Individual Experimental Project Flyer

Robin Rai

1. INTRODUCTION

This flyer introduces an experimental Information Retrieval system. It explains the techniques used to make the system efficient and relevant. The system uses a Vector Space Model and includes processes like stopwords removal, lemmatization, stemming, bigrams, and weighted elements. Further advanced processes such as query expansion and spelling correction are also explored and leveraged to maximize relevancy and efficiency. These techniques are built upon the Inverted Index Model that forms the basis for this system.

The system is split into two, encompassing an indexing mechanism in 'index.py' and a robust search functionality in 'search.py'. This helps separate the two processes removing the need to index every time it comes to using the search engine.

2. BASELINE SYSTEM

2.1 Inverted Index Model

Designed to efficiently organize and retrieve information from a collection of documents, this is the base of the system. Terms are mapped to the documents in which they appear instead of storing the documents in a table and looking up terms within each document. Essentially the relationship is "inverted".

2.2 Preprocessing

Occurring during the building of the Inverted Index Model, this is where a large portion of the techniques used are implemented and heavily modified throughout experimentation.

Regular Expressions are applied here to filter out unwanted characters and clean up text. Terms within the text are also tokenized using the nltk library [1] in which they are then counted in a 'Counter' data-type object. Using a Counter here allows a faster retrieval of Term Frequency (TF)

2.3 Vector Space Model (VSM)

The VSM framework stores documents and queries as vectors in a multi-dimensional space. Using the Cosine Similarity of two vectors, this metric measures the cosine of the angle between them, providing a normalized measure of similarity allowing for effective document retrieval.

The VSM enhances the precision and relevance of the retrieval process and complements the Inverted Index Model's structure of data. Returning a list of documents retrieved ranked based on their weighted TF-IDF scores. With this, the foundation of the system is complete.

3. EXPERIMENTATION OF TECHNIQUES

Techniques were leveraged on their effectiveness for maximizing efficiency and relevancy through a set of 10 training questions conducted on a dataset of 399 HTML documents related to videogames. By querying these questions, the system's capability to deliver relevant documents efficiently could be analyzed.

A point-based system for two questions was used. Points were manually assigned through relevance feedback based on how relevant each document in the ranked list was with two points for Highly Relevant, one point for Relevant, and zero for Not Relevant.

The rest of the eight questions used the weighted TF-IDF value of each document in the list for their question to calculate a 'precision score' metric.

3.1 Stopwords, Stemming and Lemmatization

The system employs a combination of stopwords removal, stemming and lemmatization to preprocess text data and normalize it.

Removal of stopwords helps chunk down the size of the index making it more efficient and compact while reducing noise from frequent terms with little semantic meaning. This sets up the system to be an even better foundation for when it comes to applying future techniques.

Stemming and Lemmatization fit into the same category as their end goal is to normalize the data into their base or root forms to improve recall. However, application of either of these two into the system seems to rather harm it in some cases than help (see Figure 1).

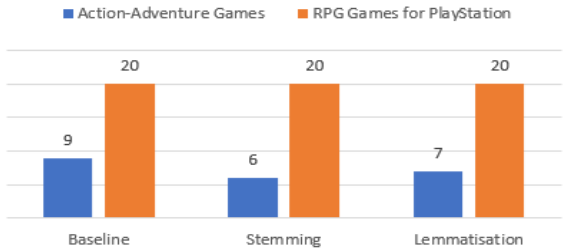

Figure 1. Effectiveness of Stemming and Lemmatization on precision using a point-based scoring system.

Even once more techniques are added, Stemming and Lemmatization show little to none impact on the effectiveness and efficiency of the system (seen in Figure 2).

3.2 Bigrams

Including bigrams in the indexing process, identified through a threshold-based approach, allows the system to create a more detailed and informative index. This can lead to more accurate and specific retrieval results meaning greater precision, and captures semantic proximity, preserving the relationship between adjacent words. This information can be valuable for understanding the subject matter of a document.

3.3 Weighted Elements (Titles, headers, etc.)

Elements such as titles and headers receive greater weighting during the building of the Inverted Index Model, resulting in a greater importance for those sections in the retrieval process [2]. In this system elements with a class attribute containing "Info" are also assigned greater weights. This improves relevancy as these sections contain important characteristics such as 'genre', 'ESRB' and 'Developer' (see Table 1).

Training Question	Baseline	Weighted
Action-Adventure Games	9 pts	14 pts
RPG Games for PlayStation	20 pts	20 Pts (unchanged)

Table 1. Impact of weighted elements on precision using a point-based scoring system

Effects of Stemming and Lemmatization were also experimented here, yet they still show little impact (see Figure 2).

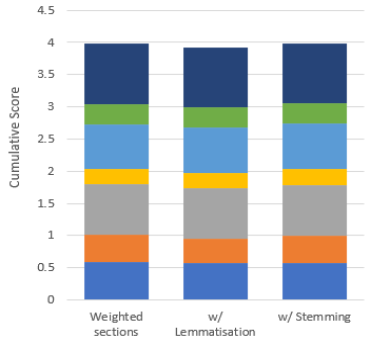

Figure 2. Comparative analysis of Stemming and Lemmatization with weighted elements using cumulative precision scores

3.4 Query Expansion

Query expansion enhances the query by augmenting it with additional synonyms and related terms through Part-of-Speech (POS) tagging, enriching the information retrieval process. These additional terms are added A pivotal technique used to employ greater recall into the system. The drawback is that it greatly reduced precision (seen in Figure 3).

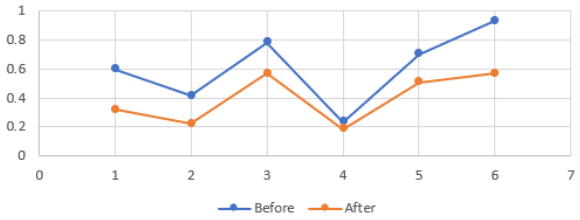

Figure 3. Impact of implementing Query Expansion on the precision of 8 training questions

3.4.1 Weighted Query Expansion

Terms injected from Query Expansion are assigned lower weights. This allows the system to benefit from the greater recall it receives while maintaining the precision of retrieval it had before Query Expansion.

3.5 Spelling Correction

A spelling correction mechanism is integrated by using the 'FuzzyWuzzy' python module [2]. With this module, the Levenshtein Distance, a widely accepted method for measuring the edit distance between two words, is calculated for incorrectly spelt or unknown query terms with pre-existing terms in the Inverted Index Model. The identification and correctness of these terms improve upon the query accuracy and enhance the precision of the retrieval system (see Figure 4).

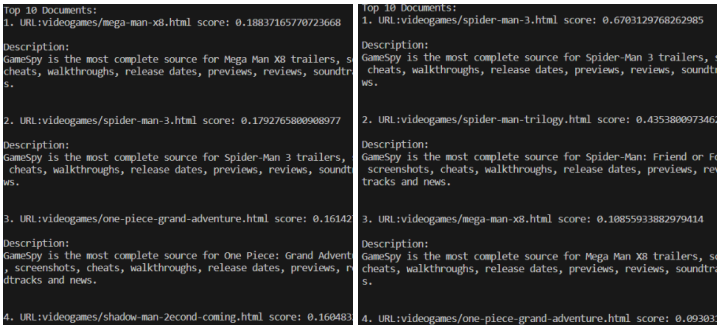

Figure 4. Before (left) and After (right) screenshots of the first few documents in the retrieved ranked list when querying "Spyder-Man"

4. CONCLUSION

In conclusion, through an amalgamation of different indexing techniques and search functionalities, the Information Retrieval system provides an effective solution for searching relevant documents with efficiency. The experiments conducted validate the performance of the system yet also show and encourage the plausibility of further exploration and experimentation to optimize the system and expand its capabilities.

5. REFERENCES

[1] NLTK Documentation. https://www.nltk.org/

[2] S. E. Robertson & K. Sparck Jones, 1976. "Relevance weighting of search terms," Journal of the American Society for Information Science, Association for Information Science & Technology, vol. 27(3), pages 129-146, May.

[3] Inc S (2014). fuzzywuzzy: Fuzzy String Matching in Python. https://github.com/seatgeek/fuzzywuzzy