Disinformation on Social Media: Truthsayer

Ren Jeik Ong
Technical University Munich
renjeik.ong@tum.de

Habib Bartu Gökalp
Technical University Munich
hbartu.goekalp@tum.de

Bilgehan Emiral
Technical University Munich
bilgehan.emiral@tum.de

Fabian Greif
Technical University Munich
fabian.greif@tum.de

Niyazi Ülke
Technical University Munich
niyazi.uelke@tum.de

Abstract

Disinformation on social media can cause significant harm. To address this, we present "Truthsayer," a tool that assesses the truthfulness of YouTube videos. Using Natural Language Processing (NLP), the tool automatically analyzes video content and verifies its accuracy against information from Wikipedia. The result is a browser extension that provides real-time feedback, helping users identify false reports while watching videos.

1 Introduction

In today's digital age, social media is a primary information source for millions globally. However, the rise of disinformation poses significant challenges by shaping public opinion and influencing events like elections by polarizing society (Tucker et al., 2018). The rapid spread of false content on platforms like YouTube can also erode trust in reliable sources and complicate emergency management, as was evident during the COVID-19 pandemic (Caceres et al., 2022).

To tackle this issue, we have developed "Truthsayer," a tool that assesses the truthfulness of YouTube videos. Using advanced Natural Language Processing (NLP), Truthsayer analyzes video content and cross-references it with verified information from Wikipedia to determine accuracy.

As a browser extension, Truthsayer provides real-time feedback on content reliability, helping users make informed decisions about the information they consume. This paper explores the development and implementation of Truthsayer, detailing how it addresses the challenges of disinformation and contributes to maintaining the integrity of online information.

2 Methodology

This section outlines the methodology used to develop and implement a browser extension designed to propose a solution effort for misinformation in YouTube videos. The solution involves a series of steps, from data extraction to claim verification, culminating in presenting the results to users through the extension interface.

2.1 The "Truthsayer" tool - Frontend and Backend

The foundation of the tool comprises a front-end application in the form of a Google Chrome browser extension and a back-end application in the form of a Django application (Django Software Foundation, 2024a). Communication between these two applications occurs through API access via network. These two applications are described below.

The browser extension, i.e. the front end, is the entry point for the user. This extension must first be loaded into the browser, which is done for the prototype with the help of developer tools (Google Chrome, 2022a). Then, the application can be used. The extension checks whether the user has opened a valid YouTube video and issues an error message or warning if necessary. If a correct video has been opened, a service worker in the background is employed to search for the current timestamp of the video (Google Chrome, 2022b). The url and timestamp are then sent to the backend application via a GET request. After processing in the background, a response is sent back in the form of JSON data. These data contain the status of the processing (whether it is successful or not) and a description of the results and the results themselves. This information is displayed in the user interface. The design of this extension is based on Google's Materials 3 style guidelines (Google, 2024). In the user interface itself, it can be seen that the title and logo of the application are displayed at the top, along with another icon. This icon represents the status, i.e. in the event of errors or warnings, it is highlighted in color and typographically. A simple black question mark is displayed during normal

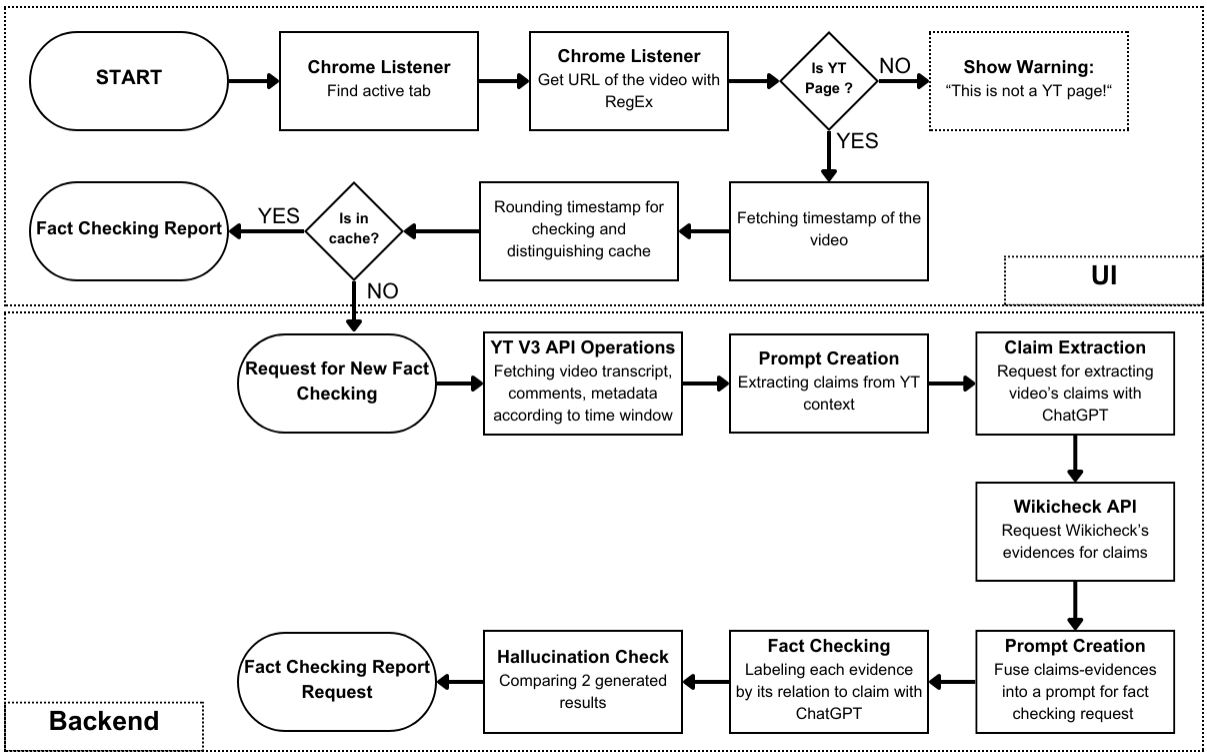

Figure 1: Workflow of TS

operation. When the "Truthsayer" is opened with a valid YouTube video, a loading circle is displayed first, which disappears after successful loading and displays the evaluated prompts in the form of a list. Each entry in this list contains descriptions and, if required, corresponding links for verification and a status bar with a score between 0 and 100, which indicates the trueness of the statement. Thus, the application does not have to recreate all requests each time; therefore, there is an internal memory that stores the results over a 10-minute interval and can retrieve them as required.

The structure of a frontend application from a technical perspective consists of a "mainfest.json" file that describes the complete structure of the application. This refers to the HTML page to be used, in our case "main.html", as well as to the service worker, in this case "background.js" and to image files such as our logo. In our "main.html" system, we structurally design the user interface and use the "styles.css" file for the corresponding design. Our communication with the backend application occurs via the linked "scripts.js" file.

The backend application is a Django application, which is a fully functional Python server (Django Software Foundation, 2024a). Only a fraction of the server's functionality is used for our prototype. We follow the Django documentation (Django Software Foundation, 2024b) for its creation and implementation. For this purpose, we create a single request, the "index(request)", in the "views.py" file. In the "urls.py" we have referred to this request, which can be called up directly via the server's url. An exact description of how results are calculated from the url and timestamp of YouTube videos can be found in the following sections.

2.2 Data Extraction from YouTube API

The first step in the pipeline involves retrieving video data from YouTube. The YouTube Data API is utilized to obtain relevant information, including the video title, description, transcripts (where available), and associated metadata. The API enables us to gather the necessary textual content that serves as the input for subsequent processing steps. To make data gathering more efficient, a 10 minutes time window based on the current timestamp of the browser media player is used as shown in Equation 1. Therefore, unnecessary text data in longer videos is avoided, and the fact-checking process focuses on the content that the user is currently viewing.

$$Window = [timestamp - 5, timestamp + 5] \quad (1)$$

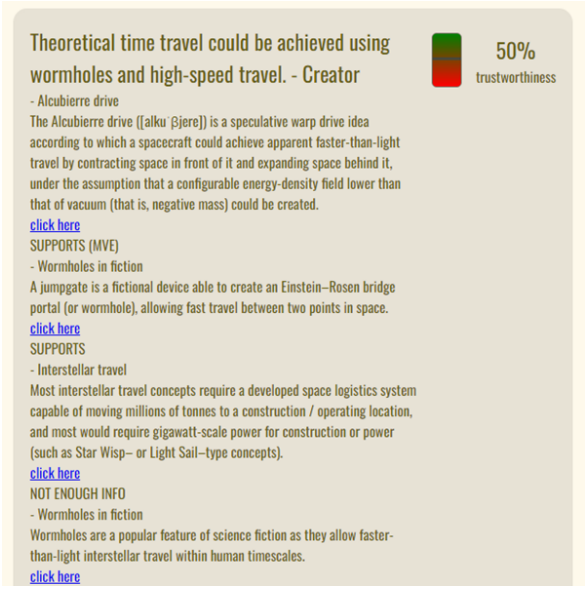

Figure 2: Screenshot of the frontend of the browser extension

2.3 Claim Extraction Using GPT with One-Shot Prompting

Once the video data is retrieved, the next step is to extract specific claims made within the video content. This is achieved by leveraging GPT (Roumeliotis and Tselikas, 2023), a state-of-the-art language model, to identify and extract statements that appear to be factual claims. The model scans the video title, description, and transcript (or subtitles if available) to isolate sentences or phrases that present verifiable information, distinguishing them from non-factual content such as opinions, questions, or entertainment. This process extracts the five most significant claims from the concatenated video text. To ensure consistency and focus, particularly on straightforward, potentially scientific, and verifiable claims rather than personal opinions or hard-to-verify statements, we adopt the one-shot prompting technique (IV et al., 2021).

2.3.1 One-Shot Prompting Technique in Claim Extraction

One-shot prompting is a technique used in natural language processing (NLP) to guide a language model like GPT in generating consistent and accurate outputs based on a single example. In our project, one-shot prompting is employed to help GPT accurately extract claims from YouTube videos. The model is provided with a single, well-crafted example that illustrates the desired format, structure, and reasoning to be applied to new inputs. This example acts as a template or guide for the model, helping it to understand the task at hand and produce outputs that align with the expected outcome.

2.3.2 Implementation of One-Shot Prompting in Claim Extraction

For the claim extraction task, we provid GPT with a specific example of how to distinguish a factual claim from other types of content within a video transcript. The example demonstrates how to identify sentences that present verifiable information, such as statistics, historical facts, or statements that can be checked against reliable sources.

For instance, if a transcript includes the sentence, "Global temperatures have risen by 1.5 degrees Celsius over the past century," the one-shot prompt would illustrate this as a factual claim, highlighting it as a statement that could be verified. Conversely, if the transcript includes, "I think the weather has been strange lately," the prompt would show that this is not a factual claim but rather an opinion and therefore should not be extracted.The claim extraction function works as follows:

Prompt Creation: A prompt is created by combining the video transcript with the predefined example. The example serves as a template, illustrating the type of claims that should be extracted.

Model Execution: The prompt is then fed into GPT. The model processes the video transcript, using the example as a guide, and extracts up to five of the most significant claims that are straightforward, objective, and factually significant.

Claim Validation: The extracted claims are checked to ensure they meet the expected format and criteria. If the claims are not in the correct format or are insufficiently precise, the process is repeated up to a set number of attempts.

By providing this single example, the model could generalize and apply the same logic to identify claims in a wide variety of contexts and videos. This approach ensures that the model consistently extracts relevant factual claims while filtering out opinions, rhetorical questions, or other non-verifiable content.

2.3.3 Advantages of One-Shot Prompting in Claim Extraction

Usage of one-shot prompting provides several advantages for claim extraction stage.

Precision: One-shot prompting ensures that only relevant factual claims were extracted, minimizing the inclusion of non-factual content.

Consistency: The use of a single example ensures that the model consistently applies the same criteria across different video transcripts, leading to uniformity in claim extraction (Liu et al., 2021).

Adaptability: With one well-crafted example, the model could adapt to different styles of language and content, making it effective across various types of videos and subjects.

In summary, one-shot prompting plays a vital role in the claim extraction phase by guiding GPT to accurately and consistently identify factual claims in YouTube video transcripts. This step is essential in ensuring that the subsequent fact-checking process is based on correctly identified claims, ultimately enhancing the reliability and effectiveness of our browser extension. At the end of the claim extraction process, a list of five verifiable claims is generated from the video data and passed further in the pipeline.

2.4 Evidence Retrieval via WikiCheck API

With a list of extracted claims, the next phase involves finding evidence to support or refute these claims. For this purpose, the WikiCheck API is employed (Trokhymovych and Saez-Trumper, 2021). This API searches a large database of information available on Wikipedia. Each claim is cross-referenced against this database to find the most relevant evidence. The API returns the 5 most relevant information that either corroborates or challenges the claim based on its search results, providing the foundation for the subsequent fact-checking process. Selection of 5 most significant evidences is made with a relevancy score also provided by the API. Evidence information contains the title of the Wikipedia article, relevant sentence within the article, a label that states whether the evidence supports or refutes the claim. Additionally, the URL for the Wikipedia article is also added later into this data by with use of the article title. However, the labels for evidences predicted by WikiCheck did not satisfy the expectations in terms of accuracy and relevance. Thus, these labels are discarded from the evidence information.

WikiCheck model architecture consists of 2 parts which are Model Level One: Wikipedia Search API and Model Level Two: Natural Language Inference (NLI) Model (Trokhymovych and Saez-Trumper, 2021). Model Level One is responsible for performing a full-text search through the entire English Wikipedia. The search is conducted using the MediaWiki API. The system enhances the query using Named Entity Recognition (NER) to improve search recall. This level's primary function is to select candidate articles from Wikipedia related to the input claim. After the candidate articles are selected, Model Level One compares the input claim with the retrieved sentences (hypotheses) from Wikipedia. This comparison determines whether the claim is supported, refuted, or if there is not enough information to decide. The architecture employs a Siamese network with a BERT-like model(Devlin et al., 2019) as a trainable encoder for sentences. The model is trained to encode the claim and hypotheses into embeddings and then compare them to classify the claim.

2.5 Fact-Checking Process

After extracting claims from video content, the next step is to validate these claims. The fact-checking process involves evaluating the extracted claims against evidence sourced from Wikipedia and assessed by GPT. This ensures that each claim is backed by the evidences. Each evidence is labeled for whether it supports, refutes, or irrelevant to the claim.

2.5.1 Fact-Checking Process Overview

The fact-checking system involves analyzing each claim by comparing it with corresponding evidence. This is achieved through a structured methodology that employs a one-shot prompting technique to guide GPT in labeling evidence as supporting, refuting, or inconclusive regarding the claim. Here's how the process unfolds:

Prompt Creation: A detailed prompt is constructed to guide GPT in evaluating claims. This prompt includes instructions for analyzing the evidence against each claim, labeling the evidence accordingly, and calculating a final score based on the provided support and refutation.

Model Execution: The prompt, along with the evidence, is sent to GPT. The model processes this input to determine the relevance and accuracy of each piece of evidence in relation to the claim. It then categorizes each piece of evidence and computes a final score.

Result Parsing: The output from GPT is parsed into a structured DataFrame. This function organizes the results, separating claims, evidence, labels, and scores for easy interpretation.

Final Scoring: Each claim receives a score based on the proportion of supporting evidence

compared to the total number of relevant pieces of evidence. The score is calculated with the Equation 2:

- S: Number of evidences that **support** the claim.
- R: Number of evidences that **refute** the claim.
- N: Number of evidences that are **not enough information** to assess the claim.

$$Score = \frac{S}{S + R + N} \tag{2}$$

This scoring mechanism quantifies how well-supported a claim is, providing a clear measure of its validity.

2.5.2 How One-Shot Prompting is Used in Fact-Checking

One-shot prompting is a key technique employed in the fact-checking process to maintain consistency and accuracy in labeling evidence. The following outlines the structure:

Example Provision: A specific example is provided to GPT to demonstrate how to evaluate a claim against evidence. This example illustrates how to label evidence as supportive, refuting, or inconclusive. For instance, if the claim is "The Eiffel Tower is the tallest structure in Paris" and the evidence is "The Eiffel Tower is the tallest structure in Paris at 330 meters," the example shows that this evidence supports the claim. Conversely, if the evidence states, "The Montparnasse Tower is the tallest structure in Paris," it would be labeled as refuting the claim.

Generalization: By using this single example, GPT learns to apply the same logic to all subsequent claim-evidence pairs. This ensures that the model consistently and accurately labels the evidence across various claims.

Advantages of One-Shot Prompting in Claim Extraction

Accuracy: One-shot prompting provides a clear example that guides the model in precisely evaluating and labeling the relationship between claims and evidence. This results in accurate classification of evidence as supportive, refutative, or inconclusive. Inclusion of "think step-by-step" in the prompt encourages GPT to generate results with better reasoning, booting accuracy (Kojima et al., 2022).

Consistency: The use of a single example ensures uniform application of reasoning across different claims and evidence pairs, minimizing inconsistencies and contradictions.

Efficiency: This method streamlines the fact-checking process, as the model learns the task with only one example. This efficiency speeds up processing without compromising accuracy.

Reliability: One-shot prompting enhances the reliability of the model's outputs by maintaining a structured approach. This consistency reduces the likelihood of erroneous fact-checking results.

In summary, one-shot prompting plays a crucial role in the fact-checking phase by guiding GPT to accurately and consistently evaluate and label the evidence for each claim. This structured approach ensures the reliability and effectiveness of the fact-checking process, providing accurate results that support the overall objective of the browser extension.

2.6 Presentation of Results on Browser Extension

Finally, the results of the fact-checking process are presented to the user through the browser extension interface. When a user watches a YouTube video, the extension automatically displays the fact-checking results for any claims detected in the video. The results are shown on the extension as a list of claims with their respective evidences, each labeled as supports, refutes, or not enough information. Evidences contain the title of the Wikipedia article with the relevant section used as the actual evidence. Users can click on links to view corresponding Wikipedia articles, or download the whole fact-checking result as a text file for further examination.

This methodology ensures a comprehensive approach to addressing disinformation on social media platforms, providing users with reliable tools to discern the accuracy of the information presented in YouTube videos.

3 Discussion & Results

A significant advantage of our approach is its ability to provide users with the sources of truth, enabling them to independently verify the information. We believe it is crucial to offer a level of explainability to users to ensure greater neutrality and transparency. Without this, the approach might be less ethical, as it could inadvertently reinforce

the biases inherent in the models that have been utilized.

One of the key challenges with large language models is the issue of hallucination, where models generate inaccurate or fictional content with confidence. To mitigate this problem, we provid GPT with contextual information from Wikipedia, which appears to be effective. However, while Wikipedia is one of the most reliable sources of information available online, it may not encompass every type of information shared in videos. Therefore, a weak point of our approach is the challenge of handling situations where no relevant information regarding the claims is found on Wikipedia. In such cases, the system may be unable to provide adequate validation, potentially leaving users without sufficient verification of the claims made in the videos.

To check and evaluate hallucination in generated text, an estimated lower bound on the self-contradiction rate (Dahl et al., 2024) is introduced using BARTScore (Yuan et al., 2021). In short, if a model produces two different answers to the same query, this is a strong indicator of hallucination or inconsistency. BARTScore score is based on negative log-likelihood of the reference text given the generated text and vice versa. A higher score indicates a lower chance of hallucination. One side note regarding *lower bound* is that there is a possibility of both answers being hallucinated.

10 pseudonymized YouTube videos are tabulated in Table 1, with generated reasonable claims for each video. The results show that these ten reliable videos have BARTScores ranging between -0.64 and -0.89. Thus, setting -0.90 as the threshold determines whether a video is hallucinated or non-hallucinated in hallucination checker.

Table 1: Analysis of the estimated lower bound on the hallucination rate: self-contradiction rate, evaluated using BARTScore (BF) on gpt-4o-mini model for YouTube videos.

YouTube video	BF Creator	BF Comments
Video 1	−0.8776	−0.8005
Video 2	−0.6571	−0.8997
Video 3	−0.6806	−0.8724
Video 4	−0.8319	−0.6863
Video 5	−0.8177	−0.7144
Video 6	−0.6651	−0.7303
Video 7	−0.7357	−0.6478
Video 8	−0.8191	−0.6655
Video 9	−0.8232	−0.6678
Video 10	−0.7826	−0.6644

4 Conclusion & Future Work

In a nutshell, we developed a disinformation detection tool that can be integrated as a browser extension. This allows users to automatically validate the claims presented in videos. Given that disinformation is a critical ethical issue, we believe our work addresses a significant gap, particularly in the video domain, where substantial efforts have been lacking.

For future work, additional sources of truth can be integrated into our system to enhance its reliability and coverage. Furthermore, the WikiCheck API can be improved or potentially replaced with a more robust and better-performing system. In addition, with the abundance of resources in YouTube videos, creating a knowledge graph would be helpful for retrieving information.

As a further improvement, we can enhance claim quality by generalizing several claims based on the original one. For example, a claim like "climate change effects" could be expanded into "impact of climate change", "consequences of global warming", or "effects of environmental changes". This will enhance retrieval results in the WikiCheck model. We could conduct an ablation study by removing Wikipedia URLs and other unhelpful attributes when feeding input text into a seq2seq model. Finally, instead of extracting claim statements, we could formulate questions, which might make the query more effective when feeding them into a seq2seq model.

References

Maria Mercedes Ferreira Caceres, Juan Pablo Sosa, Jannel A Lawrence, Cristina Sestacovschi, Atiyah Tidd-Johnson, Muhammad Haseeb UI Rasool, Vinay Kumar Gadamidi, Saleha Ozair, Krunal Pandav, Claudia Cuevas-Lou, Matthew Parrish, Ivan Rodriguez, and Javier Perez Fernandez. 2022. The impact of misinformation on the covid-19 pandemic. *AIMS Public Health*, 9(2):262–277.

Matthew Dahl, Varun Magesh, Mirac Suzgun, and Daniel E Ho. 2024. Large legal fictions: Profiling legal hallucinations in large language models. *Journal of Legal Analysis*, 16(1):64–93.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages

4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Django Software Foundation. 2024a. Django. Accessed at 15.08.2024.

Django Software Foundation. 2024b. Django documentation. Accessed at 15.08.2024.

Google. 2024. Material design. Accessed at 15.08.2024.

Google Chrome. 2022a. Hello world extension. Accessed at 15.08.2024.

Google Chrome. 2022b. Inject scripts into the active tab. Accessed at 15.08.2024.

Robert L Logan IV, Ivana Balavzevi'c, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. 2021. Cutting down on prompts and parameters: Simple few-shot learning with language models. pages 2824–2835.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55:1 – 35.

Konstantinos I. Roumeliotis and Nikolaos D. Tselikas. 2023. Chatgpt and open-ai models: A preliminary review. *Future Internet*, 15(6).

Mykola Trokhymovych and Diego Saez-Trumper. 2021. Wikicheck: An end-to-end open source automatic fact-checking api based on wikipedia. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 4155–4164, New York, NY, USA. Association for Computing Machinery.

Joshua Tucker, Andrew Guess, Pablo Barbera, Cristian Vaccari, Alexandra Siegel, Sergey Sanovich, Denis Stukal, and Brendan Nyhan. 2018. Social media, political polarization, and political disinformation: A review of the scientific literature. *SSRN Electronic Journal*.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. *CoRR*, abs/2106.11520.