Beeg Data - Beeg Meme

Kacper Grzymkowski, Jakub Fołtyn, Mikołaj Malec, Illia Tesliuk

Introduction

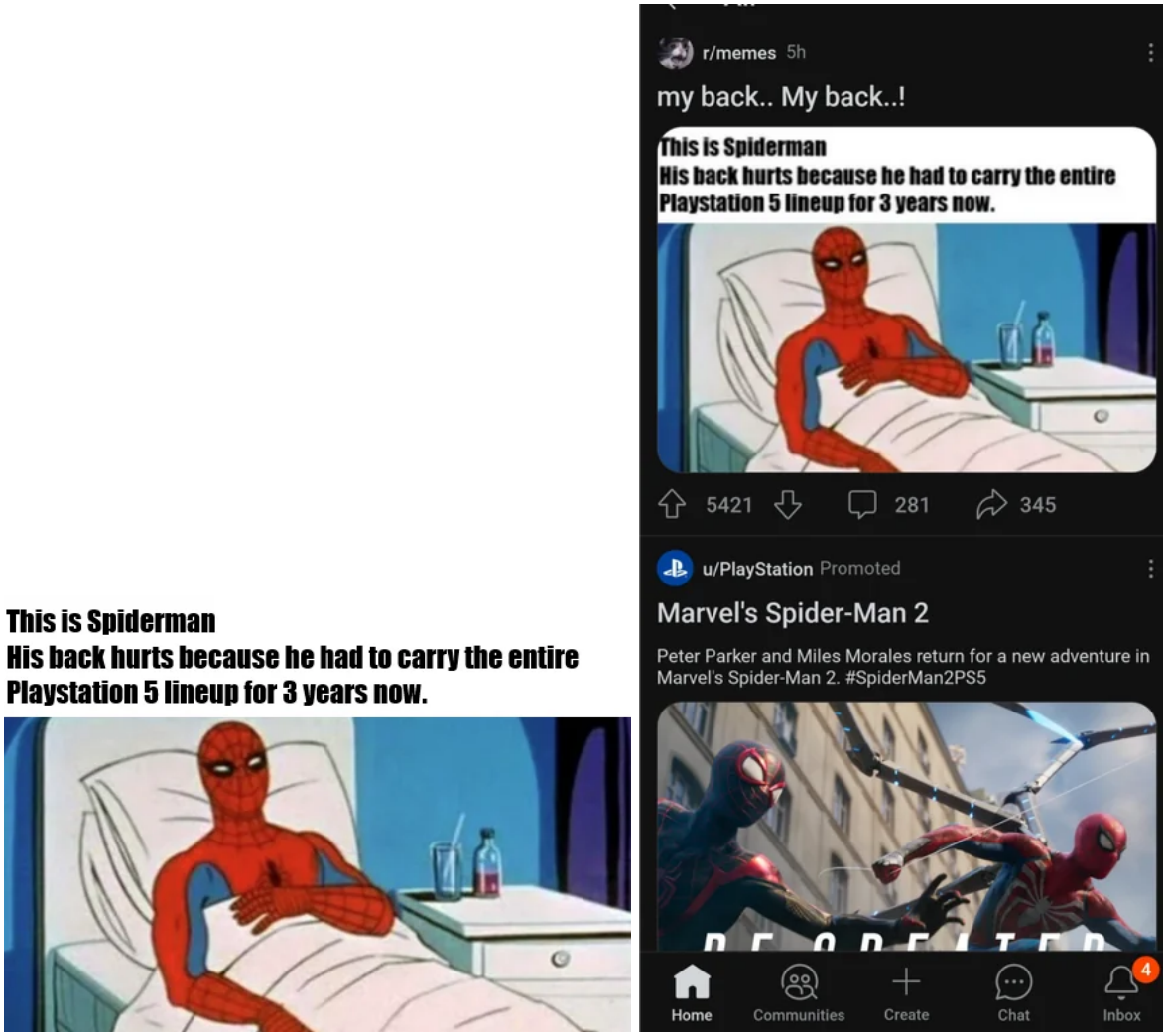
This document represents the plan and all the preliminary requirements for the project realized during the Big Data Analysis course. The project, as described in this document, will concern the issue of classifying and performing sentiment analysis on humorous images posted on the internet. The following sections will describe all the various aspects of this project, such as benefits of project deliverables from user perspective, data sources, data processing methods and the plan for the solution's architecture. The division of work among authors will also be provided at the end of this document.

Summary

The application is a comprehensive and user-friendly platform for exploring and analyzing internet memes. It offers a diverse meme library categorized by similarity, emotional impact, and metadata, ensuring easy meme discovery. Users can stay up-to-date with real-time trend analysis, allowing them to understand the latest meme trends and user sentiments. The app is a powerful tool for meme enthusiasts, marketers, and researchers alike.

Benefits from finishing the project for users

- **Meme creator support**: Meme creators can use our app to find the most promising memes and trends, allowing them to discover which memes are most likely to yield the highest reach and engagement.
- **Comprehensive trend analysis**: Researchers can delve into sentiment analysis, meme effectiveness, and user sentiment. They will gain insights into not only what memes are trending but also how they resonate with users on a daily basis.
- **Targeted advertisement creation**: By analyzing the meme library categorized by similarity, emotional impact, and metadata, advertising developers can create highly targeted and engaging advertisements. This ensures that their ads resonate with the intended audience, leading to higher conversion rates.



- Example of ad placement

Data Sources

In this section we will briefly describe all the data sources considered in this project. We have decided to use 4 different data sources in total. They all contain some information regarding the topic of humorous internet images (so called “memes”). 3 of these sources will provide us with the images themselves – using different sites will ensure variety and a steadier stream of images to process (as multiple sites are going to heighten the chances that the images will arrive in a continuous manner). We will now describe each data source in a little more detail.

Reddit.com

Reddit is a fairly popular social platform. This site is divided into various subpages, each corresponding to a different topic and include different communities. These subpages are called “subreddits”, and are denoted by **r\<name_of_the_subreddit>**. Our subreddit of interest will be

one called **r/memes**. As the name indicates, it includes memes as posted by the users of this subreddit. We are going to access said images using the official Reddit API.
This API requires authentication (OAuth2), and enables 100 queries per minute (after authenticating). The data may be accessed via URL (such as https://www.reddit.com/r/cats/comments/l0ixc9/luna_has_beautiful_eyes.json). The response is a JSON object, which contains URL for the post image, which can be then scraped.

Ifunny.co

On Ifunny, registered users can post hand-crafted (or taken from another source) memes tackling different topics, such as politics, sports, animals, current world events etc. To interact with it, we are going to use an unofficial Ifunny API wrapper for Python. Unfortunately, an official version of this API is not yet available. The fact that this API wrapper is unofficial will enable us to use it without considerable restrictions, but, on the other hand, the API itself is rather badly documented and is not maintained anymore. It is possible that in case of major issues with it, we will be forced to develop our own version. This API does require authentication (in form of registration on the website in question and retrieving an access token), howether, due to insufficient documentation it is not possible to access the information if any restrictions on the number of requests are present.

Imgur.com

Imgur is another site used for uploading memes, gifs and other images created by its users. We will use the Official Imgur API, most probably its Python wrapper (even though it is no longer supported). This API requires registration (OAuth 2.0) and is based on HTTP requests (with JSON responses). It enables approximately 12500 requests per day (so about 8 requests per minute). As is the case with other APIs, responses include URLs for images to be scraped (and it is possible to access newly posted images as well).

Knowyourmeme.com

Knowyourmeme.com contrary to all the previous data sources, this website will not be used to access memes in real-time. It is because, in addition to being another website to post humorous images on, it is also an extensive database, containing all the meta-information about memes: such as their sentiment, meaning, and example situations in which they can be used. Such information may be helpful during development of our project, as it may provide us with some meaningful information about the images scraped from other sources. To scrape information from it, we may use an unofficial Python scraper for Knowyourmeme. It is a library that enables us to easily retrieve image URLs from posts made on Knowyourmeme site (as well as additional info about them). Documentation does not mention any authentication or restriction information, howether, the official Github page does mention the possibility of being banned by the KnowYourMeme website.

Volume of the data

Unfortunately, it is not possible to accurately measure the volume of the data available in all of these sources, as the data is created by the users interacting with these websites. However, as all these sites are relatively popular, it is safe to assume that there are going to be at least several images uploaded per minute on each of these websites.

Stream and Batch processing

Kappa architecture uses a single data processing system to handle both batch processing and stream processing workloads.

The system receives data in the form of an JSON object filled with post metadata which includes some basic information such as timestamp, author's name, number of likes, number of comments, URL of the post image, etc. Images are scraped from the corresponding URLs saved in the storage in a raw form.

Since memes are received from several data sources, their JSON objects can contain different metadata. Therefore, useless key fields have to be filtered out and important post information has to be unified to a single format to be prepared for aggregation operations.

Post metadata is not used in data mining models, so meme images serve as the sole inputs to the models pipeline. Before being fed into the pipeline, each image undergoes basic preprocessing such as normalization and resizing. In order for the models to avoid overfitting and improve generalization performance, data augmentation techniques can be used on the training dataset (e.g. flipping, cropping, rotation, color jittering, etc.).

For each input image the models produce three outputs:

- Cluster id the meme belongs to
- List of entities (people names, places, events, etc.) of the meme's text
- Sentiment ("positive", "negative", "neutral", etc.) of the meme's text

Together with the post metadata, model outputs can be aggregated and used for various analysis tasks, e.g. number of memes made about a specific person in last 24 hours, sentiment ratio on an event like a football match, trending cluster of memes in a specific period of time, etc.

System architecture (outline)

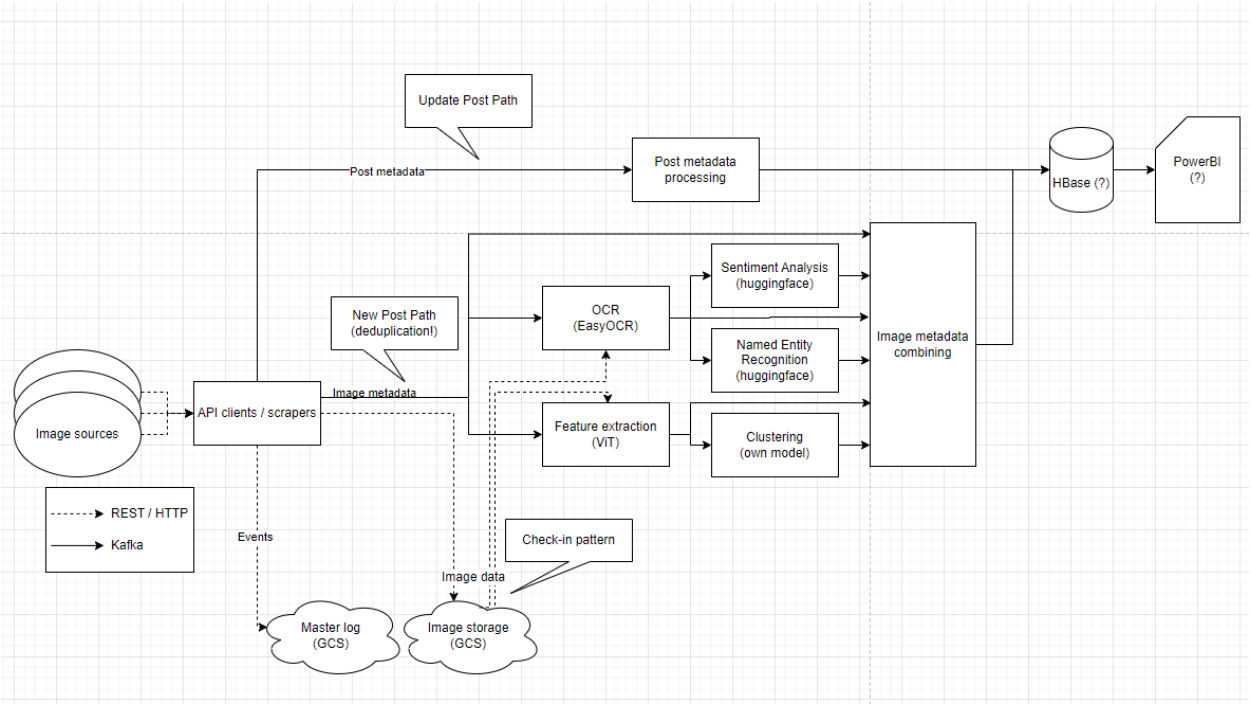
We chose the Kappa architecture model due to the focus on data mining and the rather uniform data. We decided to split the processing of Post and Image, as we think it's a safe assumption that the image of a post does not change, but the post information will change - for example, the number of Upvotes (or likes) a post gets might be valuable information. We do this because the image processing is expensive computationally, and is very unlikely to change through a post's life.

We will use Apache NiFi for ingestion, but we will likely make heavy use of Python scripts, with NiFi being used for orchestration. This is due to a large amount of libraries for collecting the data from the various API / scrape sources. The acquired image and post metadata will then be pushed into Apache Kafka. However, to ease the memory need of Kafka, we plan on using the "check-in" pattern, meaning the raw image files will be put into storage, and only an identifier will be sent forward in Kafka. We plan on using cloud storage (on GCP) for the master log as well as the main image storage. Further on, we will run many pre-trained models to mine information from the images. We plan on using PySpark Structured Streaming for this task, which we will also use to process the data further. Finally, we plan on storing the results in Apache HBase and present them in PowerBI, but due to compatibility issues between the two, we might opt for a different solution, possibly a small custom PoC.

We are considering a few alternative technologies. One is using Google Pub/Sub as a managed solution instead of Apache Kafka. Another is the choice of Database, where we are considering using Apache HBase, Apache Cassandra or even a managed solution like Google BigTable. We want to use an online learning algorithm for clustering but if that doesn't work then we will likely set up a spark job to train the model on data already collected.

Planned technologies:

- NiFi
- Kafka or Google Pub/Sub
- Google Cloud Storage
- PySpark
- HBase or Cassandra
- PowerBI or Custom solution



Work division

In this section we propose a planned work division among all the team members. It is important to note that information in this section is not final and may be subject to change during the project development.

- Kacper Grzymkowski
  - Integration and ownership of the application
  - Image feature extraction and clustering
  - Post metadata processing
  - Data extraction (one source)
- Mikołaj Malec
  - Database creation and management
  - Reporting layer
  - Data extraction (one source)
- Illia Tesliuk
  - Modeling: OCR, Sentiment Analysis, NER
  - Data extraction (one source)
- Jakub Fołtyn
  - Data transportation
  - Blob storage
  - Image metadata processing
  - Data extraction (one source)