








Expanding Queries with Maximum Likelihood Estimators and Language Models

Christos Karras¹(✉) , Aristeidis Karras¹ , Leonidas Theodorakopoulos² ,
Ioanna Giannoukou² , and Spyros Sioutas¹ 

¹ Computer Engineering and Informatics Department, University of Patras,
26504 Patras, Greece

{c.karras, akarras, sioutas}@ceid.upatras.gr

² Department of Management Science and Technology, University of Patras,
26334 Patras, Greece

{theodleo, igian}@upatras.gr

Abstract. The employment of various language modelling techniques in the area of information retrieval is gaining wide adoption in the state of the art methods. The precision of the language model enables the solution of the issue of information retrieval in a huge corpus of texts. To accomplish this, these techniques begin by estimating a probabilistic linguistic model for each article in the collection that is capable of generating a ranking of relevant texts in response to a query. One of the difficulties that this family of methods faces is a shortage of data. As a result, smoothing methods capable of changing the maximum likelihood estimator are required to account for the imprecision created. This paper highlights its use surpasses established approaches, such as tf-idf, for creating rankings of documents sorted by relevance. Finally, we examine various ideas related to query expansion by utilizing such methods.

Keywords: Expansion of queries · Language models · Cosine similarity · Maximum likelihood estimators · Decision making · Deep learning

1 Introduction

Throughout the years, query expansion strategies have been presented as a way to address the issue of term mismatches between a query and the documents it refers to. Typically, two kinds of query expansion technique families exist: local (based on Pseudo, Relevance, Indirect Feedback) and global (based on the creation and usage of a dictionary) [1]. This paper is focusing on the first category. Due to the difficulties of obtaining user input, only the first documents retrieved will be deemed significant. Pseudo-relevant documents are scanned for potential candidate phrases that may aid in expanding the query [2].

This strategy has been expanded upon inside the framework of the Language Model idea [3]. Statistical language models are frequently employed in information retrieval because they have a strong theoretical foundation and perform well empirically. The Robertson and Sparck Jones model [4] and the Croft and Harper model [5] are two well-known probabilistic techniques for information retrieval at the state of the art. Both approaches evaluate the likelihood that each page is relevant to a given query. Clearly, the two primary issues are related to accurately predicting both the query and document models. By predicting the factored form of the distribution $P(q, D)$, a Language Model determines the relevance of a document d to a query q [6].

The construction of a fine-tuned Language model must necessarily make use of smoothing models when one or more terms do not appear in a document. In the latter case, the maximum likelihood estimator would produce a probability equal to zero, invalidating the creation of the model itself [7, 8]. Another notion that is advantageous for widening the query and is extensively employed throughout the paper is that of *Word Embeddings*. The latter is gained via the usage of Language models, or more specifically through the co-occurrence of the words made accessible. This technique is predicated on the ability to map each word into a vector of real numbers contained inside a vector space. The goal is to be able to compare their distances in order to determine their similarity. If two words are similar, they are categorized as synonyms.

The remainder of the paper is organized as follows. In Sect. 3, the objectives of the paper will be introduced followed by a brief overview of the problem definition. Section 4, describes the implementation of the system in both theory and application level while Sect. 5 highlights the experimental results and their findings. Finally, the conclusions and future directions of this work are presented in Sect. 6.

2 Related Work

Information retrieval is a rapidly growing field inextricably linked with deep learning methods. Machine Learning models as in [9] are the foundation for extracting knowledge as well as events and patterns. Markov Chain Monte Carlo (MCMC) methods are on their turn employed in the knowledge representation field as they are stochastic processes that check the underlying distributions within a set of data [10, 11]. Additionally, heuristics are often utilized in gaining useful insights from complex geometric objects so as to extract features or information [12]. Smart IoT infrastructures as in [13] utilize innovative methods and emerging technologies for event detection purposes as well as for knowledge extraction in the field of smart agriculture supply chain.

3 Problem Definition

Language models have a wide range of applications, including voice recognition, spelling correction, grammar correction, and machine translation. All of these

applications are tasked with the responsibility of assigning a probability to a sequence of words based on their frequency of occurrence in one or more texts. As briefly described in the introductory section, the goal of the following paper is to prove the existence of another technique capable of extending a query that takes use of the idea of language model, especially an *n*-grams model. To accomplish this purpose, the mismatch between the words in the query and the terms in a corpus of documents must be resolved. The likelihood of creating a new q query given an estimate of the Language Model for an D document can only be determined by ranking relevant documents. When dealing with a big corpus of documents, creating n Language models with n equal to the number of documents proves to be a computationally difficult procedure.

This research paper utilizes a beneficial strategy to build a preliminary rating of documents sorted by relevance for a given query q . The *tf-idf* recovery model is being used in accordance with word weighting, which is extensively employed in informational retrieval area, leading to sufficient results. The inclusion of the *LM* and other semantic analysis approaches enabled us to surpass the *tf-idf* baseline, producing a ranking of texts, more relevant to a given query q [14].

4 Methodology

It should be emphasised that the ranking is generated using the weight vectors of the *tf-idf* technique utilising the well-known *cosine similarity* measure between the weight vectors. To meet the specified aim, the location of the relevant target document, that is, the document for which the user is seeking, was tracked. This was made feasible by selecting a query from the available ones that was similar to the title of this document. The score supplied to the target document will serve as the minimal *threshold* for constructing the new document ranking. This step is critical since specifying a higher threshold would result in the target document being omitted from future computations.

By lowering the threshold, however, documents that constitute noise will be considered. The *LMs* for each document will be generated based on this ranking, yielding n *LMs*, with n equal to the number of relevant documents based on the query keywords. It should be noted that this step was performed using the *skip-gram idea*, with step s equal to two. Each LM may be created only after using one of the available smoothing procedures. To avoid a linguistic model giving 0 probability to an unseen event, i.e. when a word in the query is not included in an LM, we should remove some probability mass from certain more common events and apply it to occurrences we have not seen before.

Smoothing may be accomplished in a number of ways, including Laplace (add-one) smoothing, Linear Interpolation smoothing [15], add-k smoothing [16], back-off smoothing [16], and Kneser-Ney smoothing [17]. Among them, the proposed method evaluates the usage of the first two smoothing approaches, each of which produces results that are effective in accomplishing the ultimate aim. The strength of the algorithm is in its ability to determine the optimal ranking of relevant pages using the initial query, iteratively, as s evolves.

This variation leads to the generation of several LMs, each with step s , with $s = \{2, 3, \dots, 10\}$. At each iteration, a new ranking of documents will be generated, thanks to the calculation of the Maximum Likelihood Estimation *MLE*, between the query q and each document d as shown in (1).

$$P(q|d) \approx P(q|M_d) \approx \prod_i^n P(w_i|w_{i-1}) \approx \frac{\text{count}(w_i, w_{i-1})}{\sum_{j=1}^n \text{count}(w_j, w_{i-1})} = \frac{\text{count}(w_i, w_{i-1})}{\text{count}(w_{i-1})} \quad (1)$$

After sorting all nine ranks, the one with the target document in the highest position, near to the top, will be selected. Additionally, this will allow for the determination of the λ parameters used in the interpolation smoothing process. By concentrating on the latter approach, it became important to execute the notion given by [7] about linear interpolation computation. Rather of adding one to the probability calculation, as the smoothing of Laplace does see Eq. (2), linear interpolation iteratively calculates the MLE of order two (*bi-grams*) see Eq. (3), all the way down to zero (*zero-grams*) see Eq. (4).

$$P(w_i|w_{i-1}) = \frac{\text{count}(w_{i-1}) + 1}{\text{count}(w_{i-1}) + |V|} \quad (2)$$

$$P(w_i|w_{i-1}) = \lambda P(q|M_d) + (1 - \lambda)P(q|M_c) \quad (3)$$

$$P(w_i) = \lambda \frac{1}{|V|} + (1 - \lambda)P(w_i) \quad (4)$$

where:

- $\sum_i \lambda_i = 1$
- M_d : represents the language model of the single document;
- M_c : represents the language model of the entire collection of documents;
- $|V|$: represents the number of unique words within the corpus of documents.

It is usually prudent to note that, similar to the iterative process of steps s , both λ parameters follow the same logic. The aim is to give both values a range of integers $\lambda = \{0.1, 0.2, \dots, 1\}$. The amount of *perplexity* between the query word set $W = \{w_1, w_2, \dots, w_N\}$ and the language model of the target page included in the ranking of relevant texts supplied by the tf-idf model will be computed in both smoothing procedures [18]. The calculation is shown in Eq. 5.

$$pp(W) = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i|w_{i-1})}} \quad (5)$$

After achieving the optimal ranking, the next step is to construct a *term-term matrix* [18], in which the terms include both those from the query and those from the language model of the whole collection of relevant texts included in the ranking. The co-occurrences of all words will be given in this matrix. On this sort of matrix, the computation of *Positive Pointwise Mutual Information (PPMI)* is feasible. PPMI is based on the assumption that the easiest method to assess the relationship between two words is to determine how much more often

the two words occur in our corpus than we would have expected them to occur randomly. This metric is derived from the conventional PMI, which is a measure of the frequency of two occurrences, x and y , in comparison to what we would anticipate if they were independent (see Eq. 6).

$$pmi(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)} \quad (6)$$

The ratio indicates how much more often the two terms appear than we would predict by chance. Positive, negative, or infinite PMI values are all possible. Negative values, which indicate that occurrences occur less often than we would anticipate by chance, are notoriously untrustworthy when dealing with documents that include few words, as we do here. To resolve this issue, the PPMI is calculated, which substitutes negative values with zero as shown in Eq. 7.

$$PPMI(x,y) = \max(\log_2 \frac{P(x,y)}{P(x)P(y)}, 0) \quad (7)$$

However, why is the PPMI computation used? This is advantageous for calculating the similarity of words, i.e. their synonymy, for searching for paraphrases, for tracking the change in meaning of words, and for automatically discovering the meanings of words in various datasets. The cosine similarity is performed on the first 10 word vectors with the greatest positive PPMI values to determine the most comparable terms to those in the query. By the end of the inquiry, each token will have a maximum of 10 expansion terms. As a result, we can immediately understand how query expansion may be accomplished. Moving on, there is one more issue to resolve before we can compute the similarity of the cosine: the *high dimensionality* of the matrix.

To achieve a high degree of similarity while maintaining a low computing cost, another idea has been implemented which is the *Singular Value Decomposition (SVD)* presented the notion of using SVD on a term-term matrix [19]. By switching from sparse to dense vectors, it is possible to do more accurate similarity comparisons. The SVD algorithm allows for the decomposition of the term-term matrix (A), with dimensions txd , into three matrices [20] as shown in Eq. 8.

$$A = USV^T \quad (8)$$

where:

- U : matrix of dimension txm where the columns represent the left singular vectors of matrix A;
- S : diagonal matrix of dimension $m \times m$, containing the singular values of matrix A;
- V^T : transposed matrix, of dimensions $m \times d$, where the columns represent the right singular vectors of matrix A.

By multiplying the matrix U by the matrix S , a new matrix \mathcal{D} of form as in 9, of dimension txm , is formed containing all the terms that will be compared

via cosine similarity with the query terms contained in the matrix $mathcal{T}$ as shown in Eq. 10 given by the product of S and V^T .

$$\mathcal{D} = U \times S \tag{9}$$

$$\mathcal{T} = S \times V^T \tag{10}$$

With all potential terms readily accessible for inclusion in the query tokens, we can now produce all conceivable queries. The number of queries generated is dependent on the number of words in the first query and any words received via the cosine similarity computation. With a maximum of ten words per token, the total number of produced queries is equal to the product of the number of possible pairings of terms and the number of possible pairs of terms as in 11.

$$\#Queries = \prod_i^q (\#sim_words_i) \tag{11}$$

where i denotes a single word in the query and q denotes the total number of words in the initial question. As new inquiries, they will follow the same method as the original query, i.e., new LMs will be generated, along with new rankings of documents that include the target document in the first locations. Finally, the best query, or queries, will be chosen based on the level of perplexity reached.

5 Experimental Results

The dataset used for the experiments is the famous *Recipes1M+* [21], a collection created by MIT, consisting of more than one million culinary recipes. Of all these recipes, only a subset of 51235 documents of it was used due to their informative content which best fits the purpose of this paper. According to the information about the line distributions for each recipe, the instruction field has a greater number than the ingredients field. To optimise our search performance, we used the ‘instructions’ field (Fig. 1). The first ranking of significant papers produced by the tf-idf algorithm was reviewed using two distinct methods that were most closely related to the notion described in [22].

To be more precise, each category in a recipe is treated as a distinct item connected with a document. Both techniques depend on the ability to match relevant documents’ categories to the category of the target document. Because the categories are not included in the dataset, the two suggested methodologies focus on “extracting” them straight from the web. The first solution makes use of an API called *Scrape Schema Recipe*, which scrapes the correct category using the link to the recipe website included in the dataset. Unfortunately, not all connections remain active, and hence the ranking is determined using three distinct methods:

1. **Overestimated:** consider the uncategorized document as good.
2. **Underestimated:** considers the uncategorized document not good.
3. **Discarded:** Discard the uncategorized document from the evaluation.

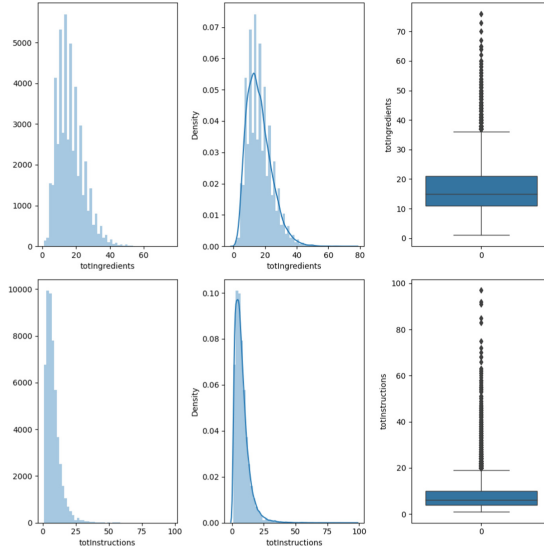


Fig. 1. Distributions of lines per ingredients and instructions.

As for the second method, this makes use of the *USDA* API, a library that has the task of taking every single ingredient of the recipe and fetching the corresponding category from a large database belonging to the United States Department of Agriculture. Compared to the first approach, the recipes that did not have a category are only four recipes. It is worth mentioning that both approaches are computationally expensive as the extraction of all categories took about two days.

In addition to all these evaluations, we decided to create a last one derived from a mixed approach (Scrape + USDA). Taking five random queries, the evaluations of the corresponding rankings are those reported in Table 1. As you can see, the mixed approach is the one that, in terms of average precision, manages to achieve the best evaluation. The performances, in terms of precision, recall and

Table 1. Average Precision on each query for each method.

Scrape schema recipe					
Queries	Overestimate	Underestimate	Discarded	USDA	Mixed (Scrape+USDA)
1	0.7562	0.3734	0.5962	0.9823	0.9953
2	0.9402	0.6792	0.9064	0.9974	0.9989
3	0.7542	0.2761	0.5429	0.9782	0.9986
4	0.8958	0.5334	0.8447	0.9528	0.9872
5	0.8346	0.5044	0.7558	0.9860	0.9948
Average	0.8379	0.4736	0.7290	0.9755	0.99487

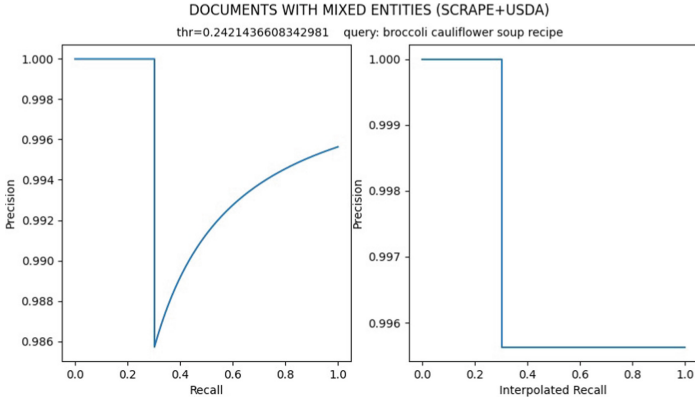


Fig. 2. Performance in terms of Precision, Recall and Interpolated Recall of mixed approach (Scrape+USDA)

interpolated recall, shown in (Fig. 2), demonstrating that the mixed approach is the best. Moving on to the evaluation of the new queries generated, a visual comparison, based on a PCA, was used between the ranking generated by the tf-idf method and each new query produced by the proposed method (Figs. 3, 4, 5, 6, 7).

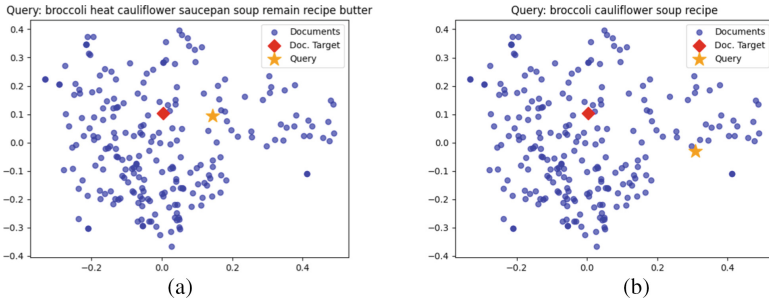


Fig. 3. Query (a) broccoli heat cauliflower saucepan soup remain recipe butter (b) broccoli cauliflower soup recipe

The goal is to find a query whose distance from the target document is less than all other distances produced by the remaining queries. As we can see, compared to the position of the original query, a query with a smaller distance has always been found. For each new query generated, the perplexity is calculated as the skip-grams step varies, with the language model of the target document. After various tests, with a higher number of queries, it can be stated that the skip-gram step and the perplexity are two inversely proportional measures, that is, as the step s increases, the perplexity p decreases. The same behavior occurs

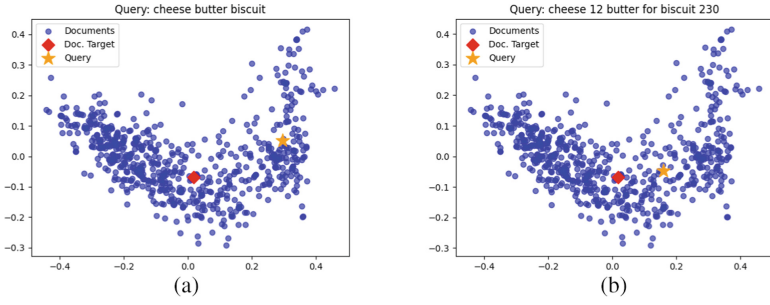


Fig. 4. Query (a) cheese butter biscuit (b) cheese 12 buffer for biscuit 230

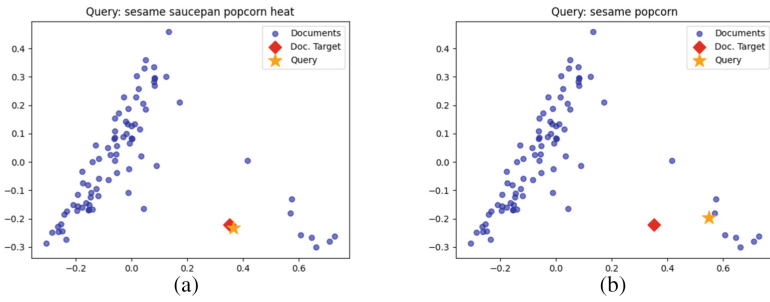


Fig. 5. Query (a) sesame saucepan popcorn heat (b) sesame popcorn

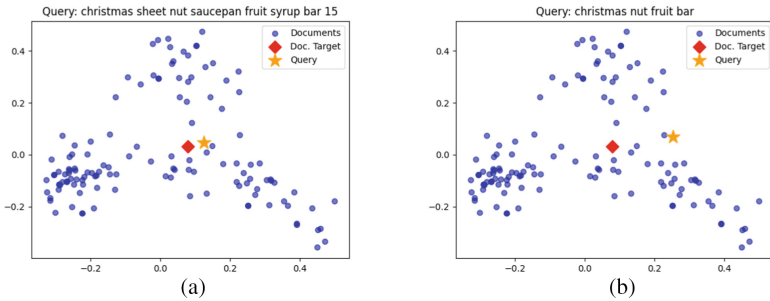


Fig. 6. Query (a) Christmas sheet nut saucepan fruit syrup bar 15 (b) Christmas nut fruit bar

with the λ_1 and λ_2 parameters present in the interpolated smoothing. When λ_1 is less than λ_2 , perplexity always tends to decrease (Fig. 8). For the evaluation of the system four distinct libraries capable of tokenizing documents and queries were utilized: **Spacy**, **Gensim**, **Nltk**, and **Keras**. Each of them generates results that vary in terms of perplexity, the number of queries produced, calculation time (Table 3), and ranking (Table 2). As shown by the results, each strategy offers

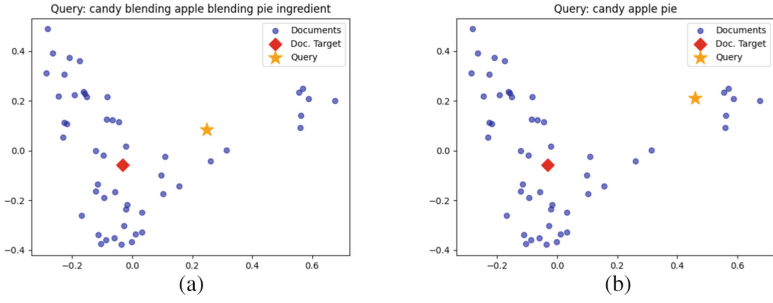


Fig. 7. Query (a) candy blending apple blending pie ingredient (b) candy apple pie

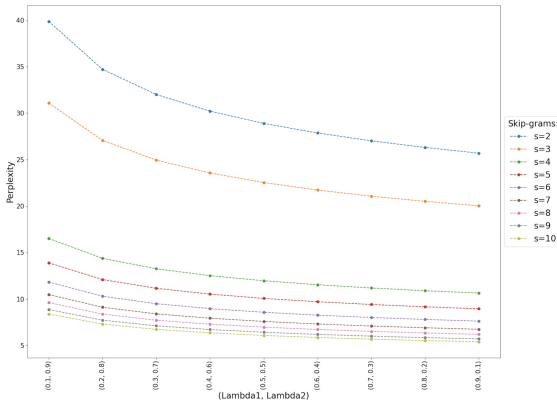


Fig. 8. Variation of the perplexity value based on the change of the step s and of the λ_1 and λ_2 values of interpolated smoothing.

Table 2. Target document position in the ranking of relevant documents. (T: *tf-idf*, P: *Proposed*)

Method	1		2		3		4		5	
	T	P	T	P	T	P	T	P	T	P
Keras	229	0	627	0	86	0	129	0	56	0
Nltk	229	0	627	0	86	0	129	0	55	0
Spacy	223	0	651	0	85	0	144	0	79	0
Gensim	241	0	656	37	88	0	117	0	42	0

distinct advantages. What each strategy has in common is that the target page is ranked first.

Looking at the results obtained with an additional five queries Table 4, for a total of ten queries, we can see that the best results come from Gensim. This is therefore able to find a greater number of queries that have less perplexity

Table 3. Performance in terms of number of queries (#Queries) and computation time.

Method	#Queries	Time
Keras	15.500	<i>Low</i>
Nltk	15.500	<i>Low</i>
Spacy	20.400	<i>High</i>
Gensim	22.090	<i>Medium</i>

Table 4. Euclidean Distance (D) between a query and the target document at a specific perplexity (P).

Queries	Keras		Nltk		Spacy		Gensim	
	D	P	D	P	D	P	D	P
1	0.15	16.42	0.15	16.58	0.11	34.87	0.13	90.16
2	227	17.91	0.11	22.37	0.21	11.56	0.15	53.44
3	0.03	05.42	0.022	05.41	0.07	05.15	0.01	05.03
4	0.04	12.39	0.05	12.45	0.09	12.44	0.01	10.83
5	0.35	23.04	0.31	23.22	0.43	42.25	0.27	10.36
6	0.24	06.39	0.22	06.39	0.24	03.93	0.21	04.02
7	0.13	1429.33	0.12	1521.33	0.24	06.03	0.12	03.43
8	0.38	13.87	0.39	13.81	0.46	13.65	0.27	08.11
9	0.57	05.31	0.56	05.31	0.61	04.76	0.41	03.92
10	0.01	04.27	0.01	04.29	0.06	05.12	0.01	04.28

with the target document than the other methods. To verify the veracity of the results, the Euclidean distance between the new query and the target document was calculated. As we can observe, a smaller distance implies a lower perplexity between the query and the target document. As for the other systems, Keras and Nltk obtain similar results, unlike the latter producing queries with the highest perplexity. As for Spacy, this produces queries with the greatest distance from the target document and in fact, comparing with the results obtained with Gensim, we can see that as the distance increases, the perplexity of each specific queries generated by Spacy also increases.

6 Conclusions and Future Work

In conclusion, it can be said that new queries are generated appropriately and that Gensim is the system that achieves the best performance. However, other things need improvement, such as defining a way for establishing a suitable

threshold for creating each ranking of relevant documents. Throughout the research, the thresholds supplied by the sklearn library's precision recall curve function were considered. Regrettably, not all of them had a sufficient score to examine the target document. As a result, it was decided to take into account the initial threshold, referred to as the weight supplied to the target document using the cosine similarity approach.

This paper could be extended in various ways. Future directions of this work include improvements in search performance, where an ad hoc barrier can be established to prevent the ranking of relevant documents from including all of the collection of documents. Another enhancement is the decrease of the term matrix (with stochastic gradient descent). Concerning the usage of the various parsers, it would be interesting to design a hybrid system capable of producing a large number of questions in a short period of time. Moreover, it would be interesting to provide a mechanism capable of extending the initial query with words from a vocabulary such as Wordnet.

References

1. Cummins, R., O'Riordan, C.: Evolving co-occurrence based query expansion schemes in information retrieval using genetic programming. In: AICS 2005, p. 137 (2005)
2. Lafferty, J., Zhai, C.: Document language models, query models, and risk minimization for information retrieval. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2001, pp. 111–119. ACM, New York (2001)
3. Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1998, pp. 275–281. Association for Computing Machinery, New York (1998)
4. Robertson, S.E., Jones, K.S.: Relevance weighting of search terms. *J. Am. Soc. Inf. Sci.* **27**(3), 129–146 (1976)
5. Croft, W.B., Harper, D.J.: Using probabilistic models of document retrieval without relevance information. *J. Doc.* (1979)
6. Zaragoza, H., Hiemstra, D., Tipping, M.: Bayesian extension to the language model for ad hoc information retrieval. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2003, pp. 4–9. Association for Computing Machinery, New York (2003)
7. Gutkin, A.: Log-linear interpolation of language models. Mémoire de DEA, University of Cambridge (2000)
8. Chen, S.F., Goodman, J.: An empirical study of smoothing techniques for language modeling. *Comput. Speech Lang.* **13**(4), 359–394 (1999)
9. Karras, C., Karras, A., Sioutas, S.: Pattern recognition and event detection on IoT data-streams. arXiv preprint [arXiv:2203.01114](https://arxiv.org/abs/2203.01114) (2022)
10. Karras, C., Karras, A., Avlonitis, M., Sioutas, S.: An overview of MCMC methods: from theory to applications. In: Maglogiannis, I., Iliadis, L., Macintyre, J., Cortez, P. (eds.) AIAI 2022. IFIPAICT, pp. 319–332. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-08341-9_26

11. Karras, C., Karras, A., Avlonitis, M., Giannoukou, I., Sioutas, S.: Maximum likelihood estimators on MCMC sampling algorithms for decision making. In: Maglogiannis, I., Iliadis, L., Macintyre, J., Cortez, P. (eds.) AIAI 2022. IFIPAICT, pp. 345–356. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-08341-9_28
12. Karras, C., Karras, A.: DBSOP: an efficient heuristic for speedy MCMC sampling on polytopes. arXiv preprint [arXiv:2203.10916](https://arxiv.org/abs/2203.10916) (2022)
13. Karras, A., Karras, C., Drakopoulos, G., Tsohis, D., Mylonas, P., Sioutas, S.: SAF: a peer to peer IoT LoRa system for smart supply chain in agriculture. In: Maglogiannis, I., Iliadis, L., Macintyre, J., Cortez, P. (eds.) AIAI 2022. IFIPAICT, pp. 41–50. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-08337-2_4
14. Lavrenko, V., Croft, W.B.: Relevance models in information retrieval. In: Croft, W.B., Lafferty, J. (eds.) Language Modeling for Information Retrieval. INRE, pp. 11–56. Springer, Dordrecht (2003). https://doi.org/10.1007/978-94-017-0171-6_2
15. Blu, T., Thevenaz, P., Unser, M.: Linear interpolation revitalized. *IEEE Trans. Image Process.* **13**(5), 710–719 (2004)
16. Falahatgar, M., Ohannessian, M.I., Orlitsky, A.: Near-optimal smoothing of structured conditional probability matrices. In: *Advances in Neural Information Processing Systems*, vol. 29 (2016)
17. Song, M., Yoo, C.D.: Multimodal representation: Kneser-ney smoothing/skip-gram based neural language model. In: 2016 IEEE International Conference on Image Processing (ICIP), pp. 2281–2285. IEEE (2016)
18. Jurafsky, D., Martin, J.H.: *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, vol. 3 (2008)
19. Schütze, H.: Dimensions of meaning. In: *Proceedings of the 1992 ACM/IEEE Conference on Supercomputing, Supercomputing 1992*, pp. 787–796. IEEE Computer Society Press, Washington (1992)
20. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. *J. Am. Soc. Inf. Sci.* **41**, 391–407 (1990)
21. Marin, J., et al.: Recipe1M+: a dataset for learning cross-modal embeddings for cooking recipes and food images. *IEEE Trans. Pattern Anal. Mach. Intell.* **43**, 187–203 (2019)
22. Parvez, M.R., Chakraborty, S., Ray, B., Chang, K.W.: Building language models for text with named entities. arXiv preprint [arXiv:1805.04836](https://arxiv.org/abs/1805.04836) (2018)