of this limitation and performs a revision on the final output of the model before making decisions on how to use the content. In the second case, we propose a more nuanced solution, that involves the calculation of an average score based on repetitive iteration of the prompting process. Our assumption is that the process of generating responses by a LLM model is stochastic, which implies that the model generates the output with the maximum probability of matching the criteria of answer requested by the user. In cases where the model is sure of the answer, subsequent calls to the model will produce similar results, regardless of the internal criteria used by the model to produce them. However, in cases where the model is unsure of the answer, it will choose the response randomly, which results in little to no consistency over subsequent calls to the model.

Even though this process is difficult to capture when the model is asked to provide text-based responses to a user's request, it is fairly straightforward to calculate when applied to numerical responses, as we can directly measure the level of correlation between the sequences of numbers produced by the model. An outline of the algorithm used can be inspected in **Algorithm 7** and **Figure 3.13**.

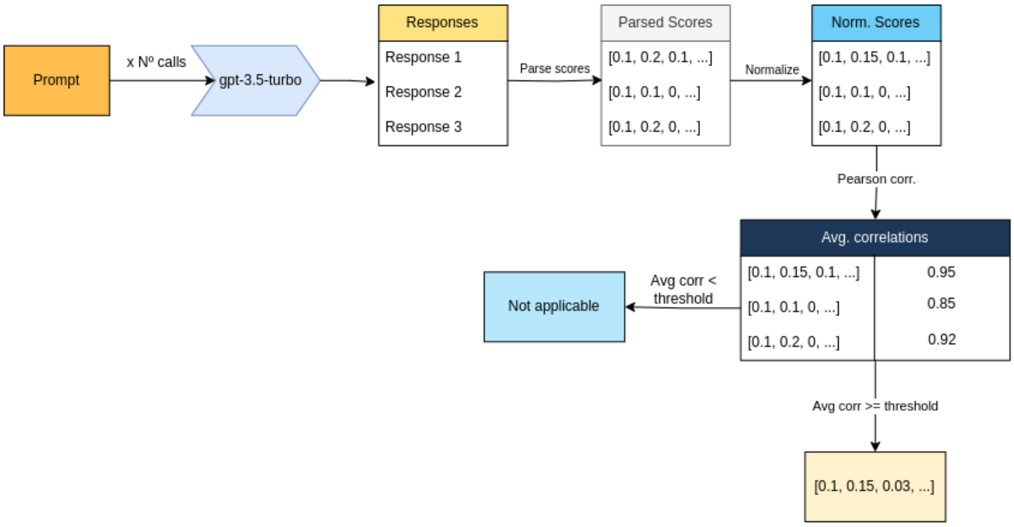

Figure 3.13: Diagram of Mean Scores Test Algorithm

The test starts by performing several subsequent calls to the gpt-3.5-turbo model using the same prompt. The number of calls to be performed is defined by the user and set to 3 by default.

After all of the different responses are collected, the scores are parsed and converted to float numbers. If some condition were to be enforced on the output (such as making the scores sum up to one), the condition is checked in a normalization step.

Once the scores have been normalized, the level of correlation between the sequences is checked using the Pearson correlation coefficient. The score is to be computed for every pair of sequences of numbers and then averaged by sequence, so that every sequence has a unique score to check.

Finally, the global average of the correlation scores is computed. If the average score is above a certain threshold (which, once again, can be set by the user), responses are considered as valid, and the final result sequence is calculated by considering all of the responses. Thus, each number in the final sequence will correspond to the average of the corresponding numbers in each of the responses. In the case where the average correlation is below the set threshold, a default response is given back, indicating that there is a high level of discrepancy.

```
Input   : Prompt, Number of Calls, Threshold
Output: Final Result
Step 1: Perform the call to the model multiple times using the same prompt;
    for i in range(Number of Calls) do
        responses[i] ← callModel(Prompt);
    end
Step 2: Parse the text responses produced by the model, isolate the
  numerical scores, and convert them into sequences of numbers;
for each response do
    numberSequences ← parseResponses(responses);
end
Step 3: Make sure that the sequences follow the specified criteria;
    correctedSequences ← correctSequences(numberSequences);
Step 4: Compute the Pearson correlation coefficient between each pair of
  answers;
    correlationCoefficients ← computeCorrelation(correctedSequences);
Step 5: Calculate the average value of the correlation coefficients;
    averageCorrelation ← calculateAverage(correlationCoefficients);
Step 6: Check if the average value of correlation is above a set threshold;
    if averageCorrelation > Threshold then
        Step 7: If the average value of correlation is above the set threshold,
          compute the final result as the average of subsequent calls to the
          model.;
            finalResult ← calculateAverage(responses);
    end
    else
        finalResult ← "Not applicable";
    end
    Output: Final Result (finalResult);
```

Algorithm 7: Mean Score Calculation Algorithm

Chapter 4

Experimentation

Due to the complex nature of the project and the high amount of pieces that make up the final solution, experimentation and iteration had to be conducted in different areas so as to arrive to the best possible solution for each of the problems that were being solved.

4.1 Text extraction

4.1.1 Content parser

Within the field of *webscraping* in Python, there are two main libraries that can be used: *BeatifulSoup4* and *Selenium*.

BeatifulSoup4 (BS4) provides a package that can download and parse the HTML content from any website. In this approach, the webpage is obtained using another Python library, *requests*, which is in charge of handling HTTP requests to the website server and retrieving the HTML content. Once retrieved, the content is loaded into a BS4 object that allows for retrieval of the different elements within the HTML structure, such as paragraphs, lists, divs, etc. Its main advantage lies in its simplicity and easy-to-debug implementation, that allows the user to design robust pipelines for content parsing. However, it only allows interaction with the static HTML served by the web, meaning it cannot interact with JavaScript components. This poses a problem in situations where interaction is inherent to the nature of the website and thus required for revealing content. A good example of this are social media platforms such as Twitter or Instagram, where the content is revealed using direct user interaction through means of *scrolling*.

Selenium was mainly popularized for automatic testing of web pages developed using Python, as it allows the developer to emulate user interaction with the live website. This feature was precisely what propelled its use in the field of *webscrapping*, by allowing interaction with native JavaScript elements and content navigation. In spite of its potential, *Selenium* proves to be much more fragile to changes in the website structure, as well as considerably more complicated to debug. Moreover, due to the *headless* mode in which it runs on production it may be unable to retrieve content from some webs that implement bot-blocking policies and/or require complex *proxy* server configurations.

Due to its simpler and more robust nature, *BeatifulSoup4* seems like the most obvious choice for extracting content from websites. However, as previously mentioned, the limitation posed by the inability to parse content from certain types of websites could pose a problem down the line. Thus, an experiment was conducted in order to compare the retrieving capacity of each approach. The links were obtained from `https://mindbluff.com/urllist.htm`, a plain HTML website that contains a total of 214 links leading to a variety of websites. Due to the simple nature of the source page, which did not require interaction with JavaScript components for revealing the links, BS4 was used in combination with *requests* to obtain all of the listed links.

Once the links were obtained, both parsers were tested on their ability to extract as much content as possible from each of the websites. The results of the experiment are reflected in **Table 4.1**.

Table 4.1: Comparison of number of webs parsed by content length for each parser

Parser	**Same content length**	**Longer content**
BS4	207	5
Selenium	207	1

As presented in the above table, both BS4 and *Selenium* performed similarly in the vast majority of cases (97%). From the remaining 3% of cases, surprisingly, BS4 was able to extract more information in 2% of the cases, while *Selenium* was only more helpful in 1% of the situations. It is to be noted that the websites used for the experiment were *dummy* websites that are perhaps far too simple. However, it does give a good estimation of the similar capabilities that both parsers can achieve in a base case scenario. Moreover, since we will primarily be dealing with text-based websites such as articles or companies presentation web-pages, the marginal advantage of taking JavaScript into account may not prove to be worth the effort of setting up and debugging *Selenium*.

In conclusion, the choice of BS4 as HTML parser was motivated by the following reasons:

1. **Speed and computational cost**: Since BS4 only implies getting the raw HTML content through an HTTP request, it is much faster and computationally efficient than using *Selenium*, which requires us to build a remote webdriver that needs to be opened and closed for each website we wish to parse. This becomes of paramount importance when dealing with large project that require a high amounts of websites to be parsed, so as not to create overhead for the rest of the processing steps down the line.

2. **Consistency and robustness**: Some interactive elements from a website, such as *cookies* or *ads*, will only be present when a user is directly interacting with it, as they are not embedded in the HTML content that the webserver returns. Thus, *Selenium* may become unable to parse content from such websites due to the specific requirement of interacting with those elements, a problem that BS4 is able to bypass by just parsing the raw HTML content. Due to the heterogenicity of webpages, it is almost impossible to create safeguards that prevent this from happening and/or prepare the webdriver to be able to respond to those elements. Thus, even though some interesting content may be out-of-scope, it seems that the most straightforward option is to choose something that will prove to be consistent and robust across multiple websites and work for most of the cases.

4.1.2 Content filtering

Initially, only the simple keyword-based filtering was implemented as a means for filtering relevant content from websites. This idea was inspired by the way most humans retrieve relevant content from a website. Rather than reading all the content from the article, a human is most likely to perform a targeted search using keywords that they know capture the information that they are looking for.

In spite of being simple and straightforward, this approach presents two limitations:

1. **Selection of keywords**: The keywords selected by the user need to be precise and pertinent to the matter at hand. Not only that, but they also need to be collectively exhaustive, so that they touch on all the matters that are relevant to answering the question. A bad selection of keywords will probably render the information filtering process useless, leading to either loss of relevant information or, alternatively, filtering information that is useless for solving the problem at hand.

2. **Presence of keywords**: It is not uncommon to encounter pieces of relevant information in which the words used are not the ones used for filtering, but close enough so that they can be considered to capture the same meaning.

Consider the following example, in which we are trying to retrieve information from a specific cat breed called *Exotic Shorthair*. Assuming we had links containing valuable information for different cat breeds, using *exotic* or *shorthair* as keywords will probably narrow the information too much, as it discards information that can be referred to the breed in paragraphs that do not contain an specific mention to its name. Moreover, it can potentially lead to the inclusion of useless information belonging to websites that primarily focus on wildcats or other breeds of felines that are indeed *exotic*, but that do not belong to the desired breed. Same logic applies to *shorthair*, which can be easily confused with cats that merely have short hair.

To address this issue, a more sophisticated approach, based on the use of similarity models was proposed. Similarity based models are able to compute the *embeddings* (i.e.: vectorial representations) of words in a high dimensional space that effectively captures their meaning. Within the vectorial space, words or sentences that are similar to each other will lay closer, whilst those that represent concepts or ideas far from each other will be separated. We can effectively measure the level of closeness between pairs of sentences using distance metrics (such as cosine similarity).

In order to quantitatively measure the potential of similarity-based filtering when compared to keyword-based filtering, a series of experiments were conducted. In these experiments, we compared the number of paragraphs that were extracted using each of the above-described approaches. For each of the experiments, gpt-3.5-turbo was asked to provide an example sentence related to a topic and relevant keywords that could summarize that sentence. The choice of topics was done according to three criteria:

1. **Diversity**: The topics had to be diverse, with no chance of overlapping between them.
2. **Expertise**: Each of the topics had to be related to a different field.
3. **Transferability**: The topics at hand had to be realistic enough so that they could represent a real-world problem that a client might be potentially interested in solving.

The results of the experiment are reflected in **Table 4.2**. For more information on the experiments, refer to **Appendix A.1**.

Table 4.2: Comparison of number of chunks filtered by filtering method

Experiment name	Keyword-filtering	Similarity-filtering
Construction	34	12
Saturn	60	28
NER	73	36

As presented in the above table, three different experiments were conducted:

1. **Construction**: This experiment was aimed at retrieving valuable information that could be related to the statement: "The proposed construction of the A-4 highway in Madrid, Spain, was delayed due to land acquisition issues in 2019", which gpt-3.5-turbo produced after being asked to provide an example of land-risk management related issue. The selected keywords for search are collected in **Listing A.1**. Single-word keywords (construction, issues, 2019) were used to perform classic keyword-filtering, whilst multi-word keywords were used to perform semantic filtering (A4-highway, Madrid delayed, land acquisition).

2. **Saturn**: In this case, the statement to be evaluated was "Voyager 1 and 2: These spacecraft were launched in 1977 with the mission to explore the outer solar system. They have since explored Jupiter, Saturn, Uranus, and Neptune, and are now in interstellar space, still transmitting valuable data back to Earth", which gpt-3.5-turbo produced when being prompted for an example of a successful space mission. The accompanying keywords can be found in **Listing A.1**.

3. **NER**: Finally, gpt-3.5-turbo was asked to provide an explanation on *Named Entity Recognition* (NER), from which a sentence was extracted to be used in this experiment: "The first significant work on NER was done in the mid-1990s by a team at the University of Massachusetts Amherst led by Ralph Grishman. They developed an NER system called the Named Entity Recognition System (NERS), which used rule-based methods to identify named entities in text". As always, the selected keywords can also be inspected in **Listing A.1**.

Retrieval of relevant links to test each of the above mentioned sources was done by searching the keywords on Google and manually selecting the three best potential results. Note that automatic link retrieval was not considered appropriate in this testing environment, as it was to be ensured that the sources had enough quality material so that both filtering methods could be tested at the best of their potential. We decided to include both HTML and PDF sources when possible, so as not to bias results to a particular information retrieval method.

In all three cases the keyword-filtering method seemed to perform better in terms of quantitative number of paragraphs retrieved. We hypothesize this is due to evaluation of similarity by paragraph instead of by sentence. Thus, we conducted three additional experiments in which we filtered sources on a sentence level. Results can be inspected in **Table 4.3**. Details on the experiments conducted can be found in **Appendix A.1**.

Table 4.3: Comparison of number of chunks by filtering method

Experiment name	**Keyword-filtering**	**Similarity-filtering**
PPP	19	326
ElasticSearch	60	47
NLP	141	194

As presented in the above table, three different experiments were conducted:

1. **PPP**: In this experiment, gpt-3.5-turbo was instructed to provide insights on how to avoid Latent Defects Risk in Public Private Partnerships (PPP). One of the answers was "Conduct thorough due diligence before entering into a PPP agreement", accompanied by the keywords presented in **Listing A.2**.
2. **ElasticSearch**: For this experiment, gpt-3.5-turbo was asked to provide information on some of the main advantages of using ElasticSearch, to which it responded that "ElasticSearch's ability to scale horizontally allows it to handle an increasing amount of data with ease, making it a great choice for large-scale applications that require real-time search and analytics". The accompanying keywords can be found in **Listing A.2**.
3. **NER**: Finally, gpt-3.5-turbo was asked to provide an explanation on *Natural Language Processing* (NLP), from which a sentence was extracted to be used in this experiment: "Natural Language Processing (NLP) is a subfield of artificial intelligence that deals with the interaction between computers and humans using natural language. It involves the use of algorithms and computational techniques to analyze, understand, and generate human language.". As before, the selected keywords can also be inspected in **Listing A.2**.

Consistent with our previous hypothesis, results show that using similarity-based keyword filtering performs at its best when being done on a sentence-level, rather on a paragraph level. For two out of three of the above-essayed experiments, the level at which the similarity-based filtering method outperforms the keyword-filtering method is quite significant, specially in the case of the PPP experiment. However, it is still relevant to consider that the same could not be said for one of the conducted experiments, where keyword-based filtering still outperformed similarity-based filtering. Whether this is related to the quality of the links provided, or the difference in quality in the selected keywords for each of the test remains to be stated. However, due to lack of time and the need to advance on this area, we did not have time to dig deeper into this matter.

Taking these findings into account, we decided to incorporate both of these approaches as complimentary steps in the content filtering pipeline, so that we could benefit from both approaches and thus reduce the potential amount of information loss during the process.

4.1.3 Repetitiveness Test

When developing the repetitiveness test, it was important for us to first investigate and decide on which models to use in order to produce the similarity score between pairs of sentences. Since different models were trained using different datasets, it was to be expected that their relative performance could vary, making the decision of which model to use important for the use case.

We decided to compare the following scoring metrics:

1. **Negative Word Movers Distance (NWMD)**: This algorithm uses words embeddings to calculate the semantic similarity between two sentences by quantifying the minimum amount of movement required to transform one sentence into the other. The process involves transforming the target sentences into their respective embedding representations and comparing them in the high-dimensional space using a distance metric such as cosine similarity. The assumption made is that two sentences that are dissimilar will require more transformations in order to become equal than they would if their meanings were alike. In this case, we are using the negative version of the metric for interpretability, which leaves us with higher numbers for sentences that are more similar to one another. This model is fairly simple and easy to compute, but it does require prior preprocessing of the sentences prior to its use (such as tokenization, lemmatization and removal of stop words).

2. **Universal Sentence Encoder (USE)**: The Universal Sentences Encoder (USE) was a model developed by Google as a generalist model used to provide embedding generating capabilities. Compared to the previous model, the USE model is more sophisticated, so we expect the comparative performance

- **Scalability**: ElasticSearch is highly scalable, allowing you to easily add or remove nodes as needed to accommodate changes in data volume and usage patterns.
- **Speed**: ElasticSearch is designed to handle large amounts of data quickly, with search results returned in milliseconds.
- **Flexibility**: With its powerful search capabilities, ElasticSearch allows you to search and analyze data in many different ways, making it an extremely flexible tool for a variety of use cases.

Figure 4.1: Bullet points used for testing repetitiveness

to be better. We will use this model as means for encoding our target sentence pairs and then use two different distance metrics, so as to perform comparison in performance between them:

(a) Cosine similarity.

(b) Dot product.

3. **STBS-RoBERTa cross-encoder**: Our final model is a transformer-based model, which we chose due to its different architecture with respect to the previously presented models. As explained in previous sections, the cross-encoder architecture is composed of two siamese networks that work on generating the embbeding representations of each of the sentences in the pair to be compared and directly performs the calculation of the similarity between them. This model does not require preprocessing of sentences, but calculating scores using it is computationally expensive.

In order to compare their performance, we used a series of three bullet points that were generated by asking the gpt-3.5-turbo model to write a list of the main advantages of using ElasticSearch. The produced bullet points can be inspected in **Figure 4.1**

These bullet points were organized into a grid that allowed their comparison in a pairwise fashion prior to applying the algorithms. In the case of the NWDM model, an additional preprocessing of the sentences was done using the following steps:

1. **Lower-casing**: All words were lower-cased prior to their processing, so that the same word may not be interpreted as two separate forms due to one of them being capitalized.

2. **Tokenization**: Tokenization is the process of breaking down a sentence into smaller pieces prior to processing. These tokens may or may not correspond

to words, as sometimes punctuation signs may be interpreted as tokens.

3. **Lemmatization**: Lemmatization is the process of removing the inflexions of a word, so that their dictionary form is obtained. This is used to normalize the form in which words are presented to the model. Compared to other normalization techniques such as stemming, where the inflexion of the word is directly removed, lemmatization is more sophisticated, and guarantees that the words resulting from the normalization process are always real words.

4. **Stop word removal**: Removal of words that are not relevant to understand the global meaning of a sentence, such as conjunctions, interjections, connectors, etc. This process has the advantage of eliminating noise from the input data, as well as speeding up the processing of information by the model.

Results from the application of the models can be inspected in **Table 4.4**.

Table 4.4: Results for Repetitiveness Experiment

Bullet 1	**Bullet 2**	**NWMD**	**USE_cos**	**USE_dot**	**RoBERTa**
Scalability	Scalability	0	1	1	1
Scalability	Speed	-0.92	0.48	0.48	0.47
Scalability	Flexibility	-0.85	0.53	0.53	0.81
Speed	Scalability	-0.91	0.48	1	0.55
Speed	Speed	0	1	0.48	1
Speed	Flexibility	-0.83	0.5	0.53	0.67
Flexibility	Scalability	-0.84	0.54	1	0.79
Flexibility	Speed	-0.84	0.5	0.48	0.57
Flexibility	Flexibility	0	1	0.53	1

From the results presented in **Table 4.4** we can draw some interesting conclusions. NWDM seems to be imprecise when assessing the similarity level between any two sentences, often giving results that are significantly different from the other models. Besides, the interpretability of the score is counter-intuitive, so it may not be an appropriate choice for the use case at hand, in which the analyst is expected to be able to interpret the test results and take actions as required.

We can also see that the distance metric used for calculating the scoring result is also of relevance. Comparing the cosine-similarity score and the dot-product scores obtained using the USE-produced embeddings, we can see that the latter is much more unstable in its performance, oftentimes giving results that are neither coherent with the rest of the models nor with the reality of the similarity between the pairs of sentences. It is also important to note that the dot product does not render symmetric results (i.e.: comparing A against B will result in a different score than comparing B against A). This behaviour can be problematic, as in order to save time the final application of the algorithm is designed to perform comparisons between two bullet points only once, so therefore the use of dot product for calculating similarities between sentences is to be discarded.

Finally, comparing the performance of USE_cos and the STBS-RoBERTa cross-encoder, we can see that the predicted level of similarity between pairs of given sentences is similar in most cases. However, there are some pairs in which there is a significant level of discrepancy between the two scores (for example, the comparison between Scalability and Flexibility, in which the USE_cos score is 0.53 whereas the RoBERTa score is 0.81). Close inspection of results reveals that the USE_cos score tends to behave in a more conservative way, giving scores that are always close to 0.5 regardless of the pair of sentences compared. On the other hand, this very feature makes it more stable in its predictions, providing a good level of consistency across comparisons as well as symmetric results. In contrast, the STBS-RoBERTa score provides more fine-tuned scores that in most cases capture the true nature of the similarity relationship. However, the test can prove to be unstable, showing discrepancies with the more conservative USE_cos metric.

Considering this information, we decided to implement both metrics as part of the repetitiveness test by averaging their respective results when providing a final score. In addition, we decided to incorporate a flag that could help us determine when the results given by the two metrics are discrepant, so that the analyst can further inspect the bullet points and be aware of such behavior.

4.1.4 Relevance Test

As highlighted by the previously explained repetitiveness experiment, when assessing similarity levels between sentences there is a pressing need to incorporate different strategies that are robust and reliable in their own way, as well as combining them through some averaging metric in order to counteract their respective side effects. The same philosophy was applied when designing the relevance test, in which we decided to incorporate averaging of several metrics that were independently calculated so as not to rely on any one of them individually.

We also realized that the similarity-based test only accounted for part of the relevance assessment problem, as it calculated how similar the question and answer were, so as to relate them thematically. However, when asking a model to answer a question, the fact that the answer is relevant to the topic at hand does not guarantee that the answer is adequate for answering the question. Thus, we needed to look for a complementary approach that could help refine the relevance score to account for this requirement.

In order to do so, we chose to assess the potential of the QNLI score for checking whether or not a given answer is likely to be the answer to a question. We designed a total of four experiments, each of them comprised of a question and a set of 4-5 potential answers. For each pair of question-answer in the test, the QNLI model scores how likely the candidate answer is to be the answer to the question at hand. The results from said tests are presented in **Appendix A.3**.

In the first test, the question posed specifically asked for a list of examples on cardiovascular associated diseases. The model was then presented with four types of answers:

a List of examples.

b Definition.

c Implications.

d Factors.

We can see that the QNLI model was able to properly identify the correct answer (a), which shows the highest overall score of all results. On the other hand, responses that correspond to definitions (b) and factors (d) show quite a low scoring, which is consistent with the fact that the question posed was expressively asking for examples. Finally, the answer that corresponded to implications (c) presented a score that lied between the previous ones, probably due to the fact that it contains a long list at then end, which the model may have interpreted as similar to a list of examples.

In the second test, the question related to the main benefits of using NoSQL databases. In this case, the types of questions to evaluate included:

a List of benefits of NoSQL databases, with a detailed explanation of each one of them.

b Synthetic explanation of benefits for NoSQL databases.

c List of benefits of SQL databases, with a detailed explanation of each one of them.

d Synthetic explanation of benefits for SQL databases.

e Example of a NoSQL database.

In this experiment, the model was able to discriminate adequately between benefits (a-d) and examples (e). However, it does not perform well regarding the correctness of the answer. This is to be expected, since answers were too similar for the QNLI model to adequately difference between them, as their only difference is the use of the term SQL versus NoSQL. Note that these two terms are closely related, so they most likely lie close to each other in the vector space of embeddings. This insight is relevant, as it highlights the fact that the QNLI score is not adequate for checking whether an answer is correct or not, but rather whether the answer is a good attempt at answering the question at hand. It is also interesting to see how succinct answers (b, d) perform better than long answers (a, c).

In test three, the question posed was related to a definition of a concept (in this case, the programming language JavaScript). The possible answers the model was given for scoring included:

a Simple definition of JavaScript.

b Extended definition of JavaScript.

c List of advantages of using JavaScript.

d Use case examples of using JavaScript.

In this experiment, we could further corroborate the previous observation of the impact of content length on QNLI model prediction. Even though answers a and b present similar content, there is a certain bias towards the shortest answer to the question (a), which presents a higher score than its long counterpart. On the other hand, the model greatly discriminates between the answer to the question (definition) and the other attempts at answering it with content that was not related to the type of question posed.

Finally, the fourth test evaluated answers for a question asking for a fact (the date of death of a famous singer). The answers presented were:

a Long answer to the fact, including further information that was not related to the question.

b Short answer to the question.

c Description of the person who the fact was related to.

d Long answer to the fact, but for a different person.

e Short answer to the question, but with a wrong fact.

In this case, answers a, b & e scored similarly, since the three of them included dates related to the information asked as well as the name of the person the fact is related to. In the case of answer d, the score is lower, since it does not contain the name of the person that is present in the original question (although it still contains a date). Finally, the lowest score corresponds to answer c, which although giving information on the person the answer is related to does not provide the date of death, which is what the question explicitly asked for.

Overall, analysis of results allows us to draw the following conclusions:

1. QNLI is proficient at differentiating the type of answer expected from question (example/fact/advantages/etc) (Tests 1, 3, 4).

2. QNLI tends to favor succinct answers to long paragraphs, even if they both contain the answer. It is thus good for discriminating when the answer is precise versus when there is additional information that may not be related to the question at hand (Tests 2, 3)

3. QNLI fails to discriminate whether an answer is correct or not based on its content (Tests 2, 4)

The above conclusions prove that QNLI indeed has the potential to work as a compliment for the similarity step in the testing of relevance, as it allows for checking whether the answer is related to the question at hand and penalizes the introduction of additional information into it, favoring precision in responses. Following the MECE principles that guide the elaboration of bullet points, this last features can also be used as a compliment to other tests (such as the **Repetitiveness Test**) that will help further enforce the desired output format on the models at hand.

4.1.5 Correctness Test

In the context of the correctness test, there were several things that needed to be addressed:

1. **Chunk size**: The size of the paragraph that was to be passed as part of the call to the model.

2. **Prompting**: The prompting strategy to be used.

3. **Model accuracy**: How well is the model performing when asked to state whether a subfact is "true" or not based on the given text.

Chunk size

As previously stated, one of the main concerns when working with LLMs is the size limitation they pose in regards to the length of content being passed to them as part of a prompt. The most obvious reason for this is limitation imposed by the API, as the model will simply not accept contents that are above a length threshold. However, through iteration we noticed that LLMs can also exhibit anomalous behavior depending on the size of the input passed, even if it lies within the established limits. This poses an important question to be addressed, as it could significantly impact the quality of the output produce. This is of special concern when dealing with large amounts of data or in cases where uniformity of output is crucial to ensuring correct post-processing of the provided response.

In order to address this issue, we conducted a simple experiment in which we prompted OpenAI's gtp-3.5-turbo model using both a fixed prompt and a context of varying length. The prompt was specifically aimed at giving very specific instructions on desired output format, so that we could evaluate how well the model was able to follow instructions when content size varied. An example of an obtained output can be found in **Appendix A.2.1**

An exhaustive examination of results concluded that optimal size of context to be passed lies between 500-700 tokens. Shorter prompts may result in the model not having enough information to correctly asses the question, which can lead to hallucinations, excessively verbose responses and/or format disregard. On the other hand, larger prompts usually result in complete disregard of response format, probably because the attention mechanism of the model is not strong enough to retain the amount of information and thus stops paying attention to subtle prompting details.

Prompting

In the context of prompting, the first approach followed was aimed at checking whether or not there was a need to incorporate content retrieval and parsing into the processing pipeline used for the correctness test. In order to do this, we designed a series of experiments that were targeted towards checking whether the model could:

1. Access the internet.
2. Understand references to content links.
3. Cite the internal sources it was using to perform fact-checking tasks.

The prompts used, as well as an example result for each of the following tests can be found in **Appendix A.2.2**.

Our first approach was to use a simple prompt (see **Listing A.4**)that could instruct the model to check whether a fact is true or not and cite the sources used. The goal of this prompt was to test if the model could directly access sources from the internet and inspect them, using them to find relevant information that could support the fact to be checked.

Even though the results from this experiment looked promising (see **Listing A.5**), we were unable to validate any of the provided sources of information, as they all referred to links that were invalid and pointed towards non-existing articles.

In the second experiment (**Listing A.6**), we modified the previous prompt to provide the model with an example of the answer format to be used. We hoped that providing the model with a valid source of information would guide it to produce a valid link that we could use to check the information source.

As presented in the example result **Listing A.7**, both the score and the response obtained from this experiment seemed promising at first. However, we soon realized that the link provided by the model could not be accessed, as it referred to an article that did not seem to exist. Thus, it was impossible for us to check the source of information the model was using for checking the subfact.

Based on the above results, it was clear that the model could not be asked to cite the sources it was using to retrieve the information that supported the conclusions it was arriving at, so our next approach was to provide the model with valid links to articles (**Listing A.8**). Even if the model was not able to access those articles in real time through the use of the internet, we hypothesized that it could have some internal reference to those articles and that it would use the information provided in them to perform the task. Results are presented in **Listing A.9**.

Interestingly, the model decided to produce some new sources of information (which, once again, pointed to non-existing domains) whilst also ignoring the information provided by the sources passed on to it, each of which provided relevant insights regarding the checking of the fact.

Based on these information, we concluded that the model was unable to fulfill any of the mentioned requirements on its own, thus proving the need for incorporating content extraction into the correctness test pipeline.

The next set of test was aimed at optimizing the prompting strategy used in order to produce an optimal response format.

As commented in the **Generators** section, there are two main strategies for designing a prompt zero-shot prompting and few-shot prompting. For this particular use case, we leaned more towards the use of few-shot prompting, as there was a need for us to give specific instructions on the desired outcome of the results, so as to keep a uniform format that could be later parsed to give concrete answers to the user.

However, we also had a strict size limitation, due to the maximum amount of tokens that the model can handle in a single API call. Since we had already checked that the optimal chunk size to be passed to the model in order to maintain consistency of output lied between 500-700 tokens, we had to design a prompt that could allow us to instruct the model effectively whilst also passing the content and list of subfacts to be checked without surpassing the established token limits.

This limitation made us opt for a bottom-up approach to prompt design, starting with a omre simple prompting strategy that we could iterate upon. This allowed us to begin the iterating process with a fairly simple prompt that we could then improve using minor modifications designed to target the parts of the model response that were not matching our needs.

Our first experiment in this line was to design a minimalistic prompt, that only instructed the model on the task to be performed (zero-shot prompting strategy). The prompt used can be inspected in **Listing A.10**.

Results from this experiment (**Listing A.11**) revealed the need for prompt optimization in the following areas:

1. **Scoring**: The scoring system used by the model seemed arbitrary, with little to no nuance to it.
2. **Source citation**: The citation of the part of the source used for checking whether the fact is true or not is excessively broad, without any reference to the specific part of the text from which the information was extracted.
3. **Format**: There is little consistency when it comes to the output format of the answers, both in regards to the scoring as well as the explanation given for it.

In order to correct the above-mentioned issues, we designed another prompt using a few-shot strategy, providing the model with a few examples of the desired output format to be used when responding (see **Listing A.12**). In addition, we also decided to eliminate the scoring system from the answer, as the basis of the metric used by the model was not reliable enough and its use could be avoided in this particular case.

In this case, even though the response format was corrected, the source was effectively ignored when producing the answer. The original paragraph of content passed to the model stated: *The twin spacecraft Voyager 1 and Voyager 2 were launched by NASA in separate months in the summer of 1977 from Cape Canaveral, Florida. As originally designed, the Voyagers were to conduct closeup studies of Jupiter and Saturn, Saturn's rings, and the larger moons of the two planets.*, but this information was ignored by the model when constructing the response (see **Listing A.13**). Thus, the use of the few-shot strategy did not prove to be as useful as expected in this case.

Taking these findings into account, we modified the prompt in order to fulfill the following objectives:

1. **Source use**: Clearly state that the model is only meant to be incorporating information from the source provided. This was done both by including an specific instruction within the prompt as well as modifying a model parameter called *temperature*, which indicated the level of freedom the model has when producing responses. A temperature of 0 means that the model will produce more deterministic responses, based solely on the instructed task. This has the added benefit of aiding in the preservation of a uniform format in the responses produced when varying the source.

2. **Format**: Instead of using a few-shot strategy, we shortened our prompt by using a single instruction on the desired output format. This was done both to reduce the amount of tokens used as well as to avoid providing information other than the content source, since we feared it may interfere with the model interpretation of the content to be used when producing the response.

3. **Subfact focus**: We also decided to incorporate subfact checking, rather than passing on a full fact to be checked. This was done in other to provide a more granular response by the model when checking each of the affirmations, which in turned could help citation of the specific lines the model was using as source of information.

The designed prompt can be inspected in **Listing A.14**. As evidenced in the results provided in **Listing A.15**, this optimized prompt is able to fulfill the previously stated goals, enforcing a uniform format of the response as well as forcing the model to use the provided content as the only source of information for crafting responses.

Model accuracy

Once the chunk size and prompting strategy were optimized, a final experiment was conducted in order to assess the level of accuracy of the responses provided by the LLM. Since the motivation of the present work was to analyze the extent at which LLMs could perform by inspecting their responses under a critical eye, it was considered appropriate to also assess the level at which we could rely on their performance when introducing them into out pipelines.

Using the prompt presented in **Listing A.14**, we conducted a series of three experiments. For each of them, we performed the full subfact-checking correctness test pipeline (see **Section 3.5.5**) in order to check the following statements:

1. *NLP works by breaking down human language into its constituent parts, such as words, phrases, and sentences, and then analyzing the relationships between these parts. This involves a range of techniques, including statistical modeling, machine learning, and deep learning.*

2. *One of the key challenges in NLP is dealing with the ambiguity and complexity of human language. For example, words can have multiple meanings depending on the context in which they are used, and sentences can have different interpretations depending on the tone and emphasis of the speaker.*

3. *To overcome these challenges, NLP systems use a range of techniques, such as semantic analysis, sentiment analysis, and named entity recognition, to extract meaning from text and speech. These techniques involve analyzing the structure and content of language to identify patterns and relationships that can be used to make predictions and generate responses.*

The corresponding lists of subfacts to be checked includes:

1. “["NLP works by breaking down human language into its constituent parts.", "The constituent parts include words, phrases, and sentences.", "The relationships between these parts are analyzed.", "The analysis involves a range of techniques.", "The techniques include statistical modeling, machine learning, and deep learning."]“

2. “["The key challenges in NLP", "Dealing with the ambiguity and complexity of human language", "Words can have multiple meanings", "Meanings depend on the context in which they are used", 'sentences can have different interpretations", "Interpretations depend on the tone and emphasis of the speaker."]“

3. “["NLP systems use techniques to overcome challenges.", "The techniques used by NLP systems include semantic analysis, sentiment analysis, and named entity recognition.", "The purpose of these techniques is to extract meaning from text and speech.", "The techniques involve analyzing the structure and content of language.", "The analysis is used to identify patterns and relationships.", "The identified patterns and relationships can be used

to make predictions.", "The identified patterns and relationships can be used to generate responses."]“

The pipeline included retrieval of keywords and links using GPT3 + Researcher (**Section 3.2.2**), as well as both keyword and similarity-based filtering of relevant information (**Section 3.3.2**), followed by paragraph chunking (**Section 3.3.3**). Once the information was parsed, filtered and chunked, each of the sources was passed to the gpt-3.5-turbo model as part of the above mentioned prompt.

After collecting all the information, we were left with a total of 145 statements that were marked as either true or false by the model, as well as a short citation on the sentence of the chunk at hand that it used to verify the subfact. Each of these affirmations was in turn verified manually, which involved reading the chunked paragraph, checking that the citation of the paragraph provided by the LLM model was part of the provided chunk and finally verifying that the assessment of the model on whether the subfact was true or not (based on the chunk of content) was correct. Through this human inspection of results, we could verify that the model was always providing citations of phrases that were indeed part of the content provided, which further validated the quality of the optimized prompt.

We then contrasted the results of the manual inspection against the results produced by the model and organized them into a confusion matrix, which can be inspected in **Table 4.5**.

Table 4.5: Confusion Matrix for Subfact-Checking Results

		Predicted	
		Positive	Negative
Actual	Positive	96	2
	Negative	32)	15

Using this information, the following performance metrics have been calculated:

- $Accuracy = 0.77$
- $Sensitivity = 0.98$
- $Specificity = 0.32$
- $Precision = 0.75$
- $FalsePositiveRate = 0.68$

The above results show that the global accuracy of the model when assessing the correctness of a subfact is of 0.77, close to 80%. The sensitivity of the model, which captures the amount of True Positives (TP) of the model over the real amount of positives is also quite high, which is indicative of a low number of False Negative results. However, the specificity of the model is low (0.32), which means that there is an alarming amount of False Positive (FP) results. This is also evidenced by the high False Positive Rate (FPR), which shows that when the model states that a subfact is true based on the content provided, there is a fairly high chance that the affirmation is not true. However, it is interesting to see that if the model states that the fact is false, there are very few cases in which the statement is true.

These results indicate that the model likely has a low threshold for considering a subfact as true, so that if the least amount of information in the content suggests that the subfact may be true, the model is likely to mark it as so. In contrast, it will only mark the answer as false when there is absolutely no mention of the subfact within the text.

Taking these findings into account, there was a need to create a pondering factor that could allow us to capture this phenomenon, so as to forcefully decrease the level of confidence that the user would have on the affirmations provided by the test. In addition, results also highlighted the importance of using this test with moderation, as a means of helping users get more information on the subfacts to be checked rather than as a source of truth, as more iterations and experimentation still need to be done in order to perfect the performance of this test.

Chapter 5

Use cases

The previously described technologies were tested in a real-world scenario, in the context of a project for one of the top 10 most important TelCo companies, which was carried out in collaboration with some of the top strategy consulting firms in the field. The scope of the project involved a team of 12 analysts working full time for a total of three months, time in which they had to work against the clock to perform at the top of their abilities and deliver results in limited time.

The project was aimed as performing extensive research in regards to the development of innovation center infrastructure, which included the investigation of both the required capabilities that the center aimed to provide as well as retrieving information from potential competitors to get an idea of the best practices to be established in it.

Our role in this project was to provide the analysts with useful tools and insights that could alleviate some of the work they had to do, so that they could focus on creating value for the client. We mainly collaborated with 5 analysts during the first phases of the project, where the level of uncertainty was the highest and, thus, when the analysts needed as much help as possible to reduce the amount of work they had to do by hand in order to move in an agile and effective way. The services we focused on providing included:

1. **Information extraction**: extracting valuable information and insights from a large corpus of validated sources.

2. **Information structuring**: finding a structure that summarizes the previously retrieved information.

3. **Information mapping**: categorizing the extracted information into a predefined structure.

The above services were implemented using the different tools and technologies outlined in **Chapter 3** and applied to four different use cases:

1. **Location** of the innovation center(s).
2. **Services** provided by an innovation center.
3. **Value proposition** of innovation center(s).
4. **Job descriptions** from employees that work at an innovation center.

Each of the above constitutes a crucial piece of information for the end-client, as they need to perform an exhaustive study of the best-practices of their competitors in order to be able to execute their vision as proficiently as possible.

5.1 Location

The first objective was to extract the location of each innovation center from a list of 230 links belonging to innovation centers. These links were extracted through automatic internet search using $Researcher_x$ capabilities (see **Section 3.2.1**). An overview of the process can be inspected in **Figure 5.1**.

Once the links were located, they were classified into either belonging to HTML or PDF and parsed accordingly (see **Section 3.3.1**). During the parsing process, we found that some of the links could not be accessed. After further investigation of the logs, we concluded that this was mainly due to 403 HTTP errors, which means that the server understood the requests for accessing the information, but refused to give access to content, either due to lack of proper credentials or anti-bot measures. This situation rendered us unable to process the information from those websites. The amount of cases that posed this problem only represented a 4% of the total amount of links that were to be processed. Since speed of response was crucial during this phase of the project, we decided to continue on without the information provided by those links, as fixing this issue would require a significant amount of work.

After content from the remaining 221 links was parsed, we then proceeded to use the Analyst generator (see **Section 3.4.1**) to extract relevant information from each of the sources. This was done through the use of a custom prompt that was integrated into a cohesive model pipeline using LangChain capabilities (see **Section 3.4**). The template prompt used for this particular case is found below in **Figure 5.2**.

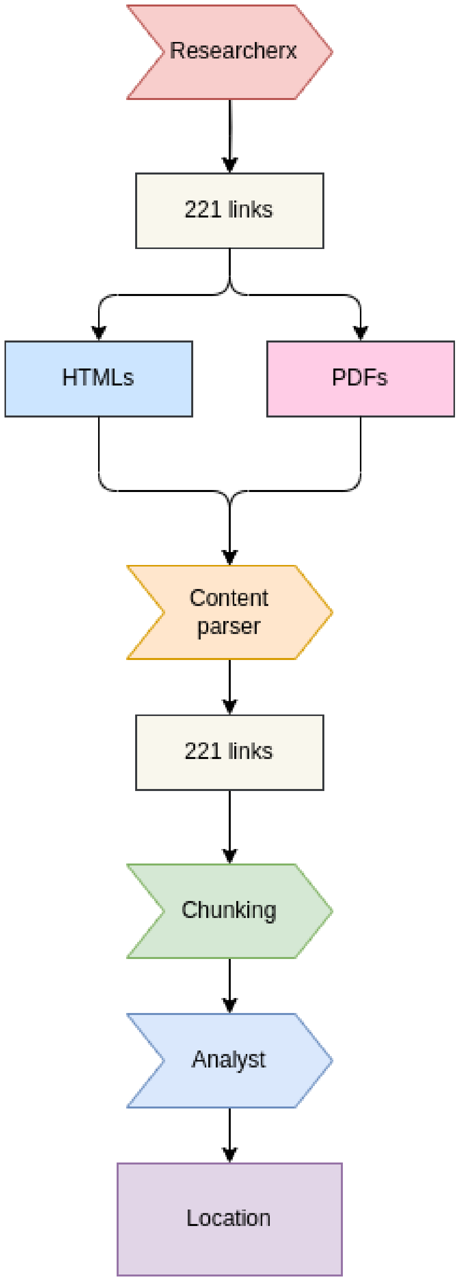

Figure 5.1: Overview of location extraction pipeline

Write an answer for the question below solely based on the provided context. If the context provides insufficient information, reply 'Not enough information'. Answer each question in a separate line.

{context}

Question: Where is the center located?

Answer:

Figure 5.2: Prompt used for retrieving location of innovation center

It is important to note that the model was instructed to only give a definite response when it was sure that the information provided contained the answer to the question, and answer with "Not enough information" when it were not sure about the answer.

Some meaningful examples of results obtained can be found in **Listing 5.1**.

Code Listing 5.1: Information on innovation center location (JSON Array).

```
[
    {
        "name": "RedHat",
        "url": "https://www.redhat.com/en/about/executive-briefing-center",
        "extracted_location": "The center is located in Boston's Seaport District."
    },
    {
        "name": "AWS",
        "url": "https://aws.amazon.com/executive-insights/ebc-executive-briefing-center/\#:~:text=The\%20AWS\%20Executive\%20Briefing\%20Center,Learn",
        "extracted_location": "Not enough information"
    },
    {
        "name": "NetAPP",
        "url": "https://www.netapp.com/company/executive-briefing-center/",
        "extracted_location": "The center is located in multiple locations including San Jose, RTP, Amsterdam, and Bengaluru."
    },
    {
        "name": "T-Mobile",
```

```
        "url": "https://www.t-mobile.com/business/enterprise/executive-briefing-centers",
        "extracted_location": "The center is located in multiple locations across the U.S., including Bellevue, Washington; New York, New York; Overland Park, Kansas; and Peachtree Corners, Georgia."
    },
    {
        "name": "Docusign",
        "url": "https://origin.docusign.com/company/executive-briefing-center/",
        "extracted_location": "The center is located in San Francisco at 221 Main St., Suite 1500."
    },
    . . .
]
```

As presented in the above example, response provided by the model is fairly complete and matches the expected behaviour from visual inspection of the websites' information on innovation center location.

5.2 Job descriptions

Another valuable source of information when helping the client design how an innovation center should look like is the one concerning the roles and preferred profiles of the people that need to work at the center. Job descriptions do not only provide valuable information on which candidates to look for, but they also provide hidden insights on the responsibilities these candidates are required to fulfill during their daily work, which is directly tied to the capabilities the innovation center needs to be able to provide to its clients. A diagram of the pipeline followed can be inspected in **Figure 5.3**.

In order to extract information from this particular source, we followed a similar approach to the one described for extracting locations of innovation centers. Our source of information was a total of 120 links that were automatically found using $Researcher_x$ capabilities. From them, we had a negative response from 6 links that could not be accessed due to a variety of error codes, which left us with a total of 114 links to be parsed and processed.

Once the information was extracted, the Analyst model was instructed to locate information related to the responsibilities described for the roles using the prompt provided in **Figure 5.4**.

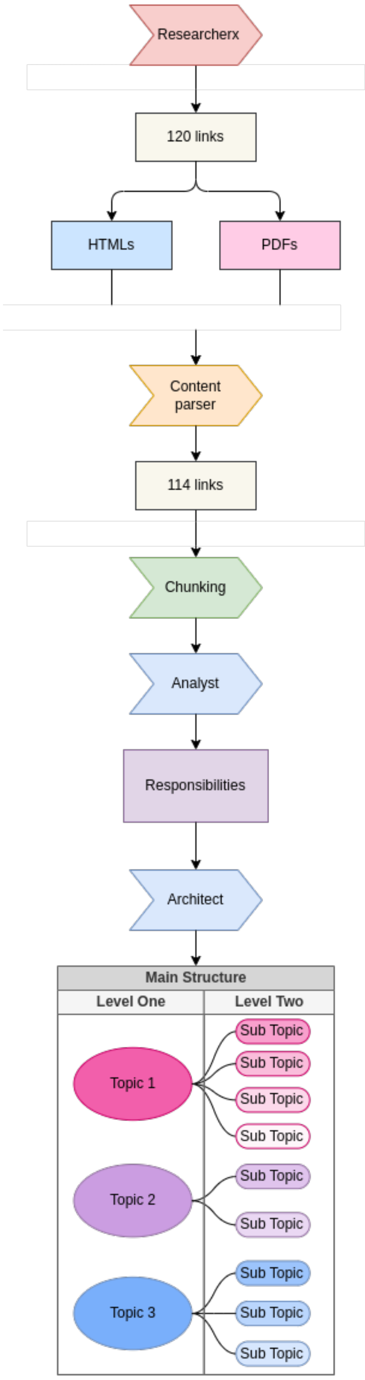

Figure 5.3: Overview of job descriptions pipeline

Given the following job description, help us extract in distinct bullet points the different responsibilities required for a job at an Executive Briefing Center. If you are unsure of the responsibilities, answer "Not enough information". If the content does not correspond to a job description, answer "Not a job description".

This is an example of the ideal output, obtained from a different source:

1. *Develop overall executive briefing program strategy.*
2. *Lead strategic planning and goal setting.*
3. *Build and nurture relationships with key business stakeholders inside Customer Success, Worldwide Sales, Worldwide Marketing, Industry Strategy and Platform Technology.*
4. *Align the program with business strategy and operational changes.*
5. *Up-level the briefing experience for internal and external participants.*
6. *Integrate the Autodesk brand through various touch points including narrative and experience.*
7. *Optimize operations from request to management to reporting.*
8. *Advocate for the value of the program to increase participation and/or investment.*
9. *Develop dashboard and regular reporting to key stakeholders.*
10. *Grow network of speakers and discussion leaders to support customer needs.*
11. *Lead an experienced, passionate and customer-focused team.*
12. *Consolidate briefing feedback and conclusions with post-briefing analysis of insights, follow up plans and opportunities for improvement.*
13. *Partner with Autodesk Workplace, Technology Centers and Gallery team to deliver an exceptional experience.*

Here is the content I want you to analyze as described above:

{content}

Figure 5.4: Prompt used for retrieving location of innovation center

In this case, the prompt to be used follows the few-shot strategy for prompting, in which we provide the model with clear instructions on both the task to be performed as well as the desired output format, together with some examples of the information we are hoping to find in the text (see **Section 3.4**). This was considered to be appropriate, as we needed to both identify services and discriminate them from value propositions. Since the difference between both concepts is subtle, we decided to give the model more context on the problem by providing a clear example of the information it was expected to find in the text. Additionally, this approach helps us in avoiding the inclusion of non-related information in the final response from the model.

Some example of results from this information extraction process can be inspected in **Listing 5.2**.

Code Listing 5.2: Information on responsibilities extracted from job descriptions (JSON Array).


```
[
    {
        "website": "Electrosonic",
        "url": "https://www.electrosonic.com/blog/2018/07/17/seven-factors
            -to-consider-when-designing-an-executive-briefing-center#:~:
            text=An%20Executive%20Briefing%20Center%20is,customers%20meet
            %20their%20business%20goals.",
        "responsibilities": ["Design and equip an Executive Briefing
            Center to demonstrate products, services, and solutions in the
            best possible conditions.", "Define the audience and purpose of
             the innovation center.", "Consider the customer journey,
            including the visitors' route through the building to the
            innovation center, who will meet the visitor and act as a guide
            , whether visitors will need to sign in or be assigned a pass,
            if refreshments will be available, the duration of an average
            visit, if visits will be presentation-based, demonstration-
            based, or a combination of the two, if presentations and/or
            demos will be one-to-one or one-to-many, if visitors will
            receive a copy of the presentation and/or other documentation,
            and if visitors will sign out or hand in a pass.", "Script the
            visitor experience for each type of customer expected.", "Use
            audio/visual technology carefully to add value and complement
            solutions and products on display.", "Build an experienced team
             of specialists, including architects, interior/exhibition
            designers, project managers, mechanical and electrical
            consultants, quantity surveyors and cost control, lighting
```

```
            designers, audio/visual specialists, builders, fit-out
            contractors, and electricians.", "Maintain the innovation
            center to achieve optimum results, including a service
            agreement that covers the audio/visual technology and, if the
            Center is used daily, a remote monitoring solution so problems
            can be fixed 24/7."
        ]
    },
    {
        "website": "Geebo",
        "url": "https://arlington-va.geebo.com/jobs-online/view/id
            /1191484005-briefing-program-manager-ebc-/",
        "responsibilities": ["Manage end-to-end briefing planning process
            for executive briefings hosted in the Arlington, VA Executive
            Briefing Center.", "Develop briefing agendas and facilitate
            experiences that connect AWS executive leaders, technical
            subject matter expert, and customers for strategic, 1:1 outcome
            -based discussions.", "Create mechanisms and drive efficiencies
             to deliver briefings at scale.", "Build key relationships to
            drive the successful delivery of a streamlined experience to
            internal stakeholders and customers.", "Partner with key
            stakeholders including innovation center programming, sales,
            marketing, and technical communities to develop enablement
            materials for AWS sales teams to promote program value and
            adoption within assigned industries.", "Improve, and in some
            cases, develop standard operating procedures." "Utilize
            exceptional communication skills to set appropriate
            expectations with internal and external stakeholders throughout
             the briefing planning process.", "Identity opportunity to
            continually raise the bar for stakeholders by driving YoY
            process improvements.", "Travel: <20% time."
        ]
    },
    . . .
]
```

I will give you a list of similar topics. I need you to summarize the topics into a list of bullet points and come up with a name that summarizes what the topics are talking about. Answer only with the name and bullet points. Make sure the bullet points are mutually exclusive.

Here are the topics:

{ topics }

Figure 5.5: Prompt used for structuring responsibilities of innovation center

Once this information was retrieved, we are left with a long list of responsibilities extracted from a wide variety of information sources. The next step is to organize this information into a cohesive structure. This step is important in order to provide the client with a summary of the key functionalities that an innovation center is expected to be able to provide, as well as the responsibilities the employees are expected to undertake when delivering said services.

In order to do so, we made use of the Architect generator (see **Section 3.4.2**, which is specialized in providing summarization and organization capabilities in order to structure content. In this case, the extracted responsibilities correspond to the sentences that are to be transformed into embeddings and clustered using the community detection algorithm. The hyperparameter settings were a minimal community size of 10 and a similarity threshold of 0.8, which resulted in 6 clusters.

After the cluster centers were retrieved, they were passed as part of a prompt for the gpt-3.5-turbo model, which can be inspected in **Figure 5.5**.

In contrast with the prompt used for retrieving information on job descriptions (**Figure 5.4**), the above prompt does not provide any information about examples of a desired structure to be obtained, thus leaving a considerable room for the model to inspect, analyze and organize the text to its own accord.

A summary of the resulting structure can be inspected in **Listing 5.3**.

Code Listing 5.3: Summary of structure of responsibilities (JSON Array).

```
[
  {
      "cluster_name": "Briefing Program Development",
      "responsibilities": ["Determine appropriate methods and procedures
           for addressing briefing needs", "Customize briefings to be
          high-level/strategic or technical/hands-on",
      "Develop and implement briefing program presentations, materials,
          content, messaging, marketing, communication, and events"
```

```
            . . .
            ]
        },
         {
            "cluster_name": "Reporting and Integration Alignment",
            "responsibilities": ["Develop dashboard and regular reporting to
                key stakeholders", "Coordinate with partners to ensure
                technology integration aligns with company vision"
            ]
        },
         {
            "cluster_name": "Program Value Advocacy",
            "responsibilities": ["Increase participation in the program", "
                Increase investment for the program", "Advocate for the value
                of the program",
            . . .
            ]
        },
         {
            "cluster_name": "Post-Briefing Analysis",
            "responsibilities": ["Gather feedback and conclusions from the
                briefings", "Analyze insights from briefings", "Identify
                opportunities for improvement".
            . . .
            ]
        },
         {
            "cluster_name": "Speaker Network Growth",
            "responsibilities": ["Identify customer needs.", "Recruit new
                speakers and discussion leaders.", "Foster collaboration and
                knowledge sharing within the network."
            ]
        },
         {
            "cluster_name": "Executive Briefing Center Management",
            "responsibilities": ["Design and equip the center to showcase
                products, services, and solutions", "Maintain the center for
                executive briefings", "Define the purpose of the center",
            . . .
            ]
        },
         {
            "cluster_name": "Stakeholder Relationship Management",
```

```
        "responsibilities": ["Develop and maintain relationships with key stakeholders", "uild and maintain relationships with key stakeholders"
        ]
    },
    . . .
]
```

5.3 Services

Another important insight that the client needed to obtain is creating a list of potential services that innovation centers need to be able to provide to clients. An overview of the followed pipeline can be inspected in **Figure 5.6**.

Similar to previous use cases, the first step in this case was to extract relevant information related to innovation center services from a series of validated links. In order to do this, we used a list of 71 links that were manually validated by a team of analysts and processed them as previously described. Fortunately, we found that only four links were unavailable for content retrieval, leaving us with a total of 67 links with valuable information to work with.

After parsing the content from each website, the content was chunked into pieces with a size that was manageable for the LLM model (refer to **Section 3.3.3** for more information on the chunking process). The chunks are then passed to the Analyst to perform retrieval of relevant information using the prompt described in **Figure 5.7**.

The prompting approach used in this case is similar to the one used for extracting information on **Section 5.2**, where we provide a comprehensive description of both the desired output format as well as a practical example of the information to be retrieved.

Some examples of results obtained can be found in **Listing 5.4**.

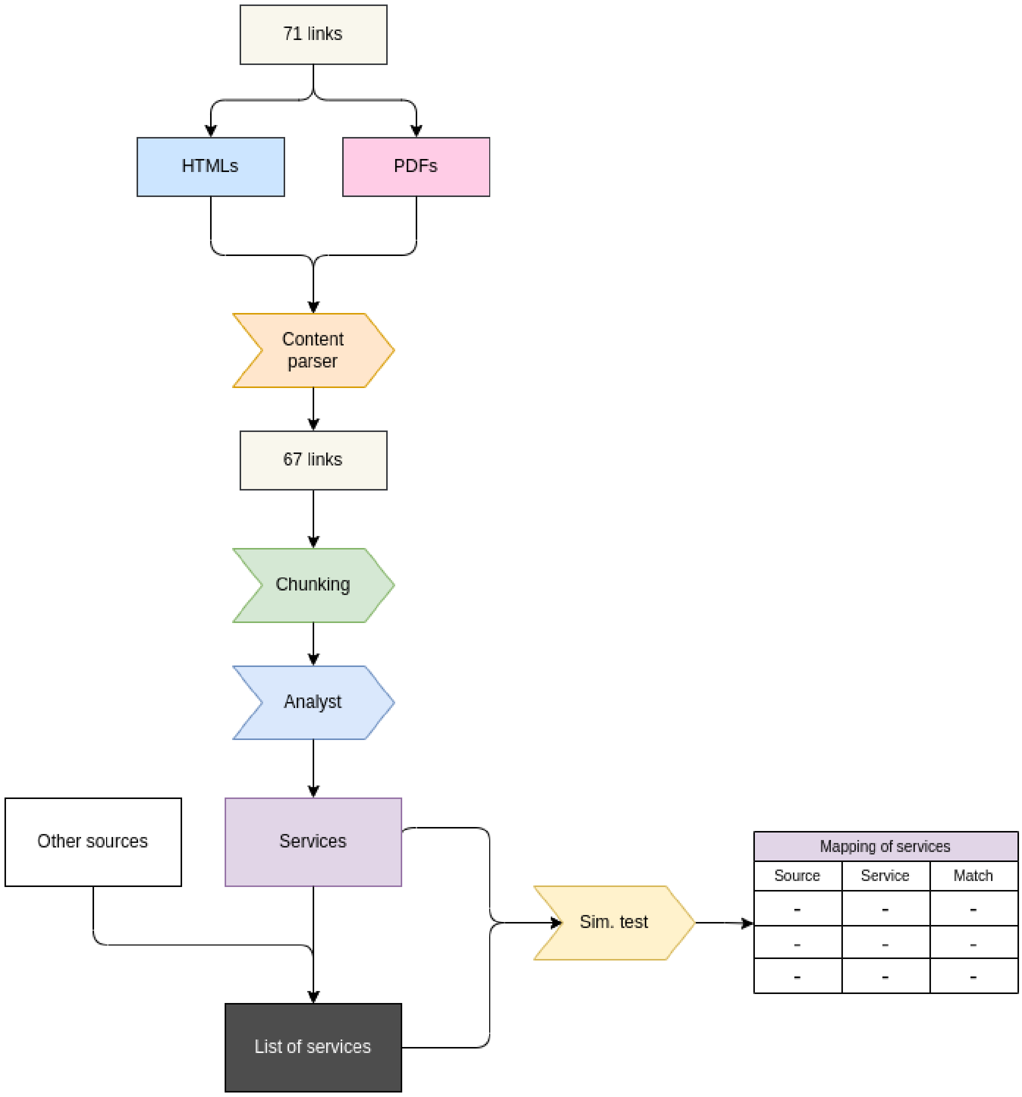

Figure 5.6: Overview of services pipeline

Given the following content, help us summarize in distinct bullet points the different services offered in this Executive Briefing Center. This is an example of the ideal output, obtained from a different source:

1. *Brand Vision & New Business Experience with innovative business solutions and smart electronics to elevate businesses to a new level.*
2. *Smart Hospitality Solution featuring SMART Signages and SMART Hospitality Displays to provide guests with a luxurious in-room experience.*
3. *Retails, Food & Beverage Experience with interactive displays to create a personalized shopping experience, and mobile devices to improve flexibility and boost sales.*
4. *Finance & Corporate new vision with video walls and other financial information to attract and retain customers, and enhance employee collaboration and communication.*
5. *School & Office Projector-less solution with interactive displays like Samsung Flip, which streamlines productivity and enables efficient collaboration, and Touch Board solutions to enable business professionals to stay connected at all times.*
6. *Smart Air Con Experience with the 360 Cassette, which creates a comfortable office environment with its innovative circular design that delivers air evenly throughout every corner in any space.*
7. *Technology & Gallery Integrated Experience with "The Frame," an innovative lifestyle TV that transforms into an in-home artwork, and Outdoor waterproof smart signage that enhances living spaces.*

Here is the content I want you to analyze as described above:

{content}

Figure 5.7: Prompt used for retrieving services of innovation centers

Code Listing 5.4: Information on innovation center services (JSON Array).

```json
[
    {
        "name": "Microfocus",
        "url": "https://www.microfocus.com/media/brochure/virtual-
            executive-briefing-center-brochure.pdf",
        "extracted_services": ["The Virtual Executive Briefing Center
            offers personalized briefings and explores technologies best
            suited to unique needs.", "Session types include IT strategy
            briefings, industry solutions briefings, facilitated business-
            strategy workshops, senior executive and peer-to-peer
            roundtables and discussions.", "The sessions focus on
            delivering value-driven outcomes by establishing a shared
            understanding of challenges and how Micro Focus can help.", "
            The Virtual innovation center briefing experience is custom
            designed to align with goals and project requirements.", "The
            briefing team knows how to deliver a high-quality, high-value
            experience.", "Micro Focus helps run and transform businesses
            with customer-centric innovation and software that provides
            critical tools to build, operate, secure, and analyze the
            enterprise.", "Micro Focus offers solutions for hybrid IT
            management, security, risk, governance, and predictive
            analytics."]
    },
    {
        "name": "Cisco",
        "url": "https://www.cisco.com/c/dam/en_us/about/ac156/
            cisco_executive_briefing_center_feature09186a00800a2fe1.pdf",
        "extracted_services": ["Identifying current and future business
            needs", "Exchanging ideas with Cisco experts", "Implementing an
             Internet business model to position for success", "Generating
            new revenue, reducing costs, and empowering employees with
            Internet applications", "Learning about new networking
            technologies", "Seeing hands-on demonstrations of leading
            solutions in cable, dial, DSL, and voice-over-IP technology", "
            Maximizing return on investment and providing strategic
            business applications such as e-commerce", "Strengthening
            relationships among employees, customers, and suppliers", "
            Providing the network foundation for cost-effective, strategic
            Internet business solutions."]
    },
    . . .
]
```

Once the information was extracted, it was passed on to a team of analysts, which combined it with other sources that they had already investigated beforehand and used it to construct a summarized list of services.

The next goal was to map back the summarized services to the list of services originally extracted. The main objective of this exercise is to be able to identify which innovation centers are incorporating which services, as well as highlighting new services that the analysts may have overlooked when building a summary of services for the client.

The task is not trivial, as it requires processing of a large amount of extracted information. In addition, the processed list of services that we were given for mapping back the content did not correspond to the original services extracted, as the analysts themselves further processed the information. Thus, we needed to implement a technique that would allow us to check whether or not the service from the source has a match within the list of services provided, pairing the source service with the closest match from the list provided.

In order to do so, we used a slight modification of the Source-Based Content Test (see **Section 3.5.4**). As a brief reminder, this test performs a similarity test between a sentences and a source and retrieves the closest match, so as to check whether the target sentence was indeed produced using the information provided by the source. In our case, we performed the similarity test to compare a service provided by an innovation center and the list of services provided by the analysts. By setting a high similarity threshold when performing selection of the candidate embeddings (matching only the most similar sentence), we are able to retrieve the closest service from the list provided.

Once the most similar candidate is retrieved, we evaluate its similarity score to decide whether or not it is truly a match to the service originally presented in the source. In this case, we decided to set a fairly low threshold for considering a match (similarity score > 0.2), as we wanted to keep a conservative approach to the problem and allow analysts to review and inspect the results. Some examples of results can be inspected in **Listing 5.5**.

Code Listing 5.5: Service Mapping Results (JSON Array).

```
[
    {
        "source_service": "Personalized briefings: Customized sessions
            tailored to unique needs, including IT strategy briefings,
            industry solutions briefings, facilitated business-strategy
            workshops, senior executive and peer-to-peer roundtables and
            discussions",
        "closest_match": "Pre-briefing scheduling and arrangements",
        "score": 0.37,
        "match: 1
    },
    {
        "source_service": "Value-driven outcomes: Sessions focused on
            delivering value-driven outcomes by establishing a shared
            understanding of challenges and how Micro Focus can help",
        "closest_match": "Benchmarking and performance assessments",
        "score": 0.25,
        "match: 1
    },
    {
        "source_service": "Hands-on product demonstrations: Experience
            today's most innovative technology solutions firsthand",
        "closest_match": "Product and Solutions demonstrations",
        "score": 0.74,
        "match: 1
    },
    {
        "source_service": "Hybrid IT management: Reliable scaling of
            DevOps across all environments, from mainframe to cloud,
            bringing innovative ideas to life at the pace of business
            demands",
        "closest_match": "& Customized workshops (architecture, roadmap)",
        "score": 0.04,
        "match: 0
    },
    . . .
]
```

5.4 Value proposition

Table 5.1: Results from Value Proposition Mapping

Source	Vision	Relationships	Training	Fun	Expertise	Collab.	Innov.	Performance	Value	Custom.
Microfocus	0	0.3	0.2	0.1	0.2	0.07	0.03	0	0.1	0
Cisco	0	0.2	0	0	0.4	0	0.13	0	0.23	0.03
CDN	0.07	0.23	0.13	0.17	0.2	0.1	0	0	0.1	0
Juniper	0.1	0.3	0	0.2	0.2	0.1	0.1	0.1	0	0

Finally, our last use case example was in the context of extracting the value proposition of a series of innovation centers, as well as mapping the content to a list of pre-defined value propositions that a team of analysts had already designed. A visual diagram of the followed pipeline can be inspected in **Figure 5.8**.

In this case, the pipeline to be implemented is slightly different than the ones explained before, since the content extraction and structure mapping steps are done using a single call to the Analyst generator. In this case, the generator will take unstructured information from a source and, through means of custom prompting, directly map back the information to a pre-defined structure.

We were provided with a total of 71 sources validated by analysts, which were the same sources we used for performing content extraction in the case of services (see **Section 5.3**. The prompt used for performing content extraction and mapping can be inspected in **Figure 5.9**.

Note that the prompt used in this case is much more complex than some of the previously used prompts. This is due to the fact that we are including several sources of information. Firstly, we are providing the Analyst generator with information on the categories it is expected to use for mapping. Secondly, we are also providing a source of content that the model is expected to process using the previously indicated categories. Finally, we are also instructing the model to perform scoring of how well the content matches the categories, making the categories sum up to one.

Since in this case we are asking the model to perform a scoring task, we also incorporated the mean scores test into the processing pipeline, so as to be sure that the model is consistent in the assignment of scores (see **Section 3.5.5** for more information on this test). In order to do this, we performed the calls to the model three times prior to applying the mean scores algorithm described in **Algorithm 7**.

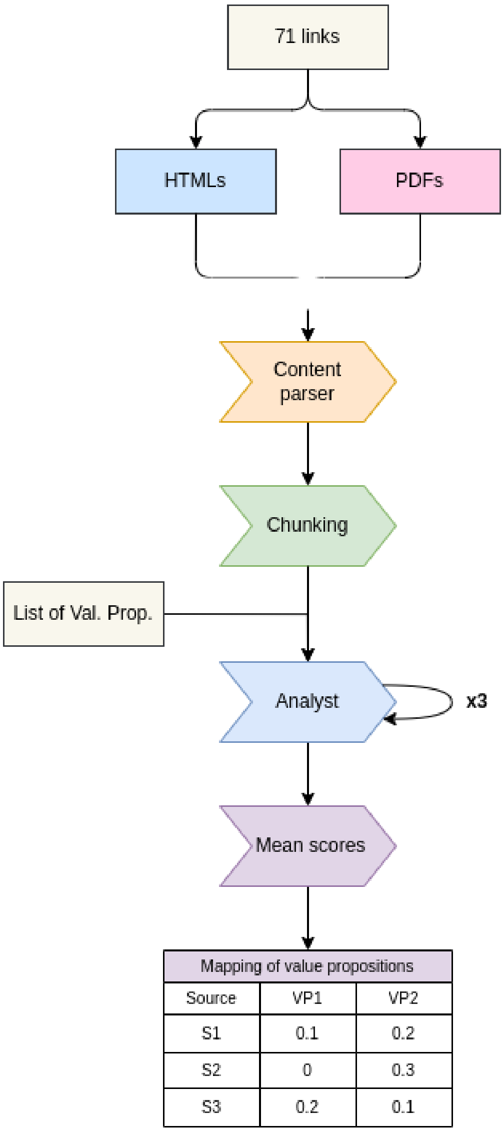

Figure 5.8: Overview of value proposition pipeline

In order to obtain even more visibility on the reasoning process of the model, we also instructed it to give a brief explanation as to why it assigned each particular score. This part of the prompt is design to force the model to conduct a deeper reasoning process when assigning scores, so as to prevent it form hallucinating and giving baseless answers. Some examples of results from this use case can be inspected in **Table 5.1**.

Please help us classify the following content related to an innovation center. Classify it into the following 10 categories and provide a short explanation for your choices. Give each category a relative weight so that in total, the 10 categories sum up to 1.

Categories:

Vision, Inspiration, Direction

Build Relationships and Networking

Development and Training

Personal Fun and Experience

Information and Expertise

Collaboration

Innovation

Improve performance

Economic return/value

Personalization/Customization

Provide the output as a list of values, with the relative numeric values followed by the comments. For example: [0, 0.35, 0.35, 0.1, 0, 0, 0, 0, 0, 0.2, "Comments explaining the choices."]. Answer with just this paragraph, without any introduction. Keep the explanations short and consistent with the given scores.

Content {content }

Figure 5.9: Prompt used for retrieving value propositions of innovation centers

Chapter 6

Future prospects

Since the development of new advancements in the field of Natural Language Processing (NLP), more and more companies are redefining the way in which they work through incorporating new technologies such as ChatGPT to help release pressure off their employees and increase their overall productivity. However, there are still some concerns on the way in which these technologies operate that hinder their full adoption in a business setting. Moreover, current solutions do not provide comprehensive frameworks that integrate several technologies together to leverage their combined power. These factors create a pressing need to develop such tools within a comprehensive framework, allowing users to safely interact with Large Language Models (LLMs) while abstracting them from the intricacies of how the underlying processes work and the connections between them.

We hope that the present work has laid the foundation for the future prospects of NLP in business environments, specially in the field of strategy consulting. However, it is important to acknowledge that this research represents only a stepping stone towards the full realization of the potential of NLP technology in a business setting. Despite the advancements made, there are still many paths for improvement and refinement that need to be explored.

The challenges lie not only in the technical aspects but also in ensuring the seamless integration of these tools into the existing processes and workflows of strategy consultants. Additionally, the ethical considerations surrounding the use of NLP, such as privacy concerns and bias mitigation, demand continued attention and research.

In this line, we have identified three main areas of improvement that we hope to address as our next steps:

6.1 Technical improvements

In this section, we discuss key technical improvements that can further enhance the effectiveness and applicability of NLP tools in the field of strategy consulting. While the present work has showcased the potential of NLP, it is important to recognize that there is still room for improvement in several technical aspects.

Tools that heavily rely on the use of LLM (such as the **Generators** and the **Subfact-checking test**) need further iteration and testing, so as to prime their performance prior to market release. As discussed in **Section 4.1.5**, prompt iteration is crucial for these applications. Even though significant efforts have been made to achieve good results, we are still far from achieving peak performance. An example of this is the level of accuracy obtained by the subfact-checking test, as well as its high rate of False Positive results.

We would also want to apply our technologies to a wider variety of use case scenarios. Using the tools developed for solving real client needs and helping in external projects frequently made us realize the potential weaknesses of our approaches, giving us the opportunity to iterate them and refine them in order to provide faster and better responses.

Another potential challenge within this project was staying up to date with the ever changing field in which it was developed. The field of LLMs is still fairly new and a lot of research is being put into constantly improving both the performance of the models as well as developing new features and integrations that could complement them. Being a small team, it was difficult to keep up with the industry standards, as they can change dramatically from one week to another. Thus, the project that is being presented as of today may be obsolete in a matter of months if we do not manage to maintain it up to date with the new advancements in the field.

In order to achieve this, we aim to refactor the code and integrate it with LangChain capabilities as much as possible. LangChain is rapidly being adopted as the state-of-art framework for developing LLM-based applications. Using it may not only mean more readable and bug-free code, but also open up possibilities for later integration with third party solutions when developing a final product to serve to clients.

In addition, we are also exploring the use of emerging technologies such as AutoGPT[1], an autonomous agent that uses GPT-4 technology to autonomously complete tasks. Even though the AutoGPT technology is still experimental, its open-source nature is allowing it to grow rapidly and the potential benefits that could

[1]https://github.com/Significant-Gravitas/Auto-GPT

unlock for advancing this project are still to be uncovered.

Finally, the scalability and efficiency of the developed tools needs to be addressed. As the volume of data to be processed continues to grow exponentially, it is imperative to develop techniques that can handle big amounts of data effectively. This involves optimizing algorithms and infrastructure to minimize processing time and resource consumption while maintaining high accuracy levels.

6.2 Product development

Another field of improvement when developing this set of tools involved the integration with the client's working pipelines, which was sometimes difficult to manage, as we had to work with them asynchronously whilst providing high quality outputs in a short amount of time. This frequently hindered the usability of our solution, as the tools were not quick enough to adapt to the fast-pace and changing environment that they are to be applied in. Furthermore, changes in the project for which the technologies were to be applied often resulted in major changes in code that delayed delivery to the final client.

In order to tackle this challenge, we first need to optimize the code performance by refactoring, reducing computational times and increasing code reusability, so that the amount of changes that need to be implemented from project to project is minimal.

We also hope to advance in the development of an application for serving our capabilities to end users. This will allow a more organic integration with the requirements of the project, as the users will be able to interact directly with a front-end interface that allows them to autonomously make use of the tools provided, thus allowing us to work on further improvements and releases as well as getting direct feedback from users help us advance our solution.

The development of the end application is still open to discussion, but we are exploring using *Streamlit*, which is a web-framework that allows for the development of Data Science applications in Python, providing a high level of abstraction from underlying front-end rendering components.

Another potential approach would be to develop a plug-in integration using Google's API. This way, the client can directly interact with the solution whilst working from their preferred tools, such as Google Docs and Google Slides, without having to interact with a separate application and migrate the results into a separate document.

The end product of this research is designed to be served using a combination of AWS Lambda and a microservice architecture. AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS), which allows for the execution of code in an scalable and cost-effective way. With AWS Lambda, the application's code is divided into small, self-contained functions that are triggered by specific events or requests.

On the other hand, the microservice architecture is a design choice in which the application is broken down into multiple independent and loosely coupled services, each responsible for a specific task or functionality. These services can be developed and deployed independently, promoting modularity, flexibility, and scalability.

By combining AWS Lambda with a microservice architecture, the end product benefits from the advantages offered by both approaches. AWS Lambda enables automatic scaling and resource allocation, eliminating the need for manual infrastructure management and optimizing costs based on the actual demand. The microservice architecture allows for the decomposition of the application into smaller, manageable services that can be individually developed, deployed, and scaled. This modular nature promotes flexibility, agility, and resilience, as each microservice can be independently updated, replaced, or scaled based on their usage patterns, optimizing resource allocation and reducing operational costs.

6.3 Ethical considerations

Lastly, the ethical implications associated with NLP deployment must be carefully considered. Strategies for addressing privacy concerns, bias mitigation, and ensuring fairness and transparency in decision-making processes should be explored and implemented. Ethical guidelines and best practices specific to NLP applications in strategy consulting need to be developed to ensure responsible and accountable use of these technologies.

Bibliography

[1] P. Norvig. *ELIZA: Paradigms of Artificial Intelligence Programming*. San Francisco: Morgan Kaufmann Publishers, 1992. ISBN: 1-55860-191-0.

[2] Terry Winograd. "Procedures as a Representation for Data in a Computer Program for Understanding Natural Language". In: (1971). URL: hdl:1721.1/7095.

[3] Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita. *A Survey of the Usages of Deep Learning in Natural Language Processing*. 2019. arXiv: 1807.10854 [cs.CL].

[4] Karen Sparck Jones. "Natural language processing: a historical review". In: *Current Issues in Computational Linguistics: in Honour of Don Walker*. Springer, 1994. DOI: 10.1007/978-94-011-1093-1_1.

[5] Aditya Malte and Pratik Ratadiya. "Evolution of Transfer Learning in Natural Language Processing". In: (2019). arXiv: 1910.07370 [cs.CL].

[6] Anton Chernyavskiy, Dmitry Ilvovsky, and Preslav Nakov. "Transformers: "The End of History" for Natural Language Processing?" In: (2021). arXiv: 2105.00813 [cs.CL].

[7] Diksha Khurana et al. "Natural language processing: state of the art, current trends and challenges". In: (2023). DOI: 10.1007/s11042-022-13428-4. URL: https://doi.org/10.1007/s11042-022-13428-4.

[8] Jeffrey L. Elman. "Finding structure in time". In: *Cognitive Science* 14.2 (1990). ISSN: 0364-0213. DOI: https://doi.org/10.1016/0364-0213(90)90002-E. URL: https://www.sciencedirect.com/science/article/pii/036402139090002E.

[9] L. Fausett and L.V. Fausett. *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*. Prentice-Hall international editions. Prentice-Hall, 1994. ISBN: 9780133341867.

[10] Stefan Kombrink et al. "Recurrent Neural Network Based Language Modeling in Meeting Recognition." In: Aug. 2011, pp. 2877–2880. DOI: 10.21437/Interspeech.2011-720.

[11] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[12] Junyoung Chung et al. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. 2014. arXiv: 1412.3555 [cs.NE].

[13] Ashish Vaswani et al. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].

[14] Colin Raffel et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2020. arXiv: 1910.10683 [cs.LG].

[15] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[16] Sébastien Bubeck et al. *Sparks of Artificial General Intelligence: Early experiments with GPT-4*. 2023. arXiv: 2303.12712 [cs.CL].

[17] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: (2018). URL: https://github.com/codelucas/newspaper.

[18] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[19] OpenAI. *GPT-4 Technical Report*. 2023. arXiv: 2303.08774 [cs.CL].

[20] Zhilin Yang et al. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. 2020. arXiv: 1906.08237 [cs.CL].

[21] Jordan Hoffmann et al. *Training Compute-Optimal Large Language Models*. 2022. arXiv: 2203.15556 [cs.CL].

[22] Karan Singhal et al. "Large Language Models Encode Clinical Knowledge". In: (2022). arXiv: 2212.13138 [cs.CL].

[23] Enkelejda Kasneci et al. "ChatGPT for good? On opportunities and challenges of large language models for education". In: *Learning and Individual Differences* 103 (2023), p. 102274. ISSN: 1041-6080. DOI: https://doi.org/10.1016/j.lindif.2023.102274. URL: https://www.sciencedirect.com/science/article/pii/S1041608023000195.

[24] A. Shaji George, A.s George, and A Martin. "A Review of ChatGPT AI's Impact on Several Business Sectors". In: 01 (Feb. 2023), pp. 9–23. DOI: 10.5281/zenodo.7644359.

[25] Samuel A. Prieto, Eyob T. Mengiste, and Borja García de Soto. "Investigating the Use of ChatGPT for the Scheduling of Construction Projects". In: *Buildings* 13.4 (2023). ISSN: 2075-5309. URL: `https://www.mdpi.com/2075-5309/13/4/857`.

[26] Nils Reimers and Iryna Gurevych. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. 2019. arXiv: `1908.10084 [cs.CL]`.

[27] *STSbenchmark - Semantic Textual Similarity Benchmark*. `https://ixa2.si.ehu.eus/stswiki/index.php/STSbenchmark`. Accessed on: May 8, 2023. 2017.

[28] Harrison Chase. *LangChain*. Oct. 2022. URL: `https://github.com/hwchase17/langchain`.

[29] Pengfei Liu et al. *Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing*. 2021. arXiv: `2107.13586 [cs.CL]`.

[30] Daniel Cer et al. *Universal Sentence Encoder*. 2018. arXiv: `1803.11175 [cs.CL]`.

[31] Sebastian Hofstätter et al. "Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling". In: *Proc. of SIGIR*. 2021.

[32] Nils Reimers and Iryna Gurevych. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. 2019. arXiv: `1908.10084 [cs.CL]`.

[33] Alex Wang et al. *GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding*. 2019. arXiv: `1804.07461 [cs.CL]`.

Appendix A

Experiment Appendix

A.1 Content Filtering Experiments

Code Listing A.1: Information on filtering experiments (JSON Array).


```
[
    {
        "experiment": "Construction",
        "keywords": ["construction", "A-4 highway", "Madrid delayed", "
            land acquisition", "issues", "2019"],
        "links": ["https://www.madrid.es/UnidadWeb/NxC/PlanRecuperacion/
            rtsingles.pdf", "http://www.sciencedirect.com/science/article/
            pii/S0967070X21002729", "http://www.sciencedirect.com/science/
            article/pii/S0967070X21002729"]
    },
    {
        "experiment": 'saturn",
        "keywords": ["Voyager", "1977", "outer solar system", "Jupiter", '
            saturn", "Uranus", "Neptune", ""Earth"],
        "links": ["https://voyager.jpl.nasa.gov/", "https://voyager.jpl.
            nasa.gov/mission", "https://spaceplace.nasa.gov/voyager-to-
            planets/en/"]
    },
    {
        "experiment": "NER",
        "keywords": ["mid-1990s", "University of Massachusetts Amherst", "
            Ralph Grishman", "named entities", "text"],
        "links": ["https://en.wikipedia.org/wiki/Named-entity_recognition
            ", "https://aclanthology.org/W98-1118.pdf", "https://julian-
```

```
            urbano.info/files/publications/003-named-entity-recognition-
            fallacies-challenges-opportunities.pdf"]
    }
]
```

Code Listing A.2: Information on filtering experiments (JSON Array).

```
[
    {
        "experiment": "PPP",
        "keywords": ["Conduct", "thorough", "due diligence", "entering", "
            PPP agreement"],
        "links": ["https://ppp.worldbank.org/public-private-partnership/
            ppp-overview/practical-tools/checklists-and-risk-matrices/due-
            diligence-checklist", "https://ppp-certification.com/ppp-
            certification-guide/10-overview-ppp-process-cycle-how-prepare-
            structure-and-manage-ppp-contract"
        ]
    },
    {
        "experiment": "ElasticSearch",
        "keywords": ["Elasticsearch", 'scale horizontally", "handle", "
            increasing amount of data", "ease", "great choice", "large-
            scale applications", "real-time search", "analytics"],
        "links": ["https://veloxsoftech.com/blog/benefits-of-using-
            elasticsearch", "https://medium.com/@AIMDekTech/what-is-
            elasticsearch-why-elasticsearch-advantages-of-elasticsearch-47
            b81b549f4d", "https://www.inviggo.com/blog/elasticsearch-101-
            key-concepts-benefits-use-cases"]
    },
    {
        "experiment": "NLP",
        "keywords": ["Natural Language Processing", "NLP", 'subfield", "
            artificial intelligence", "interaction", "computers", "humans",
             "natural language", "algorithms", "computational techniques",
            "analyze", "understand", "generate", "human language"],
        "links": ["https://www.techtarget.com/searchenterpriseai/
            definition/natural-language-processing-NLP", "https://www.ibm.
            com/topics/natural-language-processing", "https://en.wikipedia.
            org/wiki/Natural_language_processing"]
    }
]
```

A.2 Correctness Experiments

A.2.1 Chunk size Experiment

Code Listing A.3: Example result of chunking experiment (JSON Array).

```
[
    {
        "source": "Natural language processing (NLP) is the ability of a
            computer program to understand human language as it is spoken
            and written, referred to as natural language. NLP enables
            computers to understand natural language as humans do. A lot of
             the information created online and stored in databases is
            natural human language, and until recently, businesses could
            not effectively analyze this data. " If a user relies on
            natural language processing for search, the program will
            recognize that cloud computing is an entity, that cloud is an
            abbreviated form of cloud computing and that SLA is an industry
             acronym for service-level agreement. These are the types of
            vague elements that frequently appear in human language and
            that machine learning algorithms have historically been bad at
            interpreting. NLP uses syntax to assess meaning from a language
             based on grammatical rules. Semantics involves the use of and
            meaning behind words. Natural language processing applies
            algorithms to understand the meaning and structure of sentences
            . Deep learning models require massive amounts of labeled data
            for the natural language processing algorithm to train on and
            identify relevant correlations, and assembling this kind of big
             data set is one of the main hurdles to natural language
            processing. But deep learning is a more flexible, intuitive
            approach in which algorithms learn to identify speakers" intent
             from many examples -- almost like how a child would learn
            human language. The machine interprets the important elements
            of the human language sentence, which correspond to specific
            features in a data set, and returns an answer.",
        "n_tokens": 659,
        "model_response": [The key challenges in NLP: True ("Natural
            language processing applies algorithms to understand the
            meaning and structure of sentences. Deep learning models
            require massive amounts of labeled data for the natural
            language processing algorithm to train on and identify relevant
             correlations, and assembling this kind of big data set is one
            of the main hurdles to natural language processing."),
```

```
        "Dealing with the ambiguity and complexity of human language: True
            ("These are the types of vague elements that frequently appear
            in human language and that machine learning algorithms have
           historically been bad at interpreting.")",
        "Words can have multiple meanings: True ('semantics involves the
           use of and meaning behind words.")",
        "Meanings depend on the context in which they are used: True ('
           semantics involves the use of and meaning behind words.")",
        'sentences can have different interpretations: True ("NLP uses
           syntax to assess meaning from a language based on grammatical
           rules.")",
        "Interpretations depend on the tone and emphasis of the speaker:
           False. This subfact is not mentioned in the paragraph."
        ]
    }
]
```

A.2.2 Prompting Experiment

Code Listing A.4: Correctness Experiment Prompt 1

```
response = openai.ChatCompletion.create(
  model="gpt-3.5-turbo",
  messages=[
        {"role": "user", "content": "I will give you a fact. Your job is to give a
             score from 0-10 evaluating the correctness of the fact. Cite 5
            sources used to check the fact."},
        {"role": "user", "content": "The proposed construction of the A-4 highway
            in Madrid, Spain, was delayed due to land acquisition issues in 2019"}
    ]
)
```

Code Listing A.5: Experiment 1 Example Result

```
"I would rate the fact a 9 out of 10 for correctness, as there were some delays in
     the construction of the A-4 highway, but I was not able to find information
    on the specific reason for the delay in 2019.

Sources:

1. "A-4 highway in Spain" https://en.wikipedia.org/wiki/A-4_highway_(Spain)

2. "New Madrid to Cadiz A-4 highway to open in 2020 after nearly a decade of
    construction" https://www.theolivepress.es/spain-news/2020/11/11/new-madrid-to
    -cadiz-a-4-highway-to-open-in-2020-after-nearly-a-decade-of-construction/

3. "Madrids M30 and the new A4 highway" https://www.fcc.es/en/-/madrid-s-m-30-and-
    the-new-a-4-highway

```

```
4. "A4 Highway, Spain" https://www.roadtraffic-technology.com/projects/a4-highway-spain/

5. "Almodovar del Rio requests a modification of the proposal of the A4 highway due to the impact on the environment" https://www.lavanguardia.com/local/sevilla/20191011/47883889989/almodovar-del-rio-pide-una-modificacion-de-la-propuesta-de-la-a-4-por-el-impacto-en-el-medio-ambiente.html"
```

Code Listing A.6: Correctness Experiment Prompt 2

```
response = openai.ChatCompletion.create(
  model="gpt-3.5-turbo",
  messages=[
        {"role": "user", "content": "I will give you a fact. Your job is to give a score from 0-10 evaluating the correctness of the fact. Cite the sources you used to check the fact."},
        {"role": "user", "content": "Q: Public-Private Partnerships (PPPs) are long term agreements between the government and a private partner whereby the private partner delivers and funds public services using a capital asset, sharing the associated risks."},
        {"role": "assistant", "content": "10 (https://www.oecd.org/gov/budgeting/oecd-principles-for-public-governance-of-public-private-partnerships.htm#:~:text=Public%2DPrivate%20Partnerships%20(PPPs),asset%2C%20sharing%20the%20associated%20risks.)"},
        {"role": "user", "content": "The proposed construction of the A-4 highway in Madrid, Spain, was delayed due to land acquisition issues in 2019"}
    ]
)
```

Code Listing A.7: Experiment 2 Example Result

```
"8. Source: https://elpais.com/ccaa/2019/01/09/madrid/1547072582_239456.html The article reports that the construction of the A-4 highway in Madrid was delayed due to land acquisition issues, but it does not provide extensive information on the extent or duration of the delay."
```

Code Listing A.8: Correctness Experiment Prompt 3

```
response = openai.ChatCompletion.create(
  model="gpt-3.5-turbo",
  messages=[
        {"role": "user", "content": "I will give you a fact and 5 sources. Your job is to give a score from 0-10 evaluating the correctness of the fact using only the information in the sources. Indicate the paragraphs where you extracted the information from."},
        {"role": "user", "content": "Fact: The proposed construction of the A-4 highway in Madrid, Spain, was delayed due to land acquisition issues in 2019"},
        {"role": "user", "content": "source 1: https://www.madrid.es/UnidadWeb/NxC/PlanRecuperacion/rtsingles.pdf"},
        {"role": "user", "content": "source 2: http://www.sciencedirect.com/science/article/pii/S0967070X21002729"},
        {"role": "user", "content": "source 3: http://www.sciencedirect.com/science/article/pii/S0967070X21002729"},
    ]
)
```

Code Listing A.9: Experiment 3 Example Result

```
"Source 4: https://elpais.com/espana/madrid/2019/01/31/actualidad/1548981332
    _316085.html
Source 5: https://www.europapress.es/madrid/noticia-obras-ampliacion-m-30-aprueban
    -proyecto-constructivo-trazado-aire-m-20210118160545.html
Score: 7
Explanation: Source 1 does not provide any information regarding the A-4 highway
    construction and land acquisition issues in Madrid. Source 2 and 3 do not
    provide any information regarding the A-4 highway construction and land
    acquisition issues in Madrid, as they discuss a completely different topic.
    Source 4 mentions that the construction of the A-5 highway in Madrid was
    delayed due to land acquisition issues. However, it does not mention any
    delays in the construction of the A-4 highway.Source 5 mentions the approval
    of the construction project for the expansion of the M-30, which is a
    different highway in Madrid. However, it does not mention any delays in the
    construction of the A-4 highway."
```

Code Listing A.10: Correctness Experiment Prompt 4

```
response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
    messages=[
            {"role": "user", "content": "I will give ask you a source and a fact.
                Your job is to give a score from 1-10 assessing if the fact is
                true or not using only the source provided. Cite the part of the
                text that supports your conclusion."},
            {"role": "user", "content": "source: "+source+ ". Fact: "+fact}
            ]
)
```

Code Listing A.11: Experiment 4 Example Result

```
'score: 0/10

The source does not support the fact given. The source only mentions the launch
    dates and locations of Voyager 1 and Voyager 2. There is no mention of which
    planets they have explored.

Score: 10

The fact is supported by the source as it states that Voyager 1 was launched into
    outer space on September 5, 1977, along with Voyager 2.

Score: 1/10

The fact is not true. The source clearly states that Voyager 1 was launched on
    September 5, 1977, from Cape Canaveral aboard a Titan-Centaur rocket.
"
```

Code Listing A.12: Correctness Experiment Prompt 5

```
response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
    messages=[
            {"role": "user", "content": "I will give ask you a source and a fact.
                Your job is to guess whether the fact is true or not based on the
```

```
                source provided. Cite the part of the text that supports your
                conclusion."},
            {"role": "user", "content": "source: "+text+ ". Fact: Saturn is larger
                 than Earth"},
            {"role": "assistant", "content": "True. The article states that "It is
                 a gas giant with an average radius of about nine and a half times
                 that of Earth"},
            {"role": "user", "content": "Fact: Saturn is the biggest planet in the
                 solar system"},
            {"role": "assistant", "content": "False. The article states that '
                saturn is the sixth planet from the Sun and the second-largest in
                the Solar System""},
            {"role": "user", "content": "Fact: Saturn has life in it."},
            {"role": "assistant", "content": "False. The article does not have
                that information"},
            {"role": "user", "content": "source: "+source+ ". Fact: "+fact}
            ]
)
```

Code Listing A.13: Experiment 5 Example Result

```
"True. The article states that "These spacecraft were launched in 1977 with the
    mission to explore the outer solar system. They have since explored Jupiter,
    Saturn, Uranus, and Neptune, and are now in interstellar space, still
    transmitting valuable data back to Earth."
```

Code Listing A.14: Correctness Experiment Prompt 6

```
response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
    temperature=0,
    messages=[
            {"role": "user", "content": "I will give you a paragraph and a list of
                 subfacts. Your job is to check if each subfact is mentioned or
                not in the paragraph"},
            {"role": "user", "content": "Use only the paragraph. Cite the sentence
                 in the paragraph where the information is located."},
            {"role": "user", "content": "Answer each subfact separately using this
                 format: {subfact}: True/False ('sentence in paragraph")"},
            {"role": "user", "content": "Paragraph: "+source},
            {"role": "user", "content": "List of subfacts to check: " +facts}
            ]
)
```

Code Listing A.15: Experiment 6 Example Result (JSON Array).

```
[
    {
        "source": "As an example, George Lakoff offers a methodology to
            build natural language processing (NLP) algorithms through the
            perspective of cognitive science, along with the findings of
            cognitive linguistics, with two defining aspects:. , under the
            notion of "cognitive AI". The premise of symbolic NLP is well-
```

```
            summarized by John Searle’s Chinese room experiment: Given a
            collection of rules (e. , a Chinese phrasebook, with questions
            and matching answers), the computer emulates natural language
            understanding (or other NLP tasks) by applying those rules to
            the data it confronts. This is increasingly important in
            medicine and healthcare, where NLP helps analyze notes and text
             in electronic health records that would otherwise be
            inaccessible for study when seeking to improve care. Despite
            the popularity of machine learning in NLP research, symbolic
            methods are still commonly used:. Since the neural turn,
            statistical methods in NLP research have been largely replaced
            by neural networks. In some areas, this shift has entailed
            substantial changes in how NLP systems are designed, such that
            deep neural network-based approaches may be viewed as a new
            paradigm distinct from statistical natural language processing.
             Based on long-standing trends in the field, it is possible to
            extrapolate future directions of NLP. Most higher-level NLP
            applications involve aspects that emulate intelligent behaviour
             and apparent comprehension of natural language. More broadly
            speaking, the technical operationalization of increasingly
            advanced aspects of cognitive behaviour represents one of the
            developmental trajectories of NLP.",
        "model_response": ["Natural Language Processing (NLP) is a
            subfield of artificial intelligence.: True ("As an example,
            George Lakoff offers a methodology to build natural language
            processing (NLP) algorithms through the perspective of
            cognitive science")", "NLP deals with the interaction between
            computers and humans using natural language.: True ("Given a
            collection of rules (e., a Chinese phrasebook, with questions
            and matching answers), the computer emulates natural language
            understanding (or other NLP tasks) by applying those rules to
            the data it confronts.")", "NLP involves the use of algorithms
            and computational techniques.: True ("George Lakoff offers a
            methodology to build natural language processing (NLP)
            algorithms through the perspective of cognitive science")", "
            NLP is used to analyze human language.: True ("NLP helps
            analyze notes and text in electronic health records that would
            otherwise be inaccessible for study when seeking to improve
            care.")", "NLP is used to understand human language.: True ("
            Given a collection of rules (e. , a Chinese phrasebook, with
            questions and matching answers), the computer emulates natural
            language understanding (or other NLP tasks) by applying those
            rules to the data it confronts.")", "NLP is used to generate
```

```
            human language.: False"
        ]
    },
    . . .
]
```

A.3 Relevance Experiments

Code Listing A.16: Results for Relevance Test 1 (JSON Array).

```
[
    {
        "question": "Give me risks examples of cardiovascular associated
            disease",
        "answers": ["One example of a cardiovascular associated disease is
             coronary artery disease (CAD)", "Coronary artery disease
            occurs when plaque builds up in the walls of the coronary
            arteries, which supply blood to the heart muscle.", "Coronary
            artery disease can result in reduced blood flow to the heart,
            which can cause chest pain (angina), shortness of breath, heart
             attack, or even death.", "Risk factors for Coronary artery
            disease include high blood pressure, high cholesterol, smoking,
             obesity, diabetes, and a family history of the disease."],
        "scores": [0.9249324 , 0.0302692 , 0.22553106, 0.93479955],
    }
]
```

Code Listing A.17: Results for Relevance Test 2 (JSON Array).

```
[
    {
        "question": "What are the main benefits of using NoSQL databases
            ?",
        "answers": [
            { "a": """
                Scalability: NoSQL databases are designed to scale
                    horizontally, which means they can handle large amounts
                     of data and high levels of traffic with ease. They can
                     also be distributed across multiple servers, making it
                     easy to add more capacity as needed.
```

```
                Flexibility: Unlike relational databases, NoSQL databases are schema-less, which means they can store data in a more flexible way. This allows for greater agility in development, as changes to the data model can be made without having to modify the entire database schema.
                Performance: NoSQL databases are optimized for performance, particularly when it comes to read-heavy workloads. They can handle large volumes of data at high speeds, making them a good choice for applications that require real-time data processing.
                Availability: NoSQL databases are designed to be highly available, with built-in replication and failover capabilities. This means that even if one server goes down, the database can still be accessed from other servers.
                Cost-effectiveness: NoSQL databases can be more cost-effective than traditional relational databases, particularly for large-scale applications. They require less hardware and can be run on commodity hardware, reducing the overall cost of ownership.
                """
                ,
            "b": "Overall, NoSQL databases are a good choice for applications that require high scalability, flexibility, performance, and availability.",
            "c": """
                Data Integrity: SQL databases enforce data integrity through the use of constraints, which ensure that data is consistent and accurate. This helps prevent data corruption and ensures that data is reliable.
                Data Consistency: SQL databases use ACID (Atomicity, Consistency, Isolation, and Durability) transactions to ensure that data remains consistent, even in the face of concurrent access and updates.
                Standardization: SQL is a widely-used and well-established language for working with relational databases, which means that there is a large community of developers and resources available for working with SQL databases.
                Data Security: SQL databases offer robust security features, including role-based access control, encryption, and auditing, which help protect sensitive data from unauthorized access and breaches.
                Ad hoc queries: SQL databases offer the ability to perform
```

```
                  ad hoc queries, allowing users to extract insights from
                   large datasets quickly and easily.
              """,
            "d": "Overall, SQL databases are a good choice for
                applications that require high data integrity,
                consistency, security, and standardization. They are
                particularly well-suited for applications that require
                complex queries or data analysis.",
            "e": "An example of a NoSQL database would be HBase"
          }
        ],
        "scores": [0.8893801 , 0.9803964 , 0.93032193, 0.9715244 ,
            0.01551755]
    }
]
```

Code Listing A.18: Results for Relevance Test 3 (JSON Array).

```
[
    {
        "question": "What is JavaScript?",
        "answers": [
            { "a": "JavaScript is a programming language",
              "b": "JavaScript is a powerful language that is easy to learn
                   and use, with a syntax that is similar to other
                  programming languages such as C++ and Java."",
              "c": "JavaScript is a highly versatile language that can be
                  used in a wide range of contexts, including web
                  development, desktop and mobile app development, game
                  development, and even server-side scripting.",
              "d": "JavaScript can be used to create a wide variety of
                  interactive features on web pages, such as drop-down
                  menus, image carousels, form validation, and much more."
            }
        ],
        "scores": [0.6375351 , 0.25116247, 0.23333578, 0.01842343]
    }
]
```

Code Listing A.19: Results for Relevance Test 4 (JSON Array).

```
[
    {
```

```
        "question": "When did Prince die?",
        "answers": [
            { "a": "Prince, the American singer, songwriter, and musician, passed away on April 21, 2016.", ,
              "b": "Prince passed away on April 21, 2016.",
              "c": "Prince was an American singer, songwriter, and musician",
              "d": "Michael Jackson, the American singer, songwriter, and dancer, passed away on June 25, 2009.",
              "e": "Prince was born on June 7, 1958"
            }
        ],
        "scores": [0.9978123 , 0.9924223 , 0.00165526, 0.77990764, 0.99182737]
    }
]
```