**To-Do: Front Page (Daniel)**

**Abstract**

Text-generating applications like chatGPT have been seeing more usage as time goes by. Not only that, but there are also no doubts that much better text-generators await us in the future, capable of delivering far better results. Because of that, it is only a matter of time before much of the text we see on the internet starts to be authored by such software applications, raising concerns about originality. As such, many would like to have a guarantee that the text they are reading had a human hand behind it.

This report describes the development process of an application capable of distinguishing between Artificial Intelligence (AI) and human text whilst also being user-friendly, easy to use and effective at accomplishing its goal.

**Acknowledgements**

# Alexandre, Ricardo, Daniel, João.

**Contents**

Table of Contents

**1 Introduction**

* 1. Context
  2. Motivation
  3. Goals
  4. Document Structure

1. **State of the Art**
   1. Related Projects
      1. GPT-Zero
      2. Originality.AI
      3. Open AI’s AI Text Classifier
   2. Technology
   3. Conclusions
2. **System Requirements and Architecture**
   1. System Requirements
      1. Requirements Elicitation
      2. Context Description
      3. Actors
      4. Use Cases
      5. Non-Functional Requirements
   2. System Architecture
      1. Technological Model

**Table of Figures**

Table of Contents

1. **Introduction**
   1. System Architecture
2. **State of the Art**

**3. System Architecture**

3.1 Use Case Model

3.2 Domain Model

**Abbreviations**

**AI** Artificial Intelligence

**LM** Language Models

**UX** User Experience

Chapter 1

**Introduction**

With the advent and proliferation of text-generating AIs comes many concerns about originality and search-result quality. Users of search engines prefer to read human-authored articles about the topic they are searching rather than AI-generated articles that are often considered to be regurgitated, low quality and spammy when compared to articles written by humans.

Websites that rely on AI to generate their articles know this, so they try to disguise their robotic articles as human-authored articles, normally by attaching a (normally fake) name and face to its preface. This makes it hard for any normal person to reliably make the distinction.

Not only, but teachers whose jobs is to evaluate work in the form of written assignments also face the problem of having to differentiate between students who actually wrote their assignments and students who generated their assignments with AI.

* 1. **Context**

This is a software development project that focuses on AI, specifically the branch of machine learning and neural networks.

* 1. **Motivation**

As programmers, it can be hard to get a good answer for a bug or a code question as a lot of AI-generated pages clutter most search engines. Being able to steer clear of AI-using websites would greatly help us improve our productivity.

* 1. **Goals**

The goal of this project is to create a functional, top-of-the-line AI capable of correctly distinguishing between human and AI-generated text.

Chapter 2

**State of the Art**

* 1. **Related Projects**

The goal of this section is to present other projects or applications whose domain intersects with this project.

* + 1. **GPT-Zero**

GPT-Zero is a website where you paste text or upload documents to see if it was AI-generated. Has the best detection rate of all. Has a minimum of 250 characters, it only shows the first 5000 characters in the free version, and it does not give a percentage of confidence, it only shows the perplexity and burstiness scores alongside a general message of the likelihood of your text being written by an AI.

* + 1. **Originality.AI**

Is a website and an extension used to detect both AI written content and plagiarism. Has a poor AI detection rate. Costs credits to scan documents, with credits costing money. Claims to have a full website scan coming soon.

* + 1. **OpenAI’s AI Text Classifier**

Website very similar to GPT-Zero except it has an awful detection rate, as flipping a coin has a better chance of detecting if a text was written by an AI or not.

* 1. **Technology**

For now, we are limited by our language model’s and neural networks, but the application will be built in a way to allow for introduction of more complex models capable of differing results.

**2.3 Conclusions**

This can only be done by the end of the project, so this is just filler text.

Chapter 3

**System Requirements and Architecture**

* 1. **System Requirements**

This section presents the system requirements specification, as a result of the first phase of the prototype development.

* + 1. **Requirements Elicitation**

After a lengthy reunion with the stakeholders and the team of developers, various requirements were brainstormed from the perspectives of the end-user, external developers, and administrators.

* + 1. **Context Description**

This section contains concise descriptions on how the application is expected to be used by different users.

An end user, such as a teacher evaluating a student’s body of work, will open a website/pdf file and either will, automatically or manually (depending on their preferences), have their text highlighted, depicting which sentences the application notices as being written by an AI.

To an external developer, the system’s architecture was designed in such a way that implementing new languages is very simple.

* + 1. **Actors**

**End User:** Represents the user, who can take on several roles when utilizing the application. Accesses it through the extension and is able to get all the AI verification needed.

**Developer:** Capable of adding Language Models as needed to the system.

**Administrator:** Accepts or denies Language Models, can remove/disable them as needed.

* + 1. **Use Cases**

Figure 3.1 presents all the use cases of the system.

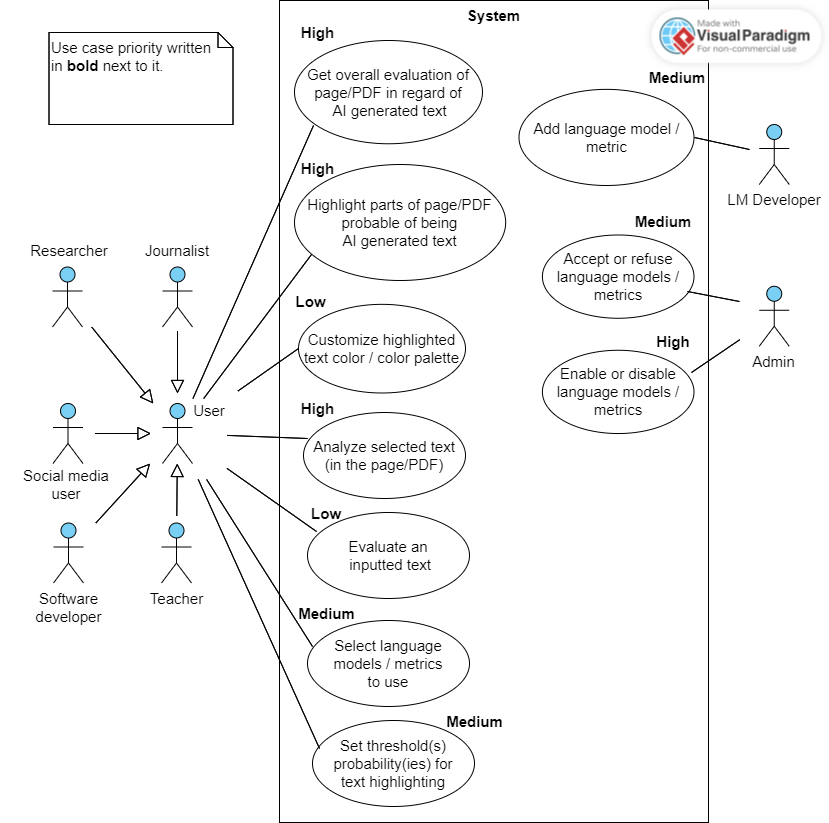
****

Figure 3.1: Use Case Model

The following list describes each use case, from highest to lowest priority:

**Analyze selected text:** The user asks the extension to analyze the text it has selected, and the application responds with a highlighting color corresponding to the probability of that text being written by an AI.

**Highlight parts of page/PDF:** The user can use the application, automatically or when prompted, to intelligently highlight certain parts of a webpage or PDF file with the colors corresponding to the probability of that text being written by an AI.

**Get overall evaluation of page/PDF:** The user can use the application to analyze a page for AI-generated text, and the application will display a percentage representing how much of what was analyzed corresponds with human or AI text.

**Enable/Disable LMs/Metrics:** The administrators, at their discretion (e.g., when a LM is outdated), are able to easily enable/disable a LM/metric.

**Accept/Refuse LMs/Metrics:** The administrators, at their discretion, are able to accept or refuse incoming LM/metric submissions from external developers.

**Add LM/Metric:** An external developer is able to submit their own LMs/metrics to be accepted or refused by the administrators.

**Select LMs/Metrics to use:** The user can customize their application’s functionality and capabilities by modifying which LMs/metrics their application analyzes with.

**Set threshold probabilities for text highlighting:** The use can customize at which thresholds certain colors appear, e.g., 80% or above = red highlighting.

**Evaluate an inputted text:** The user can input text directly into our website or application, similar to the state-of-the-art competitors, and have their text analyzed that way.

**Customize color palette:** The user is able to modify the colors that appear at certain thresholds after text analysis.

* + 1. **Non-Functional Requirements**

The following list presents the non-functional requirements that our application will have to exhibit:

**Usability**: Knowing that the competing applications all have subpar usability, our number one goal is to provide the best user experience and not necessarily better analysis. Also, because this application is not meant for any one specific demographic, we have to account for all possibilities, including users who are not experienced with using a computer. **Priority:** High

**Flexibility:** The addition and removal of LMs is crucial for keeping our application up to date with the constantly changing landscape of the machine learning industry. **Priority:** High

**Reliability:** The core functionality of our application, its capability of detecting synthetic text, must be, at the very least, on par with a competitor like GPT-Zero. There is no reason to use an app with a good user experience if it can’t do what you ask it to, which is why it cannot fall on the same level as Originality.AI or OpenAI’s AI Detector. At the same time, there isn’t much of a need to surpass GPT-Zero, as our focus should be more on the UX. **Priority:** Medium

**Performance:** Reliable LMs can be slow. Although users won’t be analyzing more than one or two pages per session, time spent analyzing can degrade the user’s experience. To combat this, the process of analyzing text must be quick. **Priority:** Medium

**Compatibility:** The application will be built for Chromium-based browsers. It can always be expanded to other browsers, but that is not as important as the other non-functional requirements. **Priority:** Low

**Privacy:** The user should be confident his data, such as his browser’s information or the websites he visits, is not being stored and used for any purpose besides improving our application, like selling it off to the highest bidder. **Priority:** Low

**Maintenance:** The code should be written in future-proof ways, as this application is intended to be used beyond the scope of this project. All issues should be tracked and recorded, for future programmers to be aware of them. **Priority:** Low

* 1. **System Architecture**

This section presents an overview of the system architecture, describing its domain model and underlying relationships.

* + 1. **Technological Model**

The technological model on Figure 3.2 gives an overview of the technologies used by the system and how they interact with each other.

A picture containing graphical user interface

Description automatically generated

Figure 3.2: Domain Model

The extension, accessed through it’s Graphical User Interface, contacts the Logic Server through an API by sending requests for text to be analyzed. The logic server calculates the probability of the text being synthetic using various LMs (only 2 are shown but more can be added), and returns the probabilities and other data to the extension to be used for highlighting. Optionally, a database can be used to store information for improving the application.