University of Glasgow | School of Computing Science

Honours Individual Project Dissertation

"A Picture is worth a Thousand Words": Automatic Illustration of Text via Multimodal Interaction

Dominykas Meistas

January 1, 2022

Abstract

A picture is often useful to illustrate a concept, e.g., imagine you would want to describe some concept to an audience, e.g., as a lecturer you may want to define vectors and the operations on them, or as an author of a paper on machine learning, you may want to describe a novel neural network architecture. Imagine a system, which extracts out the right query terms for you (based on your context) and retrieves a list of images for you to choose from to either directly use in your content or create a new image based on the ones relevant to your context. The main challenge to develop such a system is to learn a multimodal semantic space to represent both image and text. This project envisions to develop an interface (e.g. an IDE, which given a current context of text will conduct a Google Image search at the back-end with automatically formulated queries and retrieve a set of candidate images which could be used by the user to enhance the current document). This will also find applications in enhancing the content of slides with images and would help in rapid slideshow creation.

Contents

1 Introduction

1.1 Motivation

The text has always been one of the main ways to convey information. People read books, newspapers, articles, news portals, and other sources that contain primarily textual knowledge. Until a few decades ago, it was one of the only ways to communicate information to people. However, times are changing, and other modalities appear to enhance people's understanding of the topics that interest them. Such modes are linguistic/alphabetic, visual, aural, gestural, and spacial.[1] In this project, we will focus on the visual mode, which consists of images. Many say that an image is worth a thousand words. This statement means that one can express complicated and sophisticated ideas much easier using images, conveying the meaning more effectively than via a mere verbal description.

Our motivation is to implement a system that applies this theory to help people convey their thoughts in a multimodal way. A system that gathers the correct query terms from the text section, performs a query, and returns relevant images could be helpful for many people: ranging from students, composing a report for their coursework or making a slideshow presentation, to researchers, writing a scientific paper. Having both text and images helps readers perceive the information faster, without reading the whole paragraph of text, and improves their understanding by providing multiple information modalities. Such a system would also save the writer much time, as he would not need to perform the search queries by himself - the system would do this for him.

1.2 Problem Statement

Using images to illustrate a text section is an effective way to convey information to the readers. However, we need to introduce several intermediate steps essential for this system.

The first part of the workflow is to generate a library, called an index, that contains images and some contextual information about them. We need the image database to be vast and diverse so that it contains images from every category that the user might include in their work. Suppose the user writes a paper about a particular topic (e.g., dark matter). In that case, the system must have a broad enough collection to recommend at least a few images related to this topic. Contextual information, such as image captions, is needed so that it is possible to retrieve images based on that information. We focus on text-to-text comparison in our project, meaning that the system can only retrieve images based on their captions, implying that an image that does not have contextual information will never be retrieved.

A further task is to find the terms from the text section that accurately represent the passage. The more relevant images the system retrieves from the collection based on a query, the more accurate we consider the query to be. Suppose we query the system based on ubiquitous words (e.g., sight, place, environment) or the stop words (e.g., the, a, an). Such a query will not provide

[1] More information can be found at `https://openenglishatslcc.pressbooks.com/chapter/multi-modal-communication-writing-in-five-modes/`

images that accurately represent the paragraph because these words are too generic, and too many images match this query. On the other hand, using specific words (e.g., University of Glasgow, Microbiology, Game of Thrones) should return images of high relevance, as those words usually appear in specific documents that represent the topic accurately.

A considerable part of the process is information retrieval (commonly known as IR). IR is a process of obtaining information resources relevant to the user from the collection of those resources. The information retrieval process comprises an input query, a combination of words representing the documents that the user is trying to retrieve, and output, a collection of documents containing information. One of the biggest challenges in IR is finding the documents that most accurately represent the query. Another issue is finding out the order in which the documents should be displayed to the user, as more relevant documents should always appear above less relevant ones. To tackle these challenges, we introduce similarity, which is a measurement that indicates how relevant a document is to a specific query. It is common in IR to sort the relevant documents by their similarity in decreasing order to have the most relevant documents at the top and achieve good results.

Evaluation of the results is another problem that we need to discuss. Evaluation is a difficult task because it depends on the users whether they find the results relevant or not. Some users may say that the output represents their query accurately, while others may think the same results are irrelevant. It also depends on the query itself, as it may not always represent the user's intention. Users may not know the correct terms to form their queries, or they might misinterpret the meaning of certain words. A collection of images retrieved from the system may seem irrelevant to the user, while the query itself may be causing a misunderstanding between the user and the system.

A further issue is evaluating the accuracy of the model. One approach is to consider users' feedback, which is highly unreliable because of the issues mentioned above, such as multiple users perceiving the same results differently. Furthermore, it is impractical, expensive, and almost impossible to achieve because too many users are needed to gather enough data to observe accurate results; having only a few evaluators may introduce their bias into the evaluation data. Moreover, each time a change occurs, we would have to re-evaluate the system, making this approach unfeasible. In order to avoid these issues, we need to come up with automated evaluation methods that could easily be re-calculated. We can use standard IR performance metrics, such as precision and recall, to evaluate how well our model performs. Precision indicates the percentage of relevant retrieved images, while recall shows the ratio of relevant images retrieved out of the total relevant documents in the collection. These and other evaluation metrics that we will be using to evaluate our system will indicate how well our model performs.

There are also different ways to tackle the previously introduced *vocabulary mismatch*. The problem occurs when the query and the retrieved documents from the collection do not match because of the initial query's incomplete specification of the intended information by the user. A common technique to handle this issue is relevance feedback – use the initially returned results to gather user feedback and perform a new enhanced query. However, gathering users' explicit feedback is not optimal, as users are usually unwilling to spend time on that. For that reason, we introduce an alternative approach to enhance initial queries made by the users. Assuming that a certain number of top documents retrieved are relevant, we find other popular terms in those documents to expand the initial query. Using this approach, we improve the query without any interaction from the user.

1.3 Aims of the Project

The project's main goal is to build a system that extracts the right terms from the text section, performs the query, and returns images that accurately represent the text passage provided by

the user. Our other goals include testing different methods for extracting query terms, using a different number of query terms, applying similarities, relevance feedback, query expansion, and other standard IR techniques. We aim to compare these results to see which configuration works the best and explain why that is the case. We also plan to build an evaluation dataset that we will use to evaluate our model's performance.

1.4 Our Contributions

Our first contribution is the methodology. We created a system that retrieves relevant images based on the text section that the user inputs. We have tested out multiple configurations of the system, provided tables, graphs, and other forms of results to show our work and the reason for it.

Another novel contribution of ours is the evaluation datasets. These datasets are our own work and could be used for retrieval tasks by other researchers. Both evaluation datasets comprise the same 25 queries, but their ground truths are different.

1.4.1 Strict Evaluation Dataset

The first dataset we constructed consists of 25 text sections taken from 25 different Wikipedia pages. Each record contains a ground-truth set of images relevant to the text passage. We manually hand-picked these images to build an exceptionally accurate ground truth. The average length of a single text section in this dataset is 283 words, and each record contains, on average, 29 images in its ground truth. We will use this dataset later to evaluate our model.

1.4.2 Lenient Evaluation Dataset

This dataset comprises the identical 25 text sections as in the previous dataset. However, the ground truth of this dataset is different. We have built a program that retrieves all the images from the Wikipedia page and all Wikipedia pages referenced from the original one. We used this program to retrieve images from the Wikipedia page in which the text section appeared, as well as from all referenced Wikipedia pages. Consequently, the ground truth is significantly more extensive than in the previous dataset. On average, it contains over 11k images. This dataset also includes multiple layers of relevance.

The most important image is the original picture displayed next to the text section on the Wikipedia page. This image has the highest relevance because an image displayed next to a text passage on a Wikipedia page usually represents the paragraph very accurately.

The next level of relevance is all the images appearing on the same Wikipedia page as the initial paragraph. These images are less relevant than the original because they may contain more general information or a slightly different subtopic, but most images should still be highly relevant.

The least relevant images appear in the Wikipedia pages linked from the original page. We expect this collection to contain some relevant images; however, most of them might not be helpful to our model.

Following this approach provides a much bigger ground truth of images compared to the hand-picked dataset. The most relevant layer always contains one image, the second level on average contains 48 images, and the least important set of images contains over 11k images on average. This dataset will also be used to evaluate our model.

1.5 Research Questions

In this section, we formulate the specific research questions associated with the goals of our project.

The first question is related to the query terms retrieval. We are trying to retrieve the words from the paragraph that accurately represent the section. Finding the most relevant terms is essential because other system components depend on them, meaning that those components will perform ineffectively if we do not retrieve appropriate terms. The first research question is thus stated as follows.

RQ-1: *Given some context, how effectively can we extract the information need from that context?*

The second research question corresponds to retrieving relevant images based on contextual information. It depends on the previous component because it uses the retrieved terms to find the images that accurately represent them. The main goal of our system is to recommend the most relevant images to the user. Therefore, our second research question is as follows.

RQ-2: *Given the information need, how effectively can we retrieve the content of different modalities?*

To answer this question, we will be using different evaluation techniques to show us how well our system performs.

1.6 Dissertation Outline

The dissertation is structured as follows:

- **Chapter 2** provides previous work done on multimodal information retrieval and its relation to our project.
- **Chapter 3** discusses the proposed method to solve our research questions.
- **Chapter 4** analyses the details of our implementation.
- **Chapter 5** goes into detail about the experiments we had conducted and the reasons behind them.
- **Chapter 6** describes the results of our experiments.
- **Chapter 7** finalises the work done and discusses the possibility of future improvements.

2 Background

2.1 Information Retrieval with Verbose Queries

Gupta and Bendersky (2015) provides an immense amount of information about information retrieval using verbose queries. The book states that lengthy natural language queries have recently become widespread. They are used by dialogue systems, voice assistance on mobile phones, entity search engines. However, short queries still perform significantly better than their longer counterparts. Thus, it is crucial to find methods to handle verbose queries effectively to improve the information retrieval process. The book discusses different approaches to addressing such queries: weighing, reduction, expansion, segmentation, and reformulation. It combines multiple research pieces, provides an overview of proposed methods, discusses different applications where the usage of verbose queries could make a tremendous difference.

Our project was constructed around the idea of verbose queries. We have a long query, either user input or a section from Wikipedia, and we need to process it to retrieve relevant results. We build on top of the information from this book, using its knowledge and expertise to perform our experiments and evaluations.

2.2 Introduction to Information Retrieval

Manning et al. (2008) equips us with vast amounts of knowledge about information retrieval. It provides us with up-to-date information for gathering, indexing, and searching documents. Furthermore, it supplies us with methods for evaluating systems and an introduction to the use of machine learning methods on text collections.

The book describes multiple retrieval models, one of which we use in our project (BM25 similarity). It discusses the term scoring and weighing (including TF-IDF functions used in our model). Furthermore, it provides considerable information about evaluating our model, including which metrics to use and their purpose. The book also teaches about relevance feedback and query expansion which we use in our project. This book has a tremendous amount of information about IR, and I would highly recommend that whoever is interested in this topic to explore this book thoroughly.

2.3 Deep Visual-Semantic Embedding Model

Frome et al. (2013) paper discusses modern visual recognition systems and their limitations. It introduces the problem of those systems not being able to scale to large numbers of object categories because of the limitation of training data in the form of labelled images. Consequently, it presents a solution to this problem to use text data to train visual models and constrain their predictions. Furthermore, it presents a new deep visual-semantic embedding model trained to identify visual objects using labelled image data and semantic information gained from the unannotated text. Finally, it shows the results of this model and compares them to other state-of-the-art solutions.

Even though we did not implement the features suggested in this paper, it could be useful for a future extension of our model.

3 Proposed Methodology

This chapter will discuss our proposed methodology for the presented problem. The workflow diagram below (Figure 3.1) describes our system's workflow at the highest level. We will be considering the system as a whole and each component separately to build a foundation for our method and reason about the specific implementation details we have chosen.

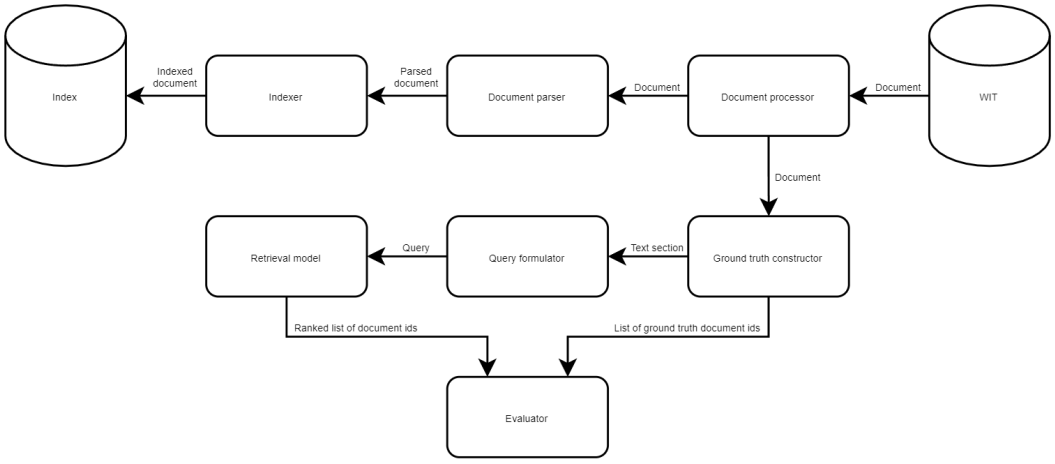

Figure 3.1: *A workflow diagram describing our system's methodology.*

3.1 System Components

This section will discuss each component separately to understand their purpose in our system and see how they interact with each other.

3.1.1 Document Processor

The first component we will discuss is the Document Processor. It is responsible for taking the documents from the dataset one by one and forwarding them to other components. In the diagram (Figure 3.1), we can see that it sends the documents in two directions. The one going to the left is towards building the index, which we will query to retrieve relevant images in the further stages of the workflow. The one moving downwards is to build the ground-truth dataset that we will use to evaluate our model.

3.1.2 Document Parser

Document Parser is an essential component in our system that parses each document received from the Document Processor and sends them forward to Indexer (Figure 3.1). The documents need parsing because the WIT Dataset contains information not beneficial for our project.

The first task of the parser is to remove all data entries from the dataset that contain non-English Wikipedia pages. This procedure is necessary because our model currently supports only English information. However, WIT Dataset supports 108 languages; thus, extending our system to support multiple languages could be a reasonable future extension.

A further task is to filter unnecessary fields from the documents. WIT Dataset documents contain 17 data columns: Wikipedia page language, URL of the Wikipedia page, URL of the image that represents a text section, Wikipedia page title, Wikipedia section title, hierarchical section title, and others. However, only three fields are necessary to build an index: the document ID, the URL of an image, and the contextual information describing the image. The dataset contains three different fields of contextual information that we could use to illustrate the pictures. The parser is responsible for choosing the most relevant and topical field available and using it to describe an image in the index. After parsing the data, the parser sends newly formatted documents to the Indexer component.

3.1.3 Indexer

The indexer is a component responsible for building an index for our system. The index is a collection of documents that we can query to retrieve relevant documents based on context. The indexer takes in the documents passed from the parser, containing three fields: the document's ID, the URL of the image we are indexing, and the contextual data describing the image. It then uses various techniques to construct the index. We will describe those techniques in this section.

One of the techniques is using an analyser. Text analysis is a technique used to automatically extract valuable insights from unstructured text data[1]. Using a text analyser is essential in our project because our system revolves around matching text to retrieve relevant information to some context. However, our system would not know that the words *image* and *images* have the same meaning without analysing the text because their endings would differ.

The first step in analysing text is applying a tokeniser. Tokeniser splits a phrase or a paragraph into smaller units, such as words or terms (called tokens). This makes it easier to perform text comparison and apply other essential techniques that we will soon discuss[2]. It also removes most punctuation symbols because they are unnecessary in text search algorithms. Having separate words makes it easier to accurately represent how many times each word appears in the document, which is one of the metrics when deciding if a document is related to a specific topic.

Another technique we can apply is possessive's filtering, which removes all possessives from words. Removing possessives from words is beneficial because words such as *person* and *person's* have the same meaning; therefore, they should be considered the same word. However, our system would not be able to match them because they are not identical. Removing the trailings of each word will increase the accuracy of our model because we will be able to count more accurately how many times a word appears in a specific document.

A further important filter we use is a lower case filter. This method normalises the token text to lowercase. Our model's performance improves because the capital letters do not change the word's meaning. Hence we should consider those words equal.

Another technique, arguably the most important one, is the stop words filter. The filter removes all the stop words from the text section (e.g., this, he, and, or, a). This procedure is crucial because stop words are more frequent than other terms in the text section, meaning that the retrieval component will retrieve documents based on those stop words. However, stop words do not capture any meaning of the contextual information, which would cause the model to recommend irrelevant images.

[1] More information about text analysis can be found at `https://monkeylearn.com/text-analysis/`

[2] More information about text tokenisation can be found at `https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-tokenizers.html`

The last filter, which is essential to our system, is the stemming filter. It reduces the words to their word stem, making similar words with different endings or beginnings appear as the same word. This technique drastically improves our retrieval process, as many words have the same stem (and the same meaning), but the system does not consider them identical.

After the indexer completes the steps mentioned above, it creates an index (described as a data component on the left in Figure 3.1). We will later use this index to get relevant images to some contextual information.

3.1.4 Ground Truth Constructor

This component is responsible for constructing the ground truth of our system. It is essential to have a ground truth to evaluate the model and measure how well it performs.

There are a few data fields that the ground truth must contain. Firstly, it needs to have contextual data for each document we include in our dataset. This contextual data is usually a paragraph similar to the text that a system user would input into the system, expecting to get relevant image recommendations. We are extracting these sections from Wikipedia because we already know at least one image that correlates to them (the image presented next to the paragraph on the Wikipedia page).

However, having such a small ground truth (containing one relevant image) is ineffective and challenging to evaluate. The system could retrieve ten great pictures relevant to some context but not include that specific image from the ground truth, and the evaluator would state that our method performs poorly.

These issues lead to another task of the ground truth constructor component. It needs to extend the dataset so that the ground truth includes multiple images. Having numerous pictures in the ground truth allows us to use more advanced evaluation metrics (e.g., MAP, NDCG) and improve the results of other evaluation metrics.

As mentioned in the introduction, we have built two evaluation datasets that we use to evaluate the system. This allows us to answer questions such as if the model performs better when the ground truth set is more extensive and if it is worth having multiple levels of relevance.

Besides images and text sections, the dataset should also have query IDs that help us identify how the system performs for each query individually.

3.1.5 Query formulator

The query formulator component is responsible for constructing the queries based on our ground-truth dataset. It retrieves the terms from the text passage that represent it most accurately and then uses them to retrieve relevant information. Our system's workflow to formulate those queries is extensive so we will describe each step separately.

The first step is to apply an analyser to the text sections from the evaluation dataset. We have to use the same analyser from the indexer component (Figure 3.1) because we want to match the words from the text section and the index. Using a different analyser would make some words appear differently from the index, resulting in our system not being able to recognise and match them. This could lead to poor system performance. Hence, we use the same analyser and apply the same techniques, such as stemming and stop-words removal, so that the tokens would match the ones in the index.

After the text has been analysed and stop-words removed, the next step is to find the best terms that correspond to the text section. In IR, we usually refer to these words as Query Terms. To complete this task, we start by finding the most common terms in the paragraph. We create a hash map and store each unique word from the passage as a key and the number of times it

appears in the section as a value. We use a hash map to store term frequencies because it efficiently retrieves values. The complexity of retrieving a value based on a key, in most cases, is O(1).

Because most text sections are long, similar words may appear many times, which means we will have to retrieve the frequency of a specific term many times, making the hash map an ideal structure for this case. However, if efficiency is not an issue, other data structures could also be used to store frequencies.

We continue the query formulation process by using a measure called Term Frequency - Inverse Document Frequency (better known in its abbreviated form TF-IDF). It quantifies the importance or relevance of string representations (e.g., words, phrases) in a document amongst a collection of documents (index in our case). This measure comprises two parts: term frequency, which is the number of times a specific word or term appears in the document, and inverse document frequency, which shows how common the word is among our index[3].

Below is the equation to calculate IDF:

$$IDF(t) = log(\frac{n}{df(t)}), \tag{3.1}$$

where t is a term for which we are calculating inverse document frequency, and df(t) is the data frequency of t (how many times t has appeared in the collection).

We divide the total number of documents in the collection by the number of records in which the term appears. The more common a word is in the collection, the lower the IDF value we get. We also take a logarithm of the result so that the IDF values would not outweigh the TF values, as we will later explain how those values are combined. The reason for dividing the number of all documents by the number of records the term appears in is to get a higher value for rare words and a lower value for general terms. Common terms are usually less descriptive and appear in many documents, making it hard to retrieve relevant images based on those terms. However, unique words that appear only in a few records from the whole collection usually mean that those specific documents accurately represent them.

Once we get our IDF values for each term, we multiply them by their frequencies in the text section that we are evaluating, and we get a new hash map that contains their relevance score (TF-IDF). We take top n terms from that hash map, including the highest relevance values. These query words should accurately represent the text section based on how often they occur and how common they are in the index. Once we have the top query terms in the paragraph, we pass them to the next component in our diagram, which is Retrieval Model (Figure 3.1)

3.1.6 Retrieval model

The next component in the workflow diagram (Figure 3.1) is the retrieval model component. This component is responsible for applying specific rules that dictate how the system will rank the documents retrieved from the index. We have experimented with different retrieval models in our project. Such models are BM25, LMDirichlet, and LMJelinekMercer (Manning et al. 2008):

- BM25 is a bag-of-words model, which means that it ignores words' grammar and order, only keeping their multiplicity. This model ranks a set of records based on the query terms appearing in each document. BM25 has two tuning parameters: k1 and b. k1 is a positive tuning parameter that calibrates the document term frequency scaling. A k1 value of 0 corresponds to a binary model (on term frequency), and a large value corresponds to using raw term frequency. b is another tuning parameter that determines the scaling by document length: b = 1 corresponds to fully scaling the term weight by the document length, whereas b = 0 corresponds to no length normalisation.

[3]More information about TF-IDF can be found at `https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/`

- LMDirichlet model assigns a negative score to the documents that contain the specific term, but the number of times the word appears in the document is less than some threshold decided by the collection language model.
- LMJelinekMercer model involves a linear interpolation of the maximum likelihood model with the collection model, using a coefficient (lambda) to control the influence of each model.

After trying out these models, we realised that the BM25 returns significantly better results than the others, so we decided to use it in our project.

3.1.7 Evaluator

Evaluator is the final component in our workflow diagram (Figure 3.1). It has two incoming data streams. The first one, coming from the Retrieval Model component, is a list of ranked retrieved documents from the index. The second one, coming from the Ground Truth Constructor component, is a list of documents from the ground truth (evaluation) dataset. The evaluator component takes these values and compares them to evaluate how well the system's retrieval process performs.

We use multiple standard IR evaluation metrics (Manning et al. 2008) to evaluate our system. The first one we will discuss is Precision (also known as P@k, where k is the number of retrieved documents we take into account). It estimates the model's performance by measuring the percentage of retrieved documents that belong to the ground truth dataset out of k retrieved documents. In other words, it is the percentage of relevant images retrieved from the index. It is an excellent metric to measure how many retrieved images are applicable. However, it doesn't take into account the order of retrieved documents. Ideally, the most pertinent documents should always appear higher in the recommendations than others.

We introduce an evaluation metric called Mean Average Precision (MAP) to tackle the issue above. It takes into account the order of retrieved documents. Relevant images retrieved at a higher position has a more significant effect on AP value compared to the same pictures retrieved at a lower rank. This way, we can evaluate how good the model is, based on images and the order they have been returned.

Another useful evaluation metric is Recall. It shows the percentage of ground truth images that our system retrieved from the index. However, there are quite a few downsides to this metric as well. For example, if the ground-truth set has only one image, then Recall will always be 0 or 1, which will not be a reasonable estimation of our model. Another downside becomes apparent if the system retrieves fewer images than the size of the ground truth. Even if all the retrieved images are from the ground truth set, it is still impossible to reach the maximum recall value because too few images are retrieved.

An additional metric we will discuss is MRR. It measures the reciprocal rank of the first retrieved relevant image. It is an ideal metric to see how high the retrieved documents' list is the first relevant document; however, it does not consider any other retrieved documents. That is the most significant limitation of this evaluation metric.

The last evaluation metric we will discuss is Normalised Discounted Cumulative Gain (NDCG). It is similar to MAP; however, it considers the relevance of the ground-truth documents. For this metric to be helpful, our ground-truth set must have multiple significance levels, and we should assign each image to one of those levels. Retrieved relevant image from the index will affect the NDCG value more if that image is highly relevant in the ground-truth set.

4 Implementation Details

4.1 Dataset choices

This section discusses different dataset choices that we considered and their applicability to our project.

4.1.1 MS COCO

The first dataset that we considered for our project was MS COCO (Microsoft Common Objects in Context) (Lin et al. 2015). This dataset consists of 330 thousand images, out of which over 200 thousand are labelled. Each labelled image contains 5 different natural language captions. COCO comprises 1.5 million object instances with 80 categories, which includes objects for which individual instances may be easily labelled (e.g., person, car, chair), and also accommodates 91 "stuff" categories, which include materials and objects with no clear boundaries (e.g., sky, street, grass). These features provide significant contextual information, which could be used to train algorithms for object detection, classification, segmentation.

Overall, we are not planning to use object detection, classification, or segmentation in our model, hence making all that contextual data unusable. Furthermore, the size of this dataset is quite small, compared to other similar publicly available datasets, which makes MS COCO even less suitable for our work, as having more data usually means training a better model. Moreover, it would be difficult to evaluate our design using this dataset, as it only provides natural language image captions, but no contextual paragraphs describing the image. Our goal is to build a system that would recommend images based on a text paragraph written by a user, meaning that the dataset we choose should be suited for such evaluation. Based on these drawbacks we decided to look for a better alternative.

4.1.2 COCO-Text V2

The first dataset that we considered for our project was MS COCO (Microsoft Common Objects in Context) (Veit et al. 2016). This dataset consists of 330 thousand images, out of which over 200 thousand are labelled. Each labelled image contains five different natural language captions. COCO comprises 1.5 million object instances with 80 categories, including objects for which individual samples may be easily labelled (e.g., person, car, chair). Also, it accommodates 91 "stuff" categories, which include materials and objects with no clear boundaries (e.g., sky, street, grass). These features provide essential contextual information, which we could use to train object detection, classification, and segmentation algorithms.

Overall, we are not planning to use object detection, classification, or segmentation in our model, making all that contextual data unusable. Furthermore, the size of this dataset is relatively small compared to other similar publicly available datasets, which makes MS COCO even less suitable for our work, as having more data usually means training a better model (Kuznetsova et al. 2020; Kim et al. 2021; Srinivasan et al. 2021). Moreover, it would be difficult to evaluate our design using this dataset, as it only provides natural language image captions but no contextual paragraphs describing the image. Our goal is to build a system that would recommend images

based on a text paragraph written by a user, meaning that the dataset we choose should be suited for such evaluation. Based on these drawbacks we decided to look for a better alternative.

4.1.3 Open Images Dataset V6

A further dataset we investigated is the Open Images Dataset V6 (Kuznetsova et al. 2020). This dataset contains 9M images annotated with image-level labels, object bounding boxes, object segmentation masks, visual relationships, and localized narratives. It has a slightly broader annotation system than the previously described MS COCO dataset. All the images were hand-annotated by professional image annotators. It contains 16M bounding boxes for 600 object classes on 1.9M images, making it the largest existing dataset with object location annotations (Kuznetsova et al. 2020). Professional annotators have manually drawn the boxes to ensure accuracy and consistency. The images are genuinely diverse and often contain complex scenes with several objects (8.3 per image on average). Finally, the dataset is annotated with 59.9M image-level labels spanning 19,957 classes.

Overall, the dataset is superior to the previously discussed MS COCO, as it contains significantly more text-image pairs and has a broader and better-quality annotation system. However, the lack of contextual data and difficulties evaluating this dataset indicate that Open Images Dataset V6 is not optimal for our project.

4.1.4 ImageNet

Another dataset is the ImageNet (Kim et al. 2021). It contains more than 14 million annotated images according to the WordNet hierarchy and over 1 million images with bounding box annotations. A set of test images is also released, with the manual annotations withheld. Annotations fall into one of two categories. One is an image-level annotation of a binary label for the presence or absence of an object class in the image, e.g. "there are cars in this image" but "there are no tigers". The other is an object-level annotation of a tight bounding box and class label around an object instance in the image, e.g. "there is a screwdriver cantered at position (20,25) with the width of 50 pixels and height of 30 pixels.

This dataset is similar to the Open Images Dataset V6 dataset described above. It has a few improvements over that dataset (e.g., image-level and object-level annotations, SIFT feature, the number of subcategories) while also being inferior in specific ways (e.g., annotations quality and depth). This dataset also lacks contextual data and has the same issues in the evaluation as mentioned in other datasets, which makes it unfeasible for our project.

4.1.5 WIT

The final dataset that we considered is WIT: A Wikipedia-Based Image-Text Dataset (Srinivasan et al. 2021). It is the largest publicly available multimodal dataset of image-text pairs, of which we are aware (Srinivasan et al. 2021). It comprises 37.5 million entity-rich image-text examples and 11.5 million unique images across 108 languages. The language coverage of WIT is over ten times bigger than most similar datasets that we examined (Kuznetsova et al. 2020; Lin et al. 2015; Kim et al. 2021).

Unlike typical multimodal datasets constituted of only a single or a few captions per image (Lin et al. 2015), WIT includes a lot of page-level and section-level contextual information. It allows us to index and retrieve images using multiple data modalities, making their retrieval more accurate.

WIT is a challenging and realistic evaluation dataset because of the vast richness of Wikipedia texts and images available, representing real-world entities and attributes. Most state-of-the-art models demonstrated significantly lower performance on WIT vs traditional evaluation sets (Jain et al. 2021). Based on that knowledge, we can assume that if our model performs well on the

WIT dataset, it should also receive good results on other evaluation sets. That is primarily because of how rich the model's contextual data is, allowing it to handle any given topic.

The dataset itself is a collection of 10 files that contain 17 fields of data. A tab character separates each data column. The fields include the Wikipedia page language, its URL, the URL of an image inside the Wikipedia page, the title of the Wikipedia page, the title of the specific section that we are examining, the text section's content, contextual data about an image, and others.

The critical part of the dataset is the contextual data. There are three fields containing information about images: Caption Reference Description, Caption Attribution Description, and Caption Alt Text Description.

Caption Reference Description is the caption visible below the image on the Wikipedia page. It is the least common among the three fields but is the most descriptive and relevant to the picture.

Caption Attribution Description can be found on the Wikimedia page of the image. This text is common among all image instances and thus can be in a different language than the Wikipedia page from which we extracted it. This field is not as informative as the Caption Reference Description but still contains descriptive information about the image.

Caption Alt-Text Description is the "alt" text associated with an image. It is not visible anywhere and is the least informative of all the three fields.

Overall, the WIT dataset suits our project very well. It provides multiple features superior to other similar datasets, including substantial amounts of contextual data, multilingualism, and massive dataset size. It also allows us to conduct evaluations to determine how well the model performs because it provides a ground truth that we can measure using different performance metrics.

4.2 Index

Our project uses Apache Lucene, an open-source search engine library initially written in Java. It provides powerful indexing, searching features, spellchecking, hit highlighting, advanced analysis, and tokenisation capabilities. It is used in many applications, ranging from mobile devices and desktops to Internet-scale solutions.

This library provided many beneficial features when building and querying the index. In Section 3.1.3, we discussed analysing and filtering contextual information when constructing the index and querying it. For this reason, we used an analyser called EnglishAnalyzer, provided by Lucene. It contains multiple filters, previously discussed in Section 3.1.3. Examples of those are StandardTokenizer, LowerCaseFilter, EnglishPossessiveFilter, StopFilter, PorterStemFilter. These filters help us normalise the data and improve the quality of our retrieval.

However, this is not the only analyser Apache Lucene offers. StandardAnalyzer was another choice that we considered. However, it did not include PorterStemFilter, which meant that our system would consider words with the same stem but with different prefixes or suffixes as different terms. It would make the retrieval process a lot less accurate, so we chose EnglishAnalyzer for our project.

In our EnglishAnalyzer, we used a custom list of stop words because the default list by the library does not remove all of them, which reduces the system's performance.

4.3 Parsing Dataset

The original WIT Dataset contained 10 files of .tsv format (tab-separated values). A tab character separates each field from other fields, and a separate line represents each document. The total size

of the dataset was over 60GB.

This format is a very efficient and readable way to represent data compared to similar formats, such as .csv (comma-separated values). That is because it only contains tab characters (that visually look the same as spaces) and no other unnecessary symbols (e.g., commas in .csv format) that would impede the readability. We decided to stick with the same format when parsing the dataset to construct an index or build our evaluation datasets.

One efficiency improvement that we made when building our evaluation datasets was to use document ids instead of image URLs to represent ground-truth images of the dataset. One reason for this is that it saves much space in the evaluation dataset file. Image URLs are usually quite lengthy, while a document ID can represent it with a single number.

Another reason is that it makes evaluation a lot more efficient. Instead of comparing each retrieved image URL from the index to each image URL in the ground-truth set, we compare the document IDs. Comparing two numbers instead of two long strings is considerably more efficient, as we do not need to compare strings character by character.

Since we built our evaluation datasets manually, we only had image URLs and not document IDs. In order to get the document IDs, we had to build a program that searches the index for each image URL and retrieves a corresponding document ID. In the end, it replaces the image URLs with document IDs.

As we mentioned before, the initial dataset size was over 60GB; however, after filtering the data columns and rows and converting image URLs into document IDs, we reduced the size to 1.6GB, which was a significant improvement.

Another vital implementation detail worth mentioning is the contextual information. We have already established (see Section 4.1.5) that each document in the dataset has some contextual information that we use to retrieve images. However, the WIT dataset has three different fields of contextual data. Those fields are Caption Reference Description, Caption Attribution Description, and Caption Alt Text Description (described more thoroughly in Section 4.1.5).

In order to capture the most descriptive contextual information field, we build a simple program that checks if the specific field exists and takes its value, otherwise continues checking other fields in order of relevance. First, we attempt to gather the Caption Reference Description column data because it is the most informative field out of all three. If this step fails, we attempt to gather data from the Caption Attribution Description field, as it usually contains quite descriptive and accurate data. In the worst-case scenario, if both fields are empty, we get the value from the Caption Alt Text Description field. However, it is usually a terrible representation of an image, although it is better than having no contextual information related to an image, in which case it will never be retrieved.

4.4 Building Evaluation Datasets

In this section, we will discuss how we built our evaluation datasets.

The initial evaluation set we built contains 25 text sections from WIT Dataset. For each text section, we are storing three fields: query ID, which is just the unique number from 0 to 24 that represents every query in the dataset, text section content, from which we are trying to retrieve query terms, and the list of image URLs, that is a set of relevant images that we are aiming to retrieve from our indexer.

We chose the text sections in the evaluation set to be relatively lengthy and rich. Choosing rich paragraphs means that we will test the query selector component appropriately, as it is more challenging to select relevant query terms from a larger text section.

We built the ground-truth image set manually. Each query usually contains 5–50 images that appear on the same Wikipedia page as the text section that we are using to query the index. We only take the most relevant images so that they represent the topic extremely accurately.

After building this dataset, we decided that we could make it more efficient. Having multiple image URLs in the evaluation dataset means that it takes a lot more space and requires us to retrieve image URLs from the index and compare them character by character to see if the images match or not. This process is extremely inefficient; hence, we decided to replace the image URLs in the dataset with the document IDs representing those images. It indicates that instead of a list of lengthy image URLs, we only keep a list of document IDs. Also, this means that when we retrieve images from the index, we only need to retrieve the document ID and then compare it with our image ID. Comparing two numbers is a lot faster than comparing long strings; hence the efficiency of our model was greatly increased.

To perform this transformation, we wrote a program that takes the initial evaluation dataset as an input, parses it, queries the index using the image URL field, retrieves the document IDs for each image, and creates a new file where the document IDs replace the image URLs. This newly created evaluation dataset will be used as an input when forming queries and retrieving relevant images from the index.

Another evaluation dataset that we have created contains the same 25 queries, except their ground truth image sets are different. Instead of hand-picking images, we used an approach that extracts all the images from the Wikipedia page in which the section appears, and from all other Wikipedia pages referenced from the initial page. This approach provides a much bigger ground truth and allows us to calculate another performance metric Normalised Discounted Cumulative Gain (better known as NDCG), which allows us to measure the quality of the retrieved documents' ranking, taking into account their relevance. We could not use the NDCG metric with the previous dataset because all the ground truth images were of the same importance, as they were all hand-picked. However, in this dataset, the images come from 3 different levels of relevance.

We expect it to improve the precision of our model, as bigger ground truth means that more retrieved images will be considered relevant to the query. However, it is very likely to reduce the recall because many images in the ground truth will not be of high relevance, so the system is not likely to retrieve them. Another reason is that the ground truth will be massive compared to the previous dataset. Our system will probably retrieve fewer images than the ground truth, meaning that technically we would not even be able to achieve a high recall. In further chapters, we compare the results of these datasets to see if our assumptions were correct.

5 Experiment Setup

This chapter will discuss the setup of the experiments that we have conducted. The goal of our experiments is to answer the research questions that we have established in Section 1.5. We will examine each experiment we had conducted, providing our reasons for it and how we executed it.

5.1 Performing Experiments

We are going to be performing experiments on two different ground truths. As we have described in the Section 4.4, we have constructed two unique evaluation datasets. We assembled the first one manually - chose the specific Wikipedia pages and added relevant images by hand. This ground truth is extremely strict: it only contains x images on average for each text section. It could be hard to accurately evaluate our system on this dataset because of its small ground truth. Even if the system is effective and returns relevant images, those images may not be in the ground truth (even though they would still be relevant to the topic), which would penalise the system harshly.

For that reason, we chose to create our second evaluation dataset, which has a more lenient ground truth. The images for this ground truth were not gathered by hand (as each text section contains y images on average). We had built a program that collects all images from the Wikipedia page, and all the Wikipedia pages referenced from the original one. We added them to the second dataset's ground truth, separating them into four levels of relevance (3 - highest relevance, 0 - completely irrelevant).

Both evaluation datasets contain the same text sections from the same Wikipedia pages. However, the second dataset ground truth is significantly more extensive and comprises multiple levels of relevance. It should be able to represent real life scenario better, as there are usually thousands of relevant images based on user's query, some of which are more relevant than the others.

We perform our experiments on different ground truths for a few reasons. Even though this evaluation will not help us improve the system (because we do not change system parameters - we only change the ground truth on which we evaluate the system), it can give us valuable insights into why specific queries perform better than others. It could also be helpful for other researchers working on similar projects.

One of the reasons is to see if having a bigger ground truth can improve our model results. Considering the evaluation metrics that we use in our project (will be discussed further in Section 5.2), we can assume that using a bigger ground truth will increase the precision (Section 5.2.1) of the model because the system is more likely to retrieve relevant images if there are more of them. We can also predict that the recall (Section 5.2.2) will decrease for the second dataset, as the set of images is more extensive and comprises less relevant images, meaning that the system is less likely to retrieve most of them. We can also speculate that MRR (Section 5.2.3) will be higher because the system is more likely to the first relevant image sooner when the set of images is larger.

However, the part that we are primarily interested in is seeing how MAP (Section 5.2.4) and NDCG (Section 5.2.5) values compare to each other. They are entirely different evaluation

metrics, but their goal is similar: evaluating the retrieved results based on at which rank they were retrieved. We will examine how multiple levels of relevance in the ground truth impact the system's results.

Based on the outcomes of this experiment, we will see which dataset evaluates the system more accurately. We can compare the metrics received from both evaluations and manually check the retrieved images to determine which model performs better and if the evaluation metrics are accurate. These results might help future researchers decide what kind of evaluation dataset to build, or persuade them to use ours. Based on the results, we can also reason why specific queries perform better on one dataset than another.

5.2 System Evaluation

In order to perform experiments, we first need to have a way to evaluate our system's performance so that we can measure the effectiveness of our proposed approach. For that purpose, we use the standard IR evaluation measures. We will discuss each of them in turn.

5.2.1 Precision

Precision is the fraction of the documents retrieved relevant to the user's information need (e.g., if the system retrieves ten documents, out of which five are relevant, precision would be 0.5). However, precision is usually calculated for a specific number of retrieved documents. If we want to calculate precision@5 (also known as p@5), we only consider the first five retrieved documents and calculate how many are relevant to the user.

We can define precision as:

$$precision = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{|\{retrieved\ documents\}|}. \tag{5.1}$$

We take an intersection of relevant and retrieved documents to find the number of retrieved documents that are relevant. Then, we divide that number by the total number of retrieved documents to get the final fraction. If we want to incorporate the variable k into the equation (in order to calculate p@k, which is precision that takes into account only the first k retrieved documents), the number of retrieved documents will always be k, and we will only consider relevant documents from the first k retrievals.

To incorporate precision into our model, we need to have a list of retrieved documents and a list of relevant documents for each query. Referring back to the workflow diagram (Figure 3.1), we can see that the output of the *Retrieval Model* component is a ranked list of document IDs, and the output of the *Ground Truth Constructor* component is a list of ground-truth document IDs (the IDs of relevant documents). From the diagram, we know that we have both retrieved and relevant documents, meaning that we can easily calculate the precision of our system.

To evaluate our model's precision, we calculate p@5 for every query in the evaluation dataset and the average precision for all the queries. The code for our algorithm can be seen below:

```
public float precisionAtK(int k, ArrayList<String> retrievedIds,
    ArrayList<String> groundTruthIds) {
   Set<String> groundTruthIdsSet = new HashSet<>(groundTruthIds);
   int matched = 0;
   for (int i = 0; i < k; i++) {
      String retrievedId = retrievedIds.get(i);
      if (groundTruthIdsSet.contains(retrievedId))
```

```
            matched++;
        }
        float precision = (float) matched / k;
        return precision;
    }
```

In this code example, we iterate over the first k retrieved document IDs and check if our evaluation set contains them. For each retrieved ID that is relevant, we increment *matched* value, and divide it by the k value after the loop finishes execution.

While precision is a good evaluation measurement, as it shows the fraction of retrieved documents relevant to the information's need, it ignores the order in which the values were retrieved. For example, assume that our ground-truth set contains IDs $\{3, 5, 8\}$. Then our model would give the same precision score for both $\{1, 2, 4, 6, 3, 5, 8\}$ and $\{3, 5, 8, 1, 2, 4, 6\}$. Although, we can see that the second set is significantly better than the first one, as it retrieves the relevant documents first. This, in fact, is the major limitation of set-based precision, and we will discuss other metrics that can avoid this issue.

5.2.2 Recall

Recall is the fraction of the documents relevant to the query that are successfully retrieved (e.g., if the ground truth contains ten documents, and the system retrieves two of them, recall is 0.2).

We can define recall as:

$$recall = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{|\{relevant\ documents\}|}. \tag{5.2}$$

We take an intersection of relevant and retrieved documents to find the number of relevant retrieved documents. Then, we divide that number by the total number of relevant documents to get the final fraction. This equation is very similar to precision; the only difference is the denominator: it contains the number of retrieved documents in the precision equation, whereas the recall equation contains the number of relevant documents.

To incorporate recall into our model, we use the same technique used for precision calculation by taking the output of *Retrieval Model* and *Ground Truth Constructor* components (Figure 3.1) and calculating recall.

We calculate recall for every query in the evaluation dataset and then find the average for all the queries to evaluate our model. The code for our implemented algorithm can be seen below:

```
public float recall(ArrayList<String> retrievedIds, ArrayList<String>
     groundTruthIds) {
    Set<String> retrievedIdsSet = new HashSet<>(retrievedIds);
    int matched = 0;
    for (String groundTruthId : groundTruthIds) {
        if (retrievedIdsSet.contains(groundTruthId))
            matched++;
    }
    float recall = (float) matched / groundTruthIds.size();
    return recall;
}
```

In this code example, we iterate over the ground-truth document IDs and calculate how many of them our system has retrieved from the index. For each relevant document retrieved, we increment *matched* value. After we compare all the values, we divide *matched* by the total number of document IDs in the evaluation dataset.

Recall is a useful evaluation measure that tells us the fraction of the ground truth that our system has retrieved. However, it has a few significant flaws. One of them is that if the number of retrieved documents is lower than the documents in the ground truth, it is impossible to achieve a maximum recall value. For example, if the ground truth contains IDs $\{1, 5, 7, 9, 3\}$, and our retrieved IDs are $\{1, 5\}$, we know that the model is extremely accurate because it retrieves the most relevant documents. However, since it only retrieves two documents, recall is 0.4 (two out of five), which is quite low compared to the model's performance.

The same issue occurs if the number of documents to retrieve is relatively large. If the ground truth contains five documents, but we are retrieving a hundred of them, the high chances are that those five documents will be among those hundred, meaning that recall will be 1. However, it does not mean that the model is performing effectively. That is why we cannot trust a single evaluation metric and need to use at least a few.

A further issue appears if our ground truth contains documents of different levels of relevance. Recall cannot distinguish the difference between relevance in the ground truth. For example, if our evaluation set contains two levels of relevance: the most relevant IDs are $\{1, 5, 7\}$, and the rest are $\{2, 6, 8\}$. If our system retrieves either $\{1, 5, 7\}$ or $\{2, 6, 8\}$, the recall value will be the same (0.5). However, the former model performs significantly better than the latter because it retrieves more relevant images, but recall does not capture this information.

5.2.3 MRR

The reciprocal rank (RR) information retrieval measure calculates the reciprocal of the rank at which the first relevant document was retrieved (e.g., RR is 1 if a relevant document was retrieved at rank 1, 0.5 if a relevant document was retrieved at rank 2). The measure is called the Mean Reciprocal Rank (MRR) when averaged across a sample of queries (Craswell 2009).

We can define MRR as:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i}, \tag{5.3}$$

where Q is the sample of queries, and rank is the rank of the first relevant document retrieved for a particular query.

We find at which rank the first relevant document is retrieved for each query in Q, get its reciprocal value, and average it out for all the queries.

The code for our implemented algorithm calculating RR can be seen below:

```
public float MRR(ArrayList<String> retrievedIds, ArrayList<String>
    groundTruthIds) {
   int rank = 1;
   float mrr = 0;
   Set<String> groundTruthSet = new HashSet<>(groundTruthIds);
   for (String retrievedImage : retrievedIds) {
      if (groundTruthSet.contains(retrievedImage)) {
         mrr = 1 / (float) rank;
         break;
```

```
        }
        rank++;
    }
    return mrr;
}
```

In this code example, we iterate over the IDs of retrieved documents, find at which rank the first relevant image is retrieved, and get its reciprocal rank (dividing one by the rank).

MRR is a useful measure that tells us how high among the retrieved documents is the first relevant record. However, it only takes into account the first relevant document retrieved. It is ideal for systems that contain a ground truth with a single document, whereas in our case, each document contains many images, which makes this metric less practical for us. If the ground truth contains IDs {1, 3, 6, 7, 9}, then this metric would give the same score for both {2, 1, 3, 6, 7, 9} and {2, 1, 4, 5, 8} retrieved IDs. Although, we can see that the first retrieval model is significantly better than the second.

5.2.4 MAP

Mean Average Precision (MAP) provides a quantification of both the effectiveness aspects in IR, namely that of precision and recall (Manning et al. 2008). The average precision (AP) approximates the area under the uninterpolated precision-recall curve for a single information need. So the MAP is roughly the average area under the precision-recall curve for a set of queries. This metric is similar to precision, but it also considers the order in which the documents are retrieved. The AP value for a single query is reduced for each document retrieved at an unsatisfactory rank. If the ground-truth set contains IDs {1, 5, 7} and our model retrieves {2, 4, 1, 5, 7}, it will gain a lower AP value than the model retrieving {1, 5, 7, 2, 4}.

We can define MAP as:

$$MAP = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{GTP} \sum_{k=1}^{n} p@k, \tag{5.4}$$

where Q is the sample of queries, GTP is the number of ground-truth positive values (how many documents retrieved are relevant), n is the total number of retrieved documents, and p@k is the precision at k.

To explain the equation in words, we iterate over each retrieved document from the system, calculate its p@k value, decide if the retrieved document is in the ground truth. If it is, we add this value to the sum of p@k values; otherwise, we do nothing for that particular value. Once we go over all the retrieved documents for a specific query, we take the average of the sum of p@k values (by dividing the sum by the number of ground-truth positive results). It gives us back the AP value for a specific query. If we repeat this process for all the queries in the sample Q and take the mean of the results, we get the MAP value for the model.

The code for our implemented algorithm calculating AP can be seen below:

```
public float AP(ArrayList<String> retrievedIds, ArrayList<String> groundTruthIds)
    {
    int count = 1;
    int rel = 0;
    float sum = 0;
    Set<String> groundTruthSet = new HashSet<>(groundTruthIds);
```

```
    for (String retrievedId: retrievedIds) {
        if (groundTruthSet.contains(retrievedId)) {
            rel++;
            sum += (float) rel / count;
        }
        count++;
    }
    if (rel == 0) {
        return rel;
    }
    return sum / rel;
}
```

In this code example, we iterate over the retrieved IDs and check if the ID is relevant. If it is, then we increment the *rel* value, which is the number of retrieved relevant documents. We also increment the sum by adding *rel* value divided by *count* (rank of the retrieved document), which is precision@k. After we go over all the retrieved documents, we either return 0 if there were no relevant documents or return the average of p@k values (divide the summed p@k values by the number of relevant retrieved documents).

MAP is an excellent metric to measure our model's performance. It gives a single value, making it easy to compare with other models or modifications of our model. One disadvantage of this metric that affects our model is that it only works with a ground truth that contains a single level of relevance (meaning that all the images in our evaluation dataset are of the same importance). However, if there are multiple levels of relevance, this measure cannot distinguish them, making it less suitable in those cases. Since we have two evaluation datasets (read more in Section 4.4, where one consists of one level of relevance, and the second consists of three levels of relevance, we can use this metric for the first one. However, we would need to find a different measure that we could use for our multi-relevance evaluation dataset.

5.2.5 NDCG

The final evaluation measure that we will discuss is the Normalised Discounted Cumulative Gain (NDCG). It is designed for situations of non-binary notions of relevance when the ground truth contains more than one level of relevance (Manning et al. 2008). NDCG is a popular method to measure the quality of search results. The "cumulative" part means that results of high relevance have more weight to the NDCG value than results of lower relevance. The "discounting" part means that relevant results influence NDCG value more if retrieved at higher ranks. The "normalisation" part means that the ranking results should be irrelevant to the query performed. We will use this metric to evaluate our second dataset, which contains four levels of relevance (high relevance, average relevance, low relevance, non-relevance).

Cumulative Gain (CG) is the sum of the graded relevance values of all documents retrieved from the index. It is the first step of the algorithm; however, it does not consider the rank at which the system retrieves the documents.

We can define Cumulative Gain (CG) as:

$$CG = \sum_{i=1}^{n} rel_i, \tag{5.5}$$

where n is the number of retrieved documents from the index, and *rel* is the relevance of a specific document retrieved.

Discounted Cumulative Gain (DCG) solves the previous measure's (CG) problem by considering the rank at which the documents are retrieved. It discounts the values appearing low in the retrieved list. Documents of high relevance that are retrieved low in the list are highly penalised as the graded relevance value is reduced logarithmically proportional to the position of the result.

We can define Discounted Cumulative Gain (DCG) as:

$$DCG = \sum_{i=1}^{n} \frac{rel_i}{ln(i+1)}, \tag{5.6}$$

where n is the number of retrieved documents from the index, and rel is the relevance of a specific document retrieved.

The final step is to apply the normalisation to the values. This part is essential because our model may perform significantly worse for some queries than others. The DCG scores for more complex queries might significantly decrease the average score, making the evaluation less accurate. NDCG solves this problem by scaling the values by the iDCG, which is the best DCG result seen. For this reason, we need to first calculate DCG values for all the queries, and only then we can calculate NDCG.

We can define Normalised Discounted Cumulative Gain (NDCG) as:

$$NDCG_i = \frac{DCG_i}{iDCG}, \tag{5.7}$$

where DCG is the Discounted Cumulative Gain for a query i, and iDCG is the highest DCG value from all the queries.

5.3 Experiments

Now that we established the metrics we use to evaluate our model, we can describe the actual experiments we had conducted to improve our model's performance with respect to the research questions described in Section 1.5.

5.3.1 BM25 Model Parameters

One experiment that we have conducted was evaluating BM25 retrieval model parameters. As we previously discussed in Section 3.1.6, BM25 has two tuning parameters *k1* and *b*. *k1* is responsible for calibrating the document term frequency scaling, meaning that a high value will indicate a close to raw term frequency. In contrast, a low value will correspond to binary representation, meaning that it will measure only if a term is in the document or not, but not its actual frequency. Parameter *b* determines the term weights' scaling by document length. A high value means that the term weights will be fully scaled (meaning that the same term in a small document will have a bigger weight than in a massive document), while a low value corresponds to minimal length normalisation.

The goal of this experiment was to measure if, given some information need, we can retrieve different modality content more effectively by varying the parameters of our retrieval model BM25 (based on RQ-2 from Section 1.5). When performing this experiment, we already have an information need (extracted query words from the text section), and the only thing we need to measure is the performance of our chosen retrieval model.

To conduct this experiment, we created a program that repeatedly calls our retrieval function with the same queries, just different parameter values. We varied our *k1* value from 0.2 to 1 with

the interval of 0.2, and our b value from 0 to 1 with the same interval. After comparing each $k1$ with each b values, we drew a graph to represent our results. Results will be presented in Chapter 6.

5.3.2 Percentage of Query Words

Another experiment we have conducted was related to the number of query words, which is the number of terms from the paragraph that we consider when creating queries. This step is critical because it directly corresponds to the user's information need. Too few query words could mean that we ignore some important words from the text section. Including those words in the query could significantly improve our model's performance. However, considering a huge number of words could make the query too broad, making our model unable to retrieve documents closely related to the user's information need because there are too many documents representing that query.

Our goal is to create such queries that would most accurately represent the paragraph entered by the user, or a Wikipedia text section. Suppose this process step fails, and the generated queries are irrelevant to the section. In that case, the documents we retrieve from the index will be irrelevant to the user's information need, even if the retrieval process is flawless.

This experiment corresponds to the RQ-1 from Section 1.5, which asks how effectively can we extract the information need from the context. The context, in our case, is the text entered by the user, and we are trying to extract the information need from it by formulating representative queries.

Our initial experiment was to vary the number of query words from 5 to 40, with an interval of 5. However, that experiment did not consider the length of the text section for which we were formulating the queries. For example, if the passage contains only ten words, a query consisting of five words would be able to represent the query accurately. However, if the paragraph consists of a thousand words, it would be impossible to represent it with only five words.

For this reason, we decided to change the experiment's setup to avoid these issues. Instead of taking a fixed number of words from each section (from 5 to 40), we considered a percentage of terms based on the section length. This way, we were able to evaluate each text paragraph more accurately without worrying about its length. Our initial idea was to vary the number of words from 5% to 25% of the paragraph length. However, after realising that 5% almost always returns the best results, we decided to examine it further by evaluating the model for values from 1% to 5%. Results will be presented in Chapter 6.

5.3.3 Relevance Feedback and Query Expansion

This experiment evaluates the effect of applying *relevance feedback* and *query expansion* to our model (introduced in Section 1.2). These techniques are essential for information retrieval because of the problem known as *synonymy*. It occurs when a user is looking for documents related to a specific word (e.g., plane), but most documents satisfying his information need contain the word's synonym (e.g., aircraft). Without applying these techniques to the user's query, the system would not know that the words *plane* and *aircraft* are synonyms, hence would not retrieve corresponding documents.

We can adopt two groups of techniques to tackle this issue: global methods and local methods. Global methods expand or reformulate the query independent of its results (e.g., *query expansion*). Local methods refine the query based on the initial results retrieved from the system (e.g., *relevance feedback*).

Query expansion is a global method that most commonly uses global analysis to expand or generate alternative queries. Thesaurus can be used to expand each query by adding words' synonyms to

it. After applying *query expansion*, the model should be able to overcome the synonymy problem.

The idea behind *relevance feedback* is to include the user in the IR process. After the initial retrieval, the user has to give feedback on the documents' relevance by marking them as relevant or non-relevant. The system returns a refined list of results, and the process can repeat multiple times. This method is effective because it is hard to match user's information need without knowing the collection well. However, it is a lot easier to judge by the documents to see what the user's information need is.

However, asking for explicit user feedback is not optimal because most users would not want to spend time on that. Especially in our project, the goal is to make users save time by recommending relevant images to illustrate their work instead of searching for images on the internet. For this reason, we introduce *pseudo relevance feedback*, also known as *blind relevance feedback*. This method provides an automatic analysis that improves the retrieval performance without extended user interaction. It assumes that the top k retrieved documents are relevant to the query; hence it retrieves other popular terms from those documents to expand the initial query.

In our experiment, we test both *query expansion* and *relevance feedback* in order to find the ideal configuration so that our system retrieves the most relevant images to the user. This experiment corresponds to the first research question (see Section 1.5), which asks how we can effectively extract the information need from the context. In our case, we do not extract the information need from the context, but we refine the already extracted information need so that it matches user's intentions more accurately.

6 Results

In this chapter, we will present the results of the experiments that we conducted, discuss and reason about them, relate them to the research questions (see Section 1.5).

6.1 System Evaluation

We will start this chapter by providing evaluation results for our system. As discussed in the previous chapter (see Section 5.1), we are going to be evaluating two different ground truths to see which one evaluates the system more accurately.

6.1.1 Strict Ground Truth

Strict ground truth is the evaluation dataset that was constructed manually - each image was found and added to it by hand. Consequently, the set of images in the evaluation dataset is relatively small (29 images on average). Having such a small ground truth may negatively affect our evaluation results, as it will be challenging for the system to retrieve exactly those 29 images out of our vast collection.

We evaluated our model using the strict ground truth and evaluation metrics (see Section 6.1), the results are visible in the Table 6.1. These are the specific configurations we used to conduct this evaluation:

- **The Number of Query Words**: 10;
- **The Number of Retrieved Documents for Each Query**: 100;
- **k1 Parameter Value for BM25 Similarity**: 1.2;
- **b Parameter Value for BM25 Similarity**: 0.75;
- **Relevance Feedback**: disabled;
- **Query Expansion**: disabled.

To understand the values from Table 6.1, we are going to analyse each of them separately.

The value for *p@5* is 0.15, meaning that, out of the first five retrieved images, on average, almost one image is relevant. Usually, the top retrieved results are the most important because the user sees them first. One could think that is 0.15 is a low value for this metric, and our model is performing poorly. However, this is not true because the other four images are not necessarily irrelevant to the query. They might still be relevant, just not included in the ground truth. This reasoning makes sense in our case because we are evaluating our model on the strict ground truth, which contains a limited image collection in its ground truth. Overall, 0.15 is a decent value, and we will compare and try to improve it in further experiments.

Similarly, *recall* is 0.21, meaning that around 20% of the ground truth is usually retrieved. Considering how extensive the collection is, retrieving 20% of the images in the ground truth is impressive. However, we need to consider that the system retrieves 100 documents for each query. The ground truth itself is 29 images on average, meaning that we retrieve more than three times more images than the size of the ground truth. Furthermore, knowing that we retrieved

Table 6.1: *The results after evaluating our system using four main evaluation metrics and default parameters.*

Evaluation Metric	Value
P@5	0.15
Recall	0.21
MRR	0.28
MAP	0.16

20% of the ground-truth images does not tell us anything about their position in the retrieved documents list, so it is impossible to evaluate our model based only on this metric. Other metrics will give us more insight into the order of retrieved documents.

MRR value is 0.28. We can interpret this result in words, saying that the first relevant image appears in the position between 3 and 4 (on average). Even though it does not mean that images retrieved in higher positions are irrelevant, it allows us to be sure that at least one relevant image usually appears at rank 3 or 4. We think that this is a satisfactory score, considering that the user will usually see at least one highly relevant image in the top predictions, which he can include in his work.

Lastly, the value of *MAP* is 0.16. It is difficult to interpret the *MAP* value in words because it is a combination of *precision* and *recall* (to be precise, it is an area under the *precision - recall* curve). However, 0.16 seems like a satisfactory score and after we evaluate our model more in-depth we can be sure if this value corresponds to accurate results.

6.1.2 Single Query Strict Evaluation

This section will evaluate a single query to make sense of the results retrieved and relate them to the evaluation metrics (see Table 6.1). We will choose one document from the evaluation set and closely examine it.

The document we will analyse is from the **Stairs** Wikipedia page[1]. It is a query consisting of 65 words and containing 27 images in its ground truth. The query specifications are fairly average (except for the four times smaller section length than an average one). However, we chose this record because the topic is very generic (there are numerous images of stairs in the collection), making our system's retrieval of ground-truth images based on this query very challenging.

The text section that we analyse is below:

A stairway, staircase, stairwell, flight of stairs, or simply stairs, is a construction designed to bridge a large vertical distance by dividing it into smaller vertical distances, called steps. Stairs may be straight, round, or may consist of two or more straight pieces connected at angles. Special types of stairs include escalators and ladders. Some alternatives to stairs are elevators, stairlifts and inclined moving walkways.

Moreover, the main ground-truth image for this paragraph is visible in Figure 6.1.

The first task of our system is to retrieve 10 query words representing the passage. By evaluating this step, we attempt to answer the RQ-1 (See 1.5). These top queries are {stair, straight, stairlift, vertic, distanc, stairwel, escal, stairwai, inclin, ladder}. Most of them are incomplete words because they were stemmed using EnglishAnalyzer and stemming filter (see Section 3.1.3). As we can see, most of these words accurately represent the paragraph. They contain different stairs, their properties, or similar objects (e.g., a ladder). Consequently, we can assume that the *Query Formulator* component (See Section 3.1.5) operates effectively, as it retrieves words that accurately represent the passage.

[1] `https://en.wikipedia.org/wiki/Stairs`

Figure 6.1: *The main ground-truth image corresponding to the section above.*

The next step is retrieving the relevant images from the system. The top 9 images acquired are shown in Figure 6.2. The images show that most of them accurately represent stairs in some shape or form. There are stairs outside, inside, circular stairs, elevators. However, we can also see that they are getting less accurate towards the end of the set. Instead of stairs, we see a ladder in the second image of the bottom row and an illustration in the last image on the bottom row.

Based on this information, it is fair to assume that the images get less representative as they are retrieved at a lower position. Therefore, we extended our experiment by examining the last nine retrieved images. The results are shown in Figure 6.3. After looking at the images, we can say that the images are less representative than the first nine images. However, most of them are still relevant. We can see that most of them represent stairs, except for the first image on the top and bottom rows. Overall, we can assume that the images get slightly less fitting towards the end of the retrieved set, and unexpected images appear occasionally. However, most of them remain relevant to the topic.

To be sure that these images correspond to the ground truth, we demonstrate what some of the ground truth images to this query look like (see Figure 6.4). Even though these pictures represent the topic accurately and do not contain any irrelevant images, they are not much different from the ones retrieved by our model. Because of that, we can assume that our model performs reasonably well.

After manually analysing the retrieved images, we can finally compare our observations with the results generated by the evaluation metrics. The results are shown in Table 6.2. Overall, the values are significantly lower than the average for all queries (see Table 6.1). We will analyse each of the results separately.

The value of $p@5$ being 0.00 implies that the system did not retrieve relevant images from the ground truth in the first five images. Although, after examining the retrieved images, we saw that most of them were relevant, just not included in the ground truth. This problem occurs because the number of images representing stairs in the collection is enormous, making it very problematic for the system to retrieve images from the ground truth, especially in the top five ranks.

The value for *recall* was 0.04, meaning that only 4% of the ground-truth images were retrieved in the process. This value is extremely low, especially considering that the system retrieved more than three times more images than the size of the ground truth. Like $p@5$, we assume that it occurs because the topic is highly generic and the collection contains a vast number of similar images.

Figure 6.2: *Top nine images retrieved by the system, based on the text section described above.*

Figure 6.3: *Last nine images retrieved by the system, based on the text section described above. The top left is the ninth to last image retrieved, while the bottom right is the last retrieved image.*

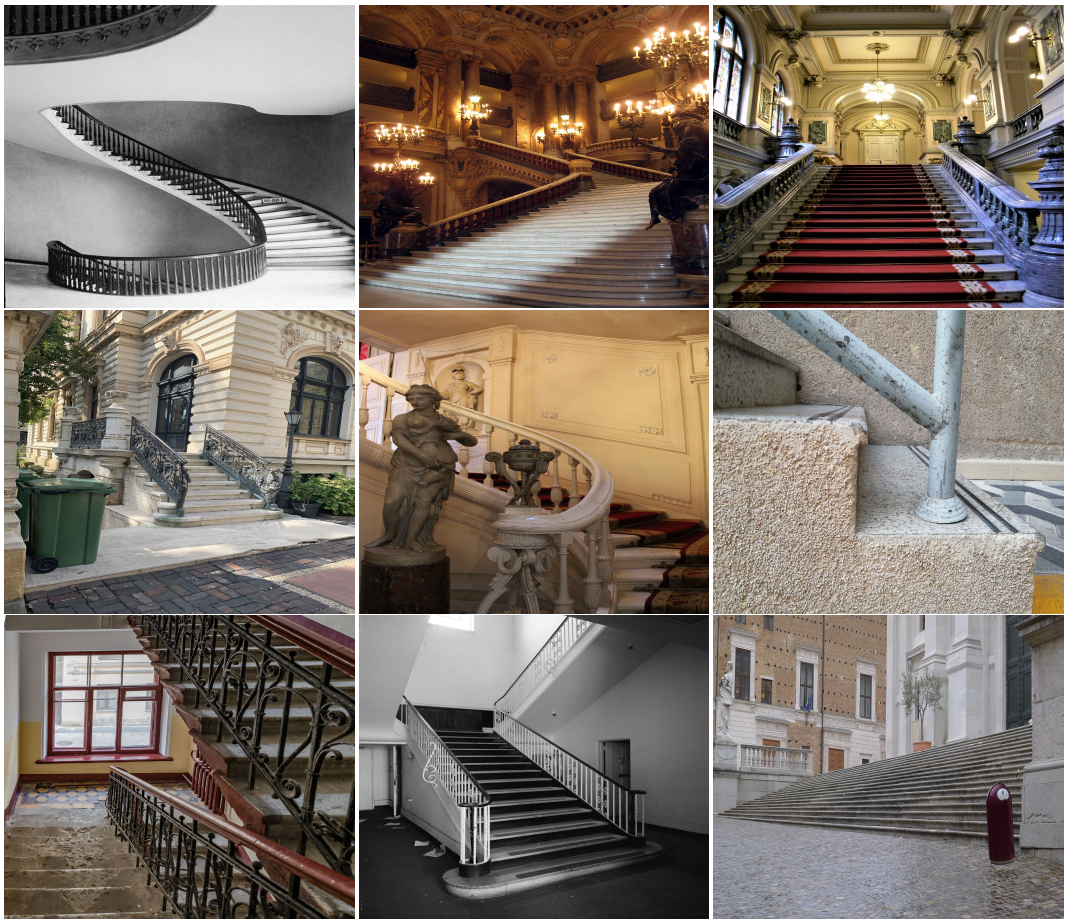

Figure 6.4: *Example images from the ground truth.*

Table 6.2: *The results after evaluating our system based on a single Wikipedia text section using four main evaluation metrics and default parameters.*

Evaluation Metric	Value
P@5	0.00
Recall	0.04
MRR	0.02
MAP	0.02

Table 6.3: *The results after evaluating our system using five main evaluation metrics and default parameters.*

Evaluation Metric	Value
P@5	0.30
Recall	0.00
MRR	0.58
MAP	0.27
NDCG	0.12

MRR value was also significantly lower than the average value (see Table 6.1), only 0.02, meaning that the system retrieved the first ground-truth image at rank 50 out of 100 total retrieved images.

The last metric *MAP* was also very low - 0.02. Compared to the average values (see Table 6.1), the MAP value for this specific query was eight times smaller than the average.

Comparing metric evaluation results with previously observed remarks, we establish that our system was evaluated inaccurately based on this query. As we have witnessed before, the query construction and image retrieval components performed extraordinarily, and most retrieved images were highly relevant. It is incredibly complicated to build a fair ground truth for a general topic such as **Stairs**; hence it is inevitable that the system will not be evaluated equitably.

However, if we find a way to extend the ground truth, then our evaluation metrics would show results closer to the actual effectiveness of the system. That is going to be our next experiment.

6.1.3 Lenient Ground Truth

Lenient ground truth is the second evaluation dataset that we have constructed. It contains the same queries as the *strict ground truth*, but the set of images it contains is much larger. We have built it by using our written program that gathers images from the Wikipedia page and all other Wikipedia pages referenced from the original one (see Section 4.4). Compared to our first evaluation dataset (see Section 6.1.1), which contains on average 29 images in its ground truth, this dataset contains over 11k images, spread across three levels of relevance. It significantly increases the size of the ground truth, and we hope it will help us evaluate our system more accurately than the *strict ground truth*.

We evaluated our model using *lenient ground truth* and the same measures as in Section 6.1.1. However, we included another evaluation metric, known as NDCG, which allows us to evaluate our model based on multiple levels of relevance (more information about NDCG in Section 5.2.5). To conduct this experiment, we used the identical configuration as in Section 6.1.1. Results are shown in Table 6.3.

We can see that our previously made assumptions were correct based on the results. Comparing these values with the ones in Table 6.1, we can see that the *p@5* value doubled (from 0.15 to 0.30). In other words, between one and two relevant images from ground truth are usually retrieved among the top 5 results, compared to less than one in the previous evaluation.

Table 6.4: *The results after evaluating our system based on a single Wikipedia text section using five main evaluation metrics and default parameters.*

Evaluation Metric	Value
P@5	0.00
Recall	0.00
MRR	0.10
MAP	0.10
NDCG	0.01

Furthermore, we can see that *recall* value is 0.00. That is the case because it is very small and gets rounded up to 0. However, we expected it to be very small because the ground truth was enormous, and we only retrieved 100 images.

Regarding *MRR*, we can see that its value 0.58 is more than double what it used to be (0.28) in the previous evaluation. It indicates that the first relevant image appears roughly in the second position, which is extremely high.

Comparing the last common evaluation metric *MAP*, we observe that the value increased from 0.16 to 0.27. It implies that our model was evaluated as more effective using *lenient ground truth* than the *strict ground truth*.

The last evaluation metric that we have measured was *NDCG*. We did not compute it in the previous evaluation because *strict ground truth* did not contain multi-relevance images. The value we obtained in this evaluation was 0.12. It implies that over 10% of retrieved documents were retrieved in the correct order, indicating that the value is satisfactory.

6.1.4 Single Query Lenient Evaluation

In this section, we will compare how a single query that we evaluated in Section 6.1.2 corresponds to the evaluation using the *strict ground truth*. The results can be seen in Table 6.4.

Comparing the two tables (Table 6.2 and Table 6.4), we observe that the values improved significantly. Although they are still not as high as the average values for all the queries, some of them (e.g., *MRR* and *MAP*) increased five times. *NDCG* value is quite low, indicating that only around 1% of retrieved documents were ordered optimally. However, this is understandable, as this query remains extremely challenging to evaluate. *p@5* and *recall* values did not change much (based on Table 6.2 and Table 6.4), although we can suspect that *recall* dropped because of the increased size of the ground truth. Low *p@5* and *recall* values are not surprising because it is hard to get a high *p@5* value for such a generic query, and recall will always be extremely low on such enormous ground truth.

6.1.5 Percentage of Query Words

Another experiment we have conducted involved varying the number of query words (described in Section). This closely corresponds to the RQ-1 (see Section 1.5). For this experiment, we used the *strict ground truth*, as well as default configuration, as mentioned in Section 6.1.1. Our goal is to extract the information need from the Wikipedia text section as effectively as possible; hence finding the optimal percentage of query words might help us achieve that goal. The results can be seen in Figure 6.5.

After inspecting the graphs, we observed that the results are the highest when the number of query words corresponds to 5% of text section length. In order to evaluate this further, we conducted another experiment, evaluating the values from 1% to 5% number of query words used. The results are visible in Figure 6.6.

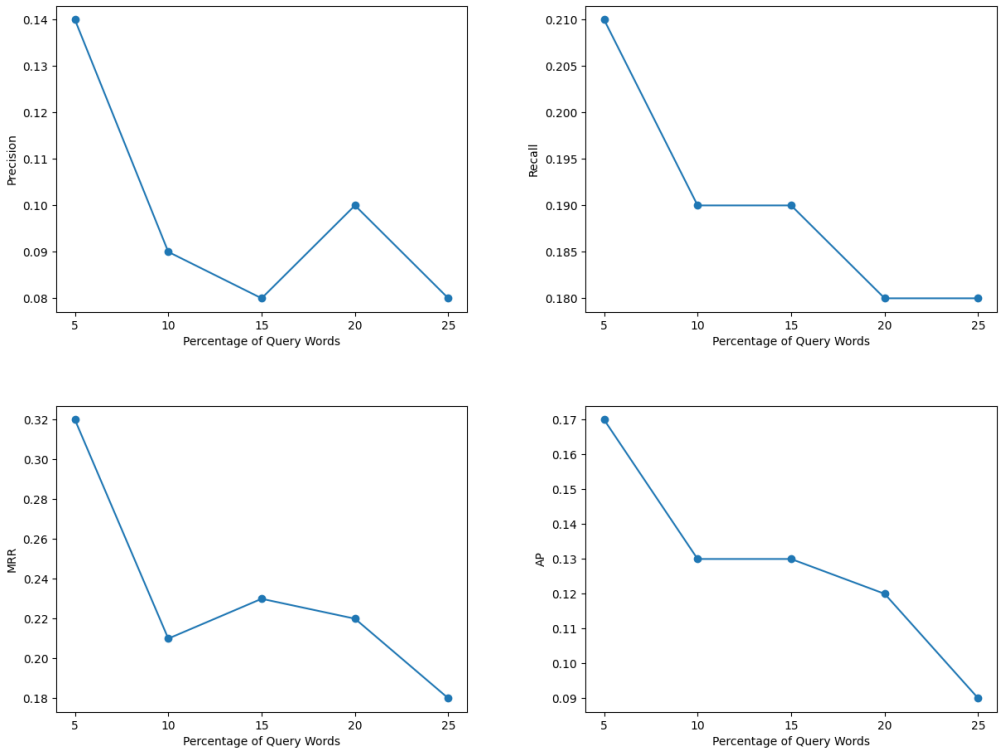

Figure 6.5: *Graphs displaying evaluation metrics (Precision, Recall, MRR and AP) values based on Percentage of Query words (from 5% to 25% of text section length)*

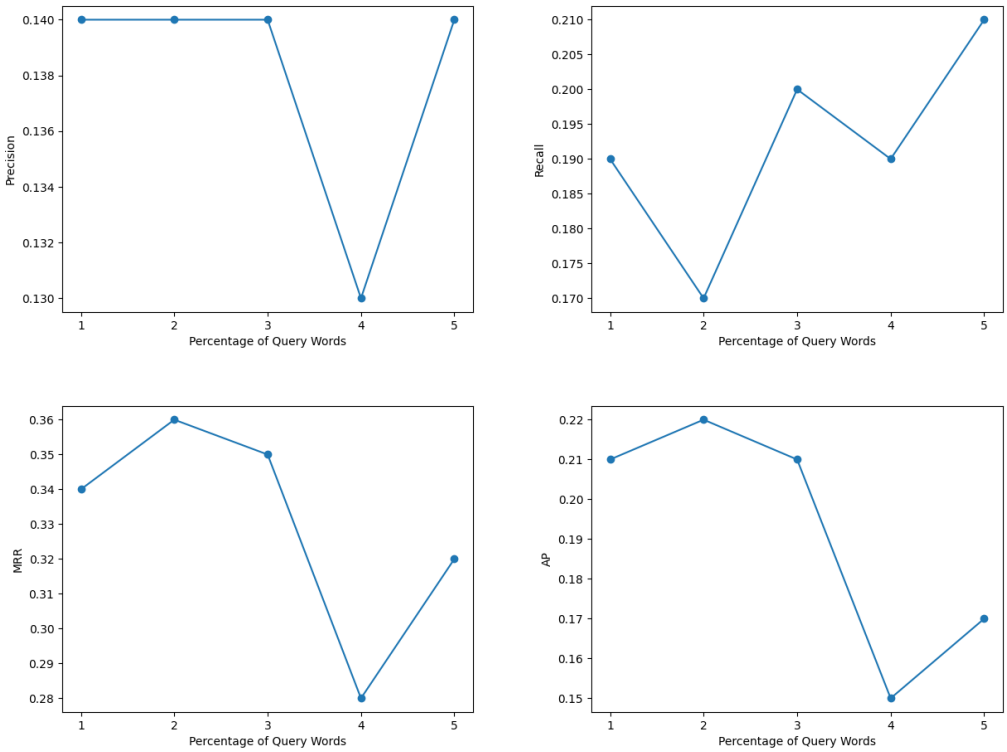

Figure 6.6: *Graphs displaying evaluation metrics (Precision, Recall, MRR and AP) values based on Percentage of Query words (from 1% to 5% of text section length)*

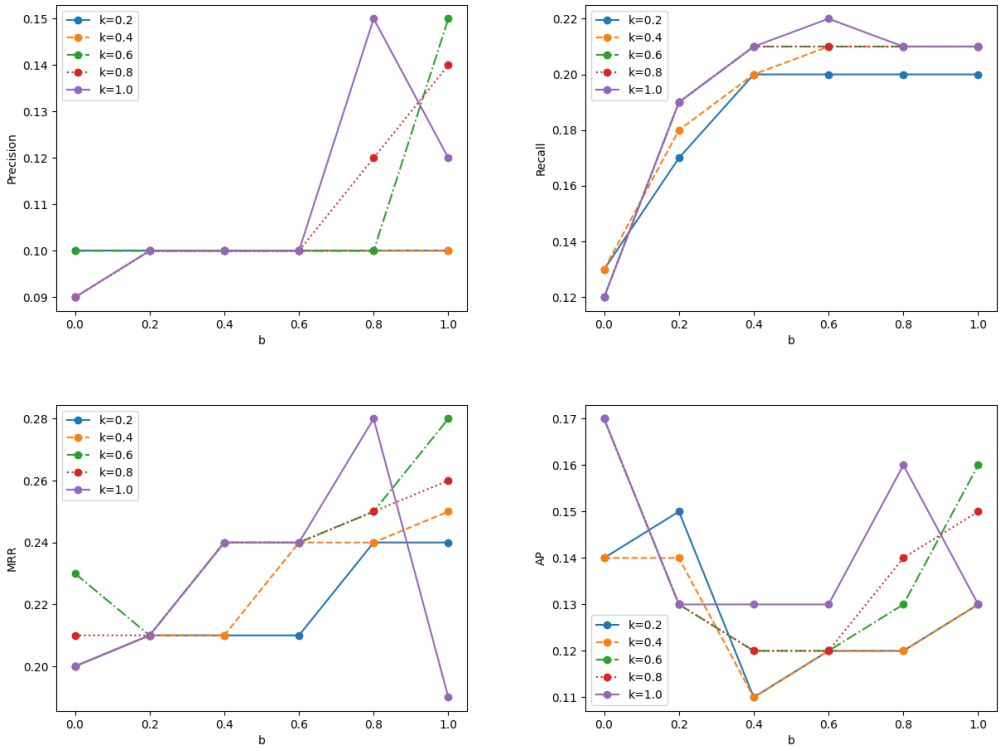

Figure 6.7: *Graphs displaying how Precision, Recall, MRR, and AP values vary based on BM25 similarity parameters k and b.*

The data from the graphs shows that we obtain the highest values for the evaluation metrics at 2% of query words (except for *recall*). That occurs most likely because considering too many words will make the queries too broad, causing us to retrieve images corresponding to those broad queries. Hence, to achieve the most satisfactory results, we should use the percentage of query words between 1% and 3%.

Compared to the first experiment we have conducted (see Section 6.1.1 and Table 6.1), the highest values we get from the current experiment are slightly higher than the results from the previous one (e.g., the highest *MRR* value we achieved in the current experiment was 0.36, while in the previous experiment it was 0.28). Hence, it is clear that using the number of query words as a percentage of the section length is a useful enhancement to the program, and we should aim to use the value between 1% to 3% to achieve the most satisfactory results.

6.1.6 BM25 Parameters

A further experiment we have conducted was varying the parameters of our retrieval model BM25 (read more about BM25 in Section 3.1.6). This experiment corresponds to the RQ-2 (see Section 1.5), as our goal is to retrieve the relevant content from the index effectively, and varying BM25 similarity parameters might help us achieve that. For this experiment, like the previous one, we used the *strict ground truth*, as well as the default configuration, as mentioned in Section 6.1.1. The results are shown in Figure 6.7.

Looking at the graphs, we can observe that the curve corresponding to the k value being 1 achieves the highest results. k value equal to 1 corresponds to using the raw frequency of terms.

Using this value makes sense because our goal is to find the documents that represent our query most accurately, and the documents containing the most words from the query will usually be the most relevant ones.

The b value is usually the highest around the 0.6 - 0.8 mark, meaning that we want to scale the term weights by the document length, but not completely. One reason could be that the query terms in very small documents would have too much weight, indicating that small documents would usually get chosen over big ones. However, small documents may not always accurately represent the query, as they may not have enough information about the specific topic.

On the other hand, the problem could also come from large documents. The system would require large documents to contain query terms multiple times, but it might be the case that only a part of the document discusses the topic that we are interested in. It would mean that the word will not occur throughout the document, so its frequency will be lower, and it will not be chosen as a relevant document. These are why we can think the scaling of 1 would not bring optimal results.

Comparing the results with our initial experiment (see Section 6.1.1 and Table 6.1), we can see that the results for current experiment are not much better than the initial one. We can see that in some cases the highest value is slightly better (e.g., highest *recall* value is 0.22 compared to 0.21 in the initial experiment, and highest *AP* value is 0.17 compared to 0.16 in the previous experiment. However, the difference is very negligible, and we are only considering the best configurations. Hence, it is probably worth to leave k and b values at their defaults (b being 0.75 and k being 1.2).

7 Conclusion

7.1 Summary

The goal of our project was to create an intelligent system that would recommend relevant images based on some text entered by the user. The system would analyse the written text, gather the query terms that most accurately represent the passage, search for documents matching that query, and return the relevant images to the user. Such a tool is highly convenient as it saves users much time because they do not have to look for images by themselves. It also makes it easier to find relevant images because the system performs intelligent techniques to improve its results. The system could be used on many occasions to help a student prepare a slideshow presentation or support a researcher in writing a scientific paper.

Our project analysed two crucial research questions (see Section 1.5).

The first research question asked how effectively can we retrieve the information need from the context. This part corresponds to extracting query terms from the Wikipedia text section, ensuring that they accurately represent the passage. It involved constructing a method to gather query terms, varying parameters, such as the number of terms, to find the configuration that accurately extracts the information needed, testing it on multiple evaluation datasets that we have built.

The second research question asked how effectively can we retrieve the content of different modalities, given some information need. It relates to the second part of our project, which focused on retrieving relevant images based on the query created in the first step. It involved creating an index, querying it, performing experiments to increase our system's performance, and evaluating our model.

Overall, we managed to answer the research questions reasonably sufficiently. We evaluated that our system is exemplary at formulating queries and retrieving relevant images from the index. Although performing experiments on our method using standard evaluation metrics did not produce excellent results, we can tell that our system performs extremely accurately by manually observing the outcomes. Furthermore, we performed multiple experiments that additionally improved our model.

7.2 Reflection

Overall, the project went reasonably well. Although the project's initial idea was slightly different and involved creating a multimodal semantic space to represent both text and images to create a more intelligent agent, we changed the specification due to the lack of time and the project's difficulty. However, we still managed to complete a work of reasonable size and provide novel contributions that could help other researchers.

If I had to do some things differently next time, I would do more research on information retrieval. It was a topic that I had not studied before, and it took considerable time and effort to understand and learn the concepts related to this field.

I would also use the resources provided more extensively. My supervisor provided me with an initial code base that I could use in my project. However, I chose to implement most methods myself, as I found it hard to comprehend the code. Next time, I would spend more time analysing the code and understanding how it works, as it could have saved me much time in this project.

7.3 Future work

There are many improvements to be made to this project. One minor modification could be adding query window size to the model. Instead of adding individual words to the query, we could add combinations of multiple words. That could improve our system's effectiveness as, in some cases, words are closely related to other terms and often appear together. This way, we could capture such occurrences and retrieve more relevant documents.

Another upgrade to the system could be creating a Graphical User Interface (GUI). Our project's main goal was to evaluate the system's performance while extracting the information need and retrieving relevant documents. However, GUI would be a valuable extension to make the system more appealing.

A significant improvement to our system would be implementing a multimodal embedding space representing both images and text together. It would enable us to correlate between different words (e.g., understand that school and university are similar terms) or images (e.g., recognise that a picture of a school and a picture of a university are closely related). However, this would require significant work and extensive research.

A Appendices

Bibliography

Craswell, N. (2009), Mean Reciprocal Rank, *in* L. LIU and M. T. ÖZSU, eds, 'Encyclopedia of Database Systems', Springer US, Boston, MA, pp. 1703–1703.
URL: *https://doi.org/10.1007/978-0-387-39940-9*$_4$88

Frome, A., Corrado, G. S., Shlens, J., Bengio, S., Dean, J., Ranzato, M. A. and Mikolov, T. (2013), DeViSE: A Deep Visual-Semantic Embedding Model, *in* 'Advances in Neural Information Processing Systems', Vol. 26, Curran Associates, Inc.
URL: *https://papers.nips.cc/paper/2013/hash/7cce53cf90577442771720a370c3c723-Abstract.html*

Gupta, M. and Bendersky, M. (2015), 'Information Retrieval with Verbose Queries', *Foundations and Trends® in Information Retrieval* **9**(3-4), 209–354.
URL: *http://www.nowpublishers.com/article/Details/INR-050*

Jain, A., Guo, M., Srinivasan, K., Chen, T., Kudugunta, S., Jia, C., Yang, Y. and Baldridge, J. (2021), 'MURAL: Multimodal, Multitask Retrieval Across Languages', *arXiv:2109.05125 [cs]* . arXiv: 2109.05125.
URL: *http://arxiv.org/abs/2109.05125*

Kim, J., Bae, J., Park, G. and Kim, Y. M. (2021), 'N-ImageNet: Towards Robust, Fine-Grained Object Recognition with Event Cameras', *arXiv:2112.01041 [cs]* . arXiv: 2112.01041.
URL: *http://arxiv.org/abs/2112.01041*

Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., Duerig, T. and Ferrari, V. (2020), 'The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale', *International Journal of Computer Vision* **128**(7), 1956–1981. arXiv: 1811.00982.
URL: *http://arxiv.org/abs/1811.00982*

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L. and Dollár, P. (2015), 'Microsoft COCO: Common Objects in Context', *arXiv:1405.0312 [cs]* . arXiv: 1405.0312.
URL: *http://arxiv.org/abs/1405.0312*

Manning, C. D., Raghavan, P. and Schutze, H. (2008), 'Introduction to Information Retrieval', p. 506.

Srinivasan, K., Raman, K., Chen, J., Bendersky, M. and Najork, M. (2021), 'WIT: Wikipedia-based Image Text Dataset for Multimodal Multilingual Machine Learning', *arXiv:2103.01913 [cs]* . arXiv: 2103.01913.
URL: *http://arxiv.org/abs/2103.01913*

Veit, A., Matera, T., Neumann, L., Matas, J. and Belongie, S. (2016), 'COCO-Text: Dataset and Benchmark for Text Detection and Recognition in Natural Images', *arXiv:1601.07140 [cs]* . arXiv: 1601.07140.
URL: *http://arxiv.org/abs/1601.07140*