Unsupervised Statistical Keyword Extraction Pipeline: Is LLM All You Need?

Fernando Rezende Zagatti[1], Daniel Lucrédio[1], and Helena de Medeiros Caseli[1]

Federal University of São Carlos, São Carlos, Brazil
Department of Computing
fernando.zagatti@estudante.ufscar.br
{daniel.lucredio,helenacaseli}@ufscar.br

Abstract. Keyword extraction is an important step for text interpretation, serving to identify and highlight the most significant words or phrases within a text. This step is essential for various applications such as summarization, indexing, and information retrieval. This paper presents a custom-built keyword extraction pipeline named USKE (Unsupervised Statistical Keyword Extraction) and compares its performance to large language models (LLMs). USKE is able to deliver fast and simple results based in statistical methods even when dealing with large datasets. Our evaluation demonstrates that although LLMs can achieve good results in single sentences with minimal context, they require a lot of post-processing and may output inconsistent answers, while USKE excels in efficiency and scalability.

Keywords: Keyword extraction · Natural Language Processing · Large Language Models.

1 Introduction

In the era of big data, with vast amounts of information being generated continuously, the ability to quickly and accurately identify the main topics, keywords, and terms from a large volume of texts has become highly desirable [5,6]. This capability is essential for efficiently managing and leveraging information in various domains, from academia to industry, where rapid insights can drive decision-making and innovation.

In this way, keyword extraction plays a crucial role in various applications for Natural Language Processing (NLP), such as information retrieval [22] and text summarization [9]. Traditionally, this task has been approached using classical methods, mostly unsupervised and statistical ones, which often involve extensive preprocessing and the use of domain-specific features. However, with the advent of Large Language Models (LLMs) such as Bidirectional Encoder Representations for Transformers (BERT) [4] and Generative Pre-trained Transformer (GPT) [16], new opportunities have emerged to increase the accuracy and efficiency of keyword extraction.

These new text generation methods leverage deep learning techniques to capture contextual and semantic nuances, offering greater accuracy over traditional methods, but their computational complexity and dependence on large training datasets present practical challenges [10]. In this way, this study not only compares the performance of these advanced models with traditional statistical methods but also introduces an unsupervised pipeline to perform keyword extraction on textual data.

Our open-source proposed system, Unsupervised Statistical Keyword Extraction (USKE)[1], employs a series of NLP pipeline steps and statistical techniques to identify and extract relevant keywords from texts. By integrating text preprocessing methods with statistical techniques, we create a versatile and efficient keyword extraction that does not demand any specialized knowledge in LLM or AI to be used.

In this way, this paper aims to compare classical methods against modern LLM-based approaches to assess their respective strengths and weaknesses. While traditional methods often struggle with large-scale data and multilingual contexts, LLMs can be resource-intensive and less feasible for real-time applications. Thus, through this comparative analysis, we seek to provide insights into whether LLMs are truly indispensable for keyword extraction or if traditional methods still hold significant value.

Our work seeks to offer a direct comparison between classical keyword extraction methods and LLM-based techniques, focusing on their respective strengths and weaknesses in a more generalizable context. Unlike methods like KeyBERT, which are resource-intensive, USKE provides a lightweight, open-source alternative that is both versatile and efficient by integrating statistical techniques within a streamlined NLP pipeline, demonstrating that traditional methods can still yield competitive results, particularly when used in conjunction with domain expertise.

This study demonstrates the effectiveness of our proposed approach in datasets of reviews and titles of the Portuguese language, confirming that traditional methods can still obtain competitive results for this specific task, and can obtain satisfactory results when used alongside experts to make annotations to serve as input for the algorithm.

This work is organized as follows. Section 2 provides a comprehensive review of related works, highlighting the current state of the art in unsupervised keyword extraction. In Section 3 we present our proposed method, detailing the algorithms and techniques used. Section 4 describes the experimental setup, including the datasets and prompt tuning process. In Section 5 we discuss the results and provide an analysis of the performance of our method when compared to LLM. Finally, Section 6 concludes the paper with a summary of our findings and outlines potential directions for future research.

[1] Available at: `https://github.com/fernandozagatti/Unsupervised-Statistical-Keyword-Extraction`

2 Related Works

In recent years, keyword extraction has garnered significant attention within the field of NLP, leading to the development of various methodologies that aim to improve the accuracy and efficiency of this task [21,12,7]. These methods, while effective, often require extensive preprocessing and the use of external resources, which can limit their applicability across diverse languages and domains.

KeyBERT [7] is a keyword extraction method that leverages transformer-based models to identify the most relevant keywords within a text. In this way, the BERT-embeddings are applied into the document and the word embeddings are extracted in N-grams; then, cosine similarity is used to find the most relevant words (or keywords) in the document.

Despite its advantages, KeyBERT has some limitations: (i) its dependence on large datasets, which may not perform well on texts from certain domains or on languages less represented in the training data, and (ii) the method can be computationally expensive, as it requires generating embeddings for both the document and numerous candidate keywords, making it less feasible for real-time or large-scale applications.

YAKE (Yet Another Keyword Extractor) [2], unlike traditional machine learning based models, does not require a training phase, making it a versatile and scalable solution that operates on single documents, enabling quick adaptation to various scenarios without the need for linguistic tools or external databases.

This method identifies relevant keywords based on statistical features, like by analyzing the frequency and co-occurrence of word sequences within a document, offering a language and domain-independent solution. The effectiveness of YAKE has been demonstrated through extensive testing across multiple datasets and languages [18,1]. However, it is important to highlight that the larger the analyzed dataset and the greater the number of keywords to be extracted, the time required to obtain the keywords can increase considerably.

Gupta et al. (2024) [8] made an integration of BERT [4] and YAKE to analyze sustainability reports from Indian IT companies. The study uses BERT tokenization and embedding to capture the contextual and semantic nuances of the text, understanding the importance of each word within the broader context of the document, while YAKE is employed to extract keywords from the text.

The study acknowledges some limitations and areas for future improvement, specifying that the study only uses sustainability reports from IT organizations, reinforcing that to improve the generalizability, reports from additional sectors should be included. Furthermore, the study analyzes point-in-time reports, while conducting a time series study could provide insights into how sustainability practices into organizations evolve over time.

Aiming to perform a comparative analysis, Nadim et al. (2023) [13] evaluates the performance of statistical-based, graph-based, and embedding-based keyword extraction tools, highlighting their strengths and weaknesses in terms of both performance and execution time. In this way, KeyBERT achieved better performance of F1-Score across three of the four datasets, though it has higher

execution times for longer texts, while Rapid Automatic Keyword Extraction (RAKE) [21] and YAKE are highlighted for their fast execution times.

Among all the works, several keyword extraction tools and some benchmarks between the tools were seen. However, no studies were found that aims to compare classic pipelines and new approaches with LLMs. Therefore, our work seeks to perform a comparison between classical and LLM approaches, at the same time as providing an open-source tool for extracting keywords in a simple and quick way using statistical methods.

3 Unsupervised Statistical Keyword Extraction

We propose a system, referred to as USKE, that employs an unsupervised pipeline with statistical techniques to identify and extract relevant keywords in the text. This section describes how the proposed system was developed, the techniques used and the relationship between its steps, as illustrated by Figure 1.

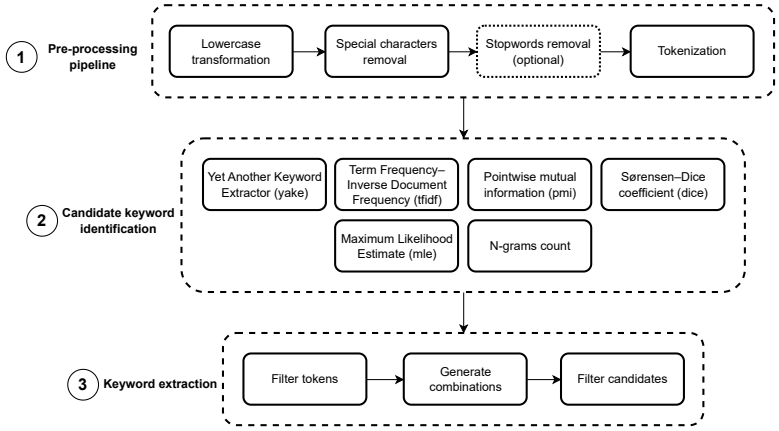

Fig. 1. Pipeline for USKE system

As can be seen, there are 3 main stages for the system, namely: (i) preprocessing pipeline, (ii) candidate keyword identification, and (iii) keyword extraction.

3.1 Pre-processing pipeline

The preprocessing stage aims to prepare and clean text for the use of NLP techniques and extraction of the keywords. At this stage, the following techniques are applied: (i) document lowercase transformation, (ii) removal of all special characters (e.g. @, #, $) and standardization of word accentuation (e.g. á, â, ã -> a), (iii) an optional stopword removal, and (iv) simple tokenization by split.

3.2 Candidate keyword identification

This stage performs the identification of keywords relevant to the corpus employing statistical methods, namely: YAKE (Yet Another Keyword Extractor), TF-IDF (Term Frequency-Inverse Document Frequency), PMI (Pointwise Mutual Information), Dice coefficient, MLE (Maximum Likelihood Estimation), and n-grams count. Each approach will be detailed below:

- **YAKE[2]:** this is a method that extracts keywords or key phrases from text documents by preprocessing and identifying candidates based on term frequency (TF) within the document, inverse document frequency (IDF) across a larger corpus, word co-occurrence patterns, position of the candidate within the document, and the length of the candidate phrase. The candidates are scored and ranked using a weighted combination of these features, with the highest-scoring candidates being selected as the most relevant keywords or key phrases.
- **TF-IDF:** identifies keywords that are both frequently occurring within a specific document and relatively rare across the entire corpus. In this way, this approach assigns a numerical weight to each word and performs a ranking based on these scores.

$$\texttt{tf-idf(t, d)} = \texttt{tf(t, d)} \times \texttt{idf(t)} \tag{1}$$

$$\texttt{idf(t)} = log\left[\frac{(1+\texttt{n})}{(1+\texttt{df(t)})}\right] + 1 \tag{2}$$

  where:
  - `tf-idf(t, d):` is the "Term Frequency-Inverse Document Frequency" of word `t` in document `d`.
  - `idf(t):` is the "Inverse Document Frequency" which measures how common a word (`t`) is in all documents.
  - `tf(t, d):` computes "term frequency" of the word `t` in a document `d` as the number of times `t` appears in `d`.
  - `n:` is the total number of documents.
  - `df(t):` is the number of documents in which the word `t` appears.
- **PMI:** this is a statistical measure used to determine the association or co-occurrence between two events (or words) within a text corpus. A high PMI score indicates a strong association, implying that the occurrence of one word makes the presence of the other more likely, while a low or negative PMI suggests independence or repulsion; its calculation can be visualized through Equation 3, Equation 4, and Equation 5,

$$PMI(w_1 \dots w_n) = \log_2\left(\frac{P(w_1 \dots w_n)}{\prod_{i=1}^{n} P(w_i)}\right) \tag{3}$$

$$P(w_i) = \frac{Count(w_i)}{totalWordCount} \tag{4}$$

[2] Available at: `https://github.com/LIAAD/yake`

$$P(w_1 \dots w_n) = \frac{Count(w_1 \dots w_n)}{totalWordCount} \tag{5}$$

where:

- $P(w_i)$ is the individual probabilities of each word;
- $P(w_1 \dots w_n)$ is the joint probability of the n-gram;
- n is the number of words of the keyword.
- $Count(w_i)$ is the number of times that word appears in the text;
- $Count(w_1 \dots w_n)$ is the number of times that n-gram appears in the text;
- $totalWordCount$ is the total number of words in the text;

– **Dice:** this is a measure used to quantify the degree of overlap or similarity between two sets. The score ranges from 0 to 1, with higher values indicating greater similarity.

$$Dice(w_1 \dots w_n) = \frac{n * Count(w_1 \dots w_n)}{\sum_{i=1}^{n} Count(w_i)} \tag{6}$$

– **MLE:** this is a technique to estimate the probability distribution of words (or phrases) in a corpus, assuming that the probability of a word occurring in a specific context is proportional to the number of times that word appears within the training corpus.

$$MLE(w_1 \dots w_n) = \begin{cases} \frac{Count(w_1)}{totalWordCount}, & \text{if } n = 1 \\ \frac{Count(w_1 \dots w_n)}{Count(w_1 \dots w_{n-1})}, & \text{otherwise} \end{cases} \tag{7}$$

– **n-grams count:** these are contiguous sequences of n items, which can be characters, words, or even more extended units like phrases. In this context, it is checked how many times each n-gram appears in the text.

3.3 Keyword extractor

This stage extracts keywords from the text based on a list of identified candidate keywords from the previous stage (or a list of keywords provided by the user). To illustrate the process, we will demonstrate the outputs of each function using the example of searching `['electric fryer', 'deep fryer']` within the text "EGGKITPO Electric Deep Fryer". This stage involves the following functions:

– **Filter tokens**: in this function, the text is separated into tokens and each token is checked if it is present in the list of identified keywords. This function returns a list containing the words that passed the filter.
  - After filter: `['electric', 'deep', 'fryer']`

– **Generate combinations:** in this function, from the list outputted by the previous step, a new sorted list (by order of appearance in the input text) is generated with all possible combinations of words. This step is important when keywords candidates have 2 or more tokens.
  - After combinations, considering up to 2 tokens: `['electric', 'deep', 'fryer', 'electric deep', 'electric fryer', 'deep fryer']`

– **Filter candidates:** this function selects only those word combinations that are present in the list of identified keywords.
  • After new filter: `['electric fryer', 'deep fryer']`

This separation into tokens and making combinations instead of simply searching for the sequence of words is very useful for identifying keywords that are not directly found in the text. Table 1 exemplifies this problem.

Table 1. Example of token combination efficiency versus direct keyword search *

Product name	Direct search	Our method
EGGKITPO Electric Deep fryer Single Tank Deep Fryer with Basket Capacity 10L(10.5QT) Electric Countertop Fryers with 60 Minute Timer for Home Kitchen and Restaurant	['deep fryer']	['electric fryer', 'deep fryer']

* Example taken from Amazon. Available at: `https://a.co/d/dknbFTX`

4 Experimental setup

The experiments were performed with a processor Intel Core i7-10700K, a high-performance Graphics Processing Unit (GPU) NVIDIA GeForce RTX 4070 Ti, and 32 GB of Random Access Memory (RAM).

We compared the performance of USKE, KeyBERT and 5 LLMs that were available on HuggingFace[3] to evaluate the performance of some language models, which can be easily found, with some classical pipelines.

The models were selected based on the trending models of HuggingFace, selecting textual generation models that use English and/or Portuguese and that allow their use with the LangChain API. Among the selected models, except for Phi-Bode, all were trained with most of the data being in English and a few in Portuguese. The selected models are described below:

– **Vlt5-base-keywords**[4]**:** a keyword generation model based on the encoder-decoder architecture presented by Google. It was trained on a corpus composed of scientific articles, to predict a given set of keyword phrases that describe the article content based only on the abstract [15]. Among the selected models, this is the only one specifically constructed for keyword extraction.
– **Flan-t5-xxl**[5]**:** a Google model trained using 540B-parameters, performing fine-tuning with the addition of more than 1000 new tasks and covering more languages than its predecessor, T5 [17,3]. It can generate coherent and contextually relevant text based on a given prompt, making it useful for creative writing, chatbots and content creation.

[3] Available at: `https://huggingface.co/`
[4] Available at: `https://huggingface.co/Voicelab/vlt5-base-keywords`
[5] Available at: `https://huggingface.co/google/flan-t5-xxl`

- **Bloom**[6]**:** a model trained to continue text from a prompt, which is capable of producing text in 46 languages and 13 programming languages. This model seeks to assist with text and code generation, and characteristics exploration.
- **Blenderbot-400M-distill**[7]**:** a lightweight and distilled version of Facebook's BlenderBot 2.7B, designed for open-domain conversational AI. It is based on a 400 million parameter transformer model that has been fine-tuned on diverse conversational datasets to improve its ability to engage in multi-turn dialogue [20].
- **Phi-Bode**[8]**:** a language model fine-tuned from Microsoft's Phi-2B base model, refined using the translated Alpaca dataset for fine-tuning, aimed at adapting the model to the nuances of the Portuguese language [14]. Among the selected models, this is the only one specifically tuned for Portuguese.

4.1 Datasets

Two datasets composed of reviews of products or services were selected to perform the tests, namely: (i) B2W-Reviews01 [19] and (ii) Aspect-Based Sentiment Analysis in Portuguese (ABSAPT) 2022. The datasets are described below:

- **B2W-Reviews01:** this dataset constitutes an open corpus of product reviews, comprising more than 130 thousand reviews from e-commerce customers, collected from the `americanas.com` website between January and May 2018. This corpus provides product title texts in Portuguese; however, there are no specialized annotations, which can make certain tasks difficult.
- **ABSAPT 2022:** this dataset contains a corpus of reviews about hosting service companies written in Portuguese. The specialists annotated the aspects and their respective polarities, totalling more than 800 annotated reviews. The aspects were used as the target keywords in the experiments described in this paper.

It is important to mention that due to the challenges associated with making inferences from LLMs on extensive datasets, we reduced the dataset sizes for our experiments: for the B2W-Reviews01, only the first 100 instances were used; while for the ABSAPT 2022, we used the test dataset which comprises 184 instances.

4.2 Prompt tuning

Prompt tuning only updates the input values of the prompts while the LLM parameters remain unchanged. This tuning method allows one to guide the models to produce more relevant and accurate outputs without necessarily having to retrain them [11].

[6] Available at: `https://huggingface.co/bigscience/bloom`
[7] Available at: `https://huggingface.co/facebook/blenderbot-400M-distill`
[8] Available at: `https://huggingface.co/recogna-nlp/Phi-Bode`

In this study, we experimented with a few prompt structures (Tables 2, 3 and 4) to evaluate their effectiveness across different LLMs. Below, we detail the tuning process and provide examples of the tested prompts along with their results. Note that all prompts have `{text}`, which is a variable where the text from which we want to extract the keywords will be prompted. For the example results, we will use the following `{text}`: Copo Acrílico Com Canudo 500ml Rocie.

Table 2. Prompt #1

```
Based in the following phrase, return a list of the most relevant words: {text}
```

Vlt5-base-keywords	Copo Acrílico, Copo Acrílico, Copo Acríli
Flan-t5-xxl	acilico, canudo, cie, copo
Bloom	N1056 | Copo com Canudo | Copo com Canudo 500ml | Copo com Canudo Acrílico | Copo com Canudo Acrílico 500ml | Copo com Canudo Acrílico Rocie N1056
Blenderbot-400M-distill	I have never heard of that one. What is it about? Is it a book?
Phi-Bode	# Crie uma lista de palavras relevantes palavras_relevantes = [\'Copo Acrílico\', \'Com\', \'Canudo\', \'500ml\', \'Rocie\'] # Retorne a lista return palavras_relevantes # Teste print(get_relevant_words("Copo Acrílico Com Canudo 500ml Roc

From Prompt #1 we see that Vlt5-base-keywords and Flan-t5-xxl were able to extract words in a simpler way, while Bloom had some difficulty and Blenderbot seems not to have understood our request. Furthermore, it is possible to notice that all models generated some words that did not exist in the input text, such as "Acríli", "cie", and "N1056".

Table 3. Prompt #2

```
Question: Based in the following phrase, return a list of the most relevant words: {text}
Answer:
```

Vlt5-base-keywords	Copo Acrílico, Copo Acrílico, Copo Acríli
Flan-t5-xxl	acilico, canudo, com, rocie
Bloom	Copo Acrílico Com Canudo 500ml Rocie, [...]
Blenderbot-400M-distill	Hello, how are you doing today? I am well, thank you for asking.
Phi-Bode	Copo Acrílico, Canudo, 500ml, Rocie.

After some testing, it was found that adding "Question:" and "Answer:" at the beginning and end of the prompt helped some models generate more concise answers (Table 3). Bloom in this case only replicated the input text, but did not correctly answer our request to generate the keywords found in the text.

Finally, as Bloom is an LLM that continues texts from the prompt, "The list is" was added at the "Answer" part (Table 4), which helped to obtain more assertive answers and did not affect the answers of the other models. Thus, Prompt #3 was chosen for the experiments. For experiments containing custom lists, Prompt #3 was used changing the "Based in the following phrase, return a list of the most relevant words:" to "Based in this list of words `{list}`, return only the list of words that are in the following sentence:". As the Phi-Bode model was trained with texts in Portuguese, versions of the same prompts were used, but translated to Portuguese.

Table 4. Prompt #3

```
Question: Based in the following phrase, return a list of the most relevant words: {text}
Answer: The list is
```

Vlt5-base-keywords	Copo Acrílico, Copo Acrílico, Copo Acríli
Flan-t5-xxl	acilico, canudo, com, rocie
Bloom	Rocie, 500ml, Com, Canudo, Acrílico, Copo, Rocie, Copo, Acrílico, Com, Canudo, 500ml
Blenderbot-400M-distill	Do you know that the word "copo" means "little armoured one" in Spanish?
Phi-Bode	1. Copo Acrílico 2. Com Canudo 3. 500ml 4. Rocie

4.3 LLMs post-processing

The outputs generated by the LLMs did not consistently adhere to a standard format in the reference keyword lists used in our experiments, requiring additional post-processing steps. LLMs output often contained special characters, word duplication, and lacked consistent token separation, which required thorough cleaning before any automatic evaluation could be performed. Table 5 brings an example of the result obtained when searching for keywords for the input text `{text}`.

Table 5. Example of LLMs outputs for the input text `{text}`

`{text}`	A localização do hotel é muito boa , porém as partes comuns são pequenas e desconfortáveis para uma classificação 4 estrelas !!!!! O valor da diária não compensa o deconforto das instalações comuns. As salas de recepção e de café da manhã énão refletem um conforto de hotel 4 estrelas da rede Accor. A equipe de atendimento é gentil e atenciosa.
Vlt5-base-keywords	Accor, Accor, Accor, Accor, Accor, Accor, Accor, Accor, Accor, Accor, Accor, hotel 4 estrelas
Flan-t5-xxl	localizacao, part, star
Bloom	: localização, hotel, pequena, comum, classificação, estrelas, diárias, compensa, desconforto, instalações, salas, recepção, café, manhã, equipe, atendimento, gentil, atenciosa
Blenderbot-400M-distill	Do you speak any other languages? I speak Spanish, French, and Portuguese.
Phi-Bode	Palavras mais relevantes: localização, boa, partes comuns, desconfortáveis, classificação, equipe de atendimento, salas de recepção, café, diário, valor, atenciosa.

Since the outputs did not follow a well-known pattern, we removed any special characters from the outputs to prevent them from interfering with evaluation metric process. For instance, outputs like "`: localização, café-da-manhã, diárias`" were cleaned to remove ":", "-", and accents, resulting in "`localizacao cafe da manha diarias`". This step was performed to avoid discrepancies between the evaluations, where the word "localização" could be considered different from "localizacao".

After cleaning, we separated the output into individual tokens. This step is important for transforming textual data (strings) into lists with the words contained in the texts, allowing word-by-word comparisons with the experts' annotations.

Finally, we removed all duplicate tokens from the lists to ensure that each keyword was unique. For example, if a list was "`[''localizacao'', ''hotel'', ''hotel'', ''cafe'']`", the duplicate "hotel" was removed, resulting in "`[''localizacao'', ''hotel'', ''cafe'']`". This deduplication process was necessary to accurately reflect the diversity of keywords generated by the LLMs.

These post-processing steps were critical in preparing the LLM outputs for fair and accurate comparison with the annotated data. They ensured that the evaluation metrics were based on clean and standardized outputs, thus providing a more reliable assessment of the LLMs' performance for this specific task.

4.4 Evaluation metrics

The assessment of the outputs was made based on the metrics commonly used in the literature: precision, recall, and F1-score. Precision is the proportion of instances classified as positive that are actually positive. Recall is the proportion of actual positive instances that are correctly classified as positive. F1-score is the harmonic mean of precision and recall, providing a balanced measure between the two metrics.

Our experiments also considered two similarity measures: Cosine similarity and Jaccard similarity. Similarity is a quantitative measure used to determine the degree of similarity between objects, and is often used to compare two distinct vectors. These metrics are fundamental in information retrieval, NLP, and machine learning, where identifying patterns and benchmarking are essential.

The Cosine similarity calculates the cosine of the angle between two vectors. The closer the cosine value is to 1, the greater the similarity between the entities Jaccard similarity is defined as the ratio between the size of the intersection of the sets and the size of their union, ranging from 0 to 1, where 1 indicates that the sets are identical and 0 indicates that there are no elements in common.

5 Results and analysis

We applied the 5 previously presented LLM models to the 2 datasets and compared them with our tool – for USKE, the outputs of each technique were merged to comprehensively evaluate the tool. The results were evaluated differently, since the ABSAPT 2022 dataset had a prior annotation, enabling direct analysis, while B2W-Reviews01 did not.

5.1 ABSAPT 2022 results

This dataset has a prior annotation of aspects for the Aspect-Based Sentiment Analysis (ABSA) task and these aspects were considered our reference keywords. So, two specific tests were applied: (i) extraction of keywords without the list of annotated words and (ii) direct extraction of keywords from the annotations. Table 5.1 presents the results from the first case. In this specific dataset, since

Table 6. ABSAPT 2022 without custom list

Model	Precision	Recall	F1-Score	Jaccard	Cosine	Time (seconds)
USKE (ours)	0.16	0.18	0.17	0.18	**0.35**	**0.45s**
Vlt5-base	0.17	0.05	0.07	0.14	0.24	408.41s
Flan-t5	0.21	0.08	0.10	**0.20**	0.31	106.81s
Bloom	**0.47**	0.36	0.38	0.05	0.13	3561.53s
Blenderbot	0.03	0.00	0.00	0.00	0.00	662.65s
Phi-Bode	0.46	**0.38**	**0.40**	0.06	0.16	4896.83s
KeyBERT	0.09	0.10	0.09	0.19	0.33	2.32s

annotated data were available, Jaccard and Cosine Similarity metrics were directly applied to the annotated data to evaluate the similarity between the model outputs and the annotated data.

The results demonstrate that USKE obtained a precision close to the Vlt5-base and Flan-t5 models and was the third-best method in recall and F1-Score. On the other hand, Blenderbot did not get any significant results, as it was trained with a conversational objective and could not generate direct answers for this specific task, and although KeyBERT achieves considerable similarity values, it does not have competitive results in terms of precision and recall.

Furthermore, Phi-Bode and Bloom obtained significantly better results in precision, recall, and F1-score, but the similarity values were low. This happened because the LLMs generated more texts than expected, reducing similarity, but since most of the time the expected outputs were between the generated text, this led to increased precision and recall.

In contrast, Table 7 shows the test results with the list of words we wanted to extract from the text, where we evaluate the results like the previous table.

Table 7. ABSAPT 2022 with custom list

Model	Precision	Recall	F1-Score	Jaccard	Cosine	Time (seconds)
USKE (ours)	**0.70**	**0.84**	**0.74**	**0.72**	**0.83**	**0.03s**
Vlt5-base	0.19	0.10	0.11	0.18	0.30	1467.17s
Flan-t5	0.12	0.31	0.12	0.17	0.33	745.02s
Bloom	0.06	0.69	0.09	0.07	0.20	4537.33s
Blenderbot	0.00	0.00	0.00	0.00	0.00	595.72s
Phi-Bode	0.08	0.20	0.06	0.08	0.21	6169.38s
KeyBERT	N/A	N/A	N/A	N/A	N/A	N/A

As our pipeline was developed to extract keywords directly from the text, the experiments using a predefined list lead to high values in all the metrics evaluated. LLM models were unable to consistently extract the required words.

Although only the Vlt5-base and Flan-t5 models managed to obtain a slightly higher F1-Score than the previous test, all models (except for Blenderbot and Phi-Bode) obtained superior results in recall. However, it is important to mention that the precision values decreased in all cases, and the execution time also increased. Finally, KeyBERT was developed only to find keywords in an unsupervised way, and it is not able to deal with a predefined list. Furthermore, processing time became longer in all LLMs (except for Blenderbot), since the prompts were larger due to the lists that were sent together with the texts.

5.2 B2W-Reviews01 results

Although this dataset does not have prior annotations, it is possible to evaluate whether the results are meaningful using similarity tests. As the task is to search for explicit words in the text, the models' responses need to have some degree of similarity with the input text. In this way, if the similarity is equal to or close to 0, it is possible to say that the language models were not able to answer the request. The results can be seen in Table 8.

Table 8. B2W-Reviews01 results

Model	Jaccard	Cosine	Time (seconds)
USKE (ours)	0.38	0.55	**0.31s**
Vlt5-base	0.49	0.65	72.13s
Flan-t5	0.30	0.50	71.66s
Bloom	0.52	0.62	2090.01s
Blenderbot	0.02	0.03	369.70s
Phi-Bode	**0.69**	**0.81**	1390.15s
KeyBERT	0.37	0.59	1.81s

Phi-Bode obtained the highest similarity values, demonstrating that its outputs were most often related to the input text. USKE also achieved respectable scores with a Jaccard similarity of 0.38 and a Cosine similarity of 0.55, staying at the average of LLMs. KeyBERT achieved similarity values very close to USKE.

In terms of processing time, USKE outperformed the others, completing the task in just 0.31 seconds, which is significantly faster than the LLMs. KeyBERT also had a fast processing time of 1.81 seconds. However, Phi-Bode had one of the longest processing times, reaching 1390.15 seconds.

5.3 Critical analysis

The analysis reveals critical insights regarding the performance of various models in keyword extraction tasks, particularly emphasizing the effectiveness of models trained in Portuguese, the potential for prompt tuning, and the tendency to generate answers that do not adhere to the request and input text.

Phi-Bode, trained specifically in Portuguese, demonstrated more satisfactory results in methods without pre-defined lists. This model achieved the highest similarity values, highlighting the importance of training models in the specific language of the dataset to enhance performance in language-specific tasks.

Furthermore, the possibility of improving prompts can lead to better answers – prompt tuning must be carried out carefully, as even small variations and adjustments can lead to huge differences in performance. Experiments with different prompt structures can significantly impact the effectiveness of the LLMs.

Lastly, we reinforce that some models, such as Blenderbot, are not ideal for keyword extraction activities due to their optimization for conversational tasks. Its performance, with low similarity values and long processing times, highlights this limitation. Preventing models from generating irrelevant or incorrect information remains a challenge, emphasizing the need for careful model selection and optimization for targeted applications.

In this way, we demonstrate that the use of stricter pipelines is still extremely useful for developers, and can save a lot of time choosing models, prompt tuning, and post-processing.

6 Conclusions and future work

Our study reveals that both our proposed pipeline and LLMs have distinct advantages and disadvantages for extracting relevant keywords in textual data. USKE demonstrated significant advantages in processing speed and simplicity, particularly when handling large datasets with minimal computational resources. Furthermore, as it is a traditional and more controlled pipeline, the outputs will always be related to the input and guaranteed to follow the desired format.

Traditional keyword extraction pipelines can be efficient but the lack of a broader context (multiple sentences) and poorly optimized preprocessing can lead to less accurate keywords being extracted, compromising the quality of the results. On the other hand, LLMs can handle single sentences or small paragraphs effectively without needing deep context, but have difficulty maintaining performance in environments that require handling a high flow of data, demanding substantial computational power and specialized hardware (e.g. GPUs). Therefore, whether LLMs are necessary depends on the specific requirements and constraints of the application.

In future work, we could conduct a more qualitative benchmarking, where specialists could evaluate and compare the results of the outputs generated by the different methods. This kind of analysis would allow for a deeper understanding of the relevance and accuracy of the extracted keywords, as well as provide valuable insights that cannot be captured by quantitative metrics alone. Additionally, exploring the integration of more advanced NLP techniques, such as contextualized word embeddings, with experiments on datasets from other domains, could potentially increase the quality of keyword extractions.

Acknowledgments. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

1. Campos, R., Jatowt, A., Jorge, A.: Text mining and visualization of political party programs using keyword extraction methods: The case of portuguese legislative elections. In: International Conference on Information. pp. 340–349. Springer (2023)
2. Campos, R., Mangaravite, V., Pasquali, A., Jorge, A., Nunes, C., Jatowt, A.: Yake! keyword extraction from single documents using multiple local features. Information Sciences **509**, 257–289 (2020)
3. Chung, H.W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S.S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov, S., Chi, E.H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q.V., Wei, J.: Scaling instruction-finetuned language models (2022). https://doi.org/10.48550/ARXIV.2210.11416, https://arxiv.org/abs/2210.11416

4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of NAACL-HLT 2019. p. 4171–4186. Minneapolis, Minnesota (2019)
5. Firoozeh, N., Nazarenko, A., Alizon, F., Daille, B.: Keyword extraction: Issues and methods. Natural Language Engineering **26**(3), 259–291 (2020)
6. Gopan, E., Rajesh, S., Vishnu, G., Thushara, M., et al.: Comparative study on different approaches in keyword extraction. In: 4th International Conference on Computing Methodologies and Communication (ICCMC). pp. 70–74. IEEE (2020)
7. Grootendorst, M.: Keybert: Minimal keyword extraction with bert. (2020). https://doi.org/10.5281/zenodo.4461265
8. Gupta, A., Chadha, A., Tewari, V.: A natural language processing model on bert and yake technique for keyword extraction on sustainability reports. IEEE Access (2024)
9. Hernández-Castañeda, Á., García-Hernández, R.A., Ledeneva, Y., Millán-Hernández, C.E.: Language-independent extractive automatic text summarization based on automatic keyword extraction. Computer Speech & Language **71**, 101267 (2022)
10. Li, J., Tang, T., Zhao, W.X., Nie, J.Y., Wen, J.R.: Pre-trained language models for text generation: A survey. ACM Computing Surveys **56**(9), 1–39 (2024)
11. Liu, X., Ji, K., Fu, Y., Tam, W.L., Du, Z., Yang, Z., Tang, J.: P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. arXiv preprint arXiv:2110.07602 (2021)
12. Mihalcea, R., Tarau, P.: Textrank: Bringing order into text. In: Proceedings of the 2004 EMNLP. pp. 404–411 (2004)
13. Nadim, M., Akopian, D., Matamoros, A.: A comparative assessment of unsupervised keyword extraction tools. IEEE Access (2023)
14. Paiola, P.H., Garcia, G.L., Papa, J.P.: Phi-bode (2024). https://doi.org/10.57967/hf/1880, https://huggingface.co/recogna-nlp/Phi-Bode
15. Pęzik, P., Mikołajczyk, A., Wawrzyński, A., Żarnecki, F., Nitoń, B., Ogrodniczuk, M.: Transferable keyword extraction and generation with text-to-text language models. In: International Conference on Computational Science. pp. 398–405. Springer (2023)
16. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al.: Improving language understanding by generative pre-training (2018)
17. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research **21**(140), 1–67 (2020), http://jmlr.org/papers/v21/20-074.html
18. Ramachandran, R., Mohan, M.K., Sara, S.K.: Document clustering using keyword extraction. In: 2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT). pp. 01–06. IEEE (2022)
19. Real, L., Oshiro, M., Mafra, A.: B2w-reviews01 - a brazilian portuguese reviews corpus (2019), https://opencor.gitlab.io/corpora/real19b2wreviews01/
20. Roller, S., Dinan, E., Goyal, N., Ju, D., Williamson, M., Liu, Y., Xu, J., Ott, M., Shuster, K., Smith, E.M., et al.: Recipes for building an open-domain chatbot. arXiv preprint arXiv:2004.13637 (2020)
21. Rose, S., Engel, D., Cramer, N., Cowley, W.: Automatic keyword extraction from individual documents. Text mining: applications and theory pp. 1–20 (2010)
22. Yang, Z., Yu, H., Tang, J., Liu, H.: Toward keyword extraction in constrained information retrieval in vehicle social network. IEEE Transactions on Vehicular Technology **68**(5), 4285–4294 (2019)