Query expansion using Language Models

Flavio Forenza

September 11, 2021

Abstract

The use of different methods of language modeling, within the field of information retrieval, is finding a wide diffusion in the state of the art. Based on the accuracy of the language model, the problem related to the information retrieval, in a large corpus of documents, can be solved. In order to do this, the basic idea of these approaches is to estimate a probabilistic linguistic model, for each document in the collection, which is able to generate a ranking of relevant documents given a query. One of the problems that afflicts this family of methods is due to the lack of data present. From this, it is necessary to apply smoothing techniques capable of adjusting the maximum likelihood estimator in order to correct the generated imprecision. This paper shows how their application outperforms the performance of classic methods, such as *tf-idf*, useful for generating rankings of documents ordered by relevance. Using them, we will look at some concepts that are useful for query expansion.

Introduction

Over the years, query expansion techniques have been proposed as a solution to the problem of term mismatches between a query and its relevant documents. There are typically two types of query expansion method families: Local (based on Pseudo / Relevance / Indirect Feedback) and Global (based on the generation and use of a thesaurus) [9]. This paper focuses on the first category. Given the difficulty in gathering the users' feedback, only the

first documents recovered will be considered relevant. Pseudo-relevant documents are used to find possible candidate terms to help expand the query [6]. This method has been further developed within the concept of the Language Model [8]. Statistical language models are widely used within Information Retrieval as they have a solid theoretical background and good empirical performance. At the state of the art, two well known probabilistic approaches in infromation retrieval are the Robertson and Sparck Jones model [13] and the Croft and Harper model [16]. Both models estimate the probability of relevance of each document to the query. Clearly, the two main problems relate to correctly estimating both the query model and the document model. A Language Model calculates the relevance of a document d to a query q by estimating a factored form of the distribution $P(q, D)$ [5]. The construction of a good 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 [4] [14]. Another concept, useful for expanding the query and widely used within the project, is that of *Word Embeddings*. The latter is obtained precisely from the use of Language models, or rather thanks to the co-occurrence of the terms made available. This method is based on being able to map every single word into a vector of real numbers, within a vector space. The idea is to be able to compare the distance of these in order to understand their similarity relationship. If one word is similar to another, then these will be considered as synonyms. The remainder of the paper is organized as follows. In the next section, *Research question and Methodology*, the objectives of the project will be introduced followed by an overview of the proposed approach. The third section, *Experimental result*, describes the whole system and the results obtained. Finally, the conclusions are presented in section four, *Concluding remarks*.

Research question and Methodology

Language models can be used for a variety of purposes, such as: speech recognition, spelling correction, grammar correction and automatic translation. All these applications have the task of assigning a probability to a sequence of words, based on the number of times they appear in one or more documents. As briefly mentioned in the introductory chapter, the purpose of the following project is to verify the existence of a further method, which

makes use of the concept of language model, specifically an *n-grams* model, capable of expanding a query. Achieving this goal means solving the problem of mismatch between the terms present in the query and those present in a corpus of documents. The probability of generating a new q query given the estimate of a Language Model for a D document can only occur through a ranking of relevant documents. If the corpus of documents is large, thinking of generating n Language models, with n equal to the number of documents, turns out to be a computationally expensive operation. This paper has used a useful approach to be able to generate a first ranking of documents ordered by relevance with the query q. The technique applied is the *tf-idf* recovery model. The adoption of this method of weighing the terms, as well as being widely used in the state of the art, produces excellent results. The introduction of the *LM*, and of other semantic analysis techniques, made it possible to outperform the performance of the *tf-idf* baseline, generating ranking, starting from the latter, of documents more pertinent to the query q [15]. In the following paper, tests are carried out which certify the veracity of this thesis. It should be noted that the generation of the ranking, obtained from the weights of the *tf-idf* method, is obtained using the well-known *cosine similarity* metric between the weight vectors. To comply with the set objective, the position of the relevant target document, that is the document that the user is searching for, was kept track. This was possible because a query was chosen, among those available, that was close to the title of this document. The score assigned to the target document will represent the minimum *threshold* to be able to form the new ranking of documents. This step is fundamental as, by setting a higher threshold, the target document would be lost in subsequent calculations. By setting a lower threshold, however, those documents that represent noise will be taken into account. From this ranking, the *LMs* for each document will be calculated [2], generating n LMs, with n equal to the number of relevant documents, based on the terms in the query. It should be noted that, in order to carry out this step, the concept of *skip-gram* has been applied to the query, with step s equal to two. The creation of each LM is possible only after using one of the existing smoothing techniques. To prevent a linguistic model from assigning zero probability to an invisible event, i.e. when a term present in the query is not present in an LM, we should eliminate some probability mass from some more frequent events and give it to events that we haven't never seen. There are a variety of methods for smoothing, some of these are: *Laplace (add- one) smoothing*, *Linear Interpolation smoothing*, *add-k smoothing*, *back-off smoothing* and *Kneser-Ney*

smoothing. Among these, the following paper has experimented the use of the first two smoothing methods, each of which will produce different results useful for achieving the final goal. The core of the algorithm lies in being able to derive the best ranking of relevant documents, using the initial query, through an iterative process, as s changes. This variation will lead to the generation of several LMs, each with step s, with $s = \{2, 3, \ldots, 10\}$. 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 $d(1)$.

$$\begin{aligned} P(q|d) &\approx P(q|M_d) \\ &\approx \prod_i^n P(w_i|w_{i-1}) \\ &\approx \frac{count(w_i, w_{i-1})}{\sum_{j=1}^n count(w_j, w_{i-1})} \\ &= \frac{count(w_i, w_{i-1})}{count(w_{i-1})} \end{aligned} \tag{1}$$

Among all these nine rankings, after being sorted, the one that will have the target document in the highest position, close to the first, will be chosen. Also thanks to this, it will be possible to determine the lambda parameters present in the interpolation smoothing technique. By focusing on the latter technique, it was necessary to implement the concept suggested by [1] as regards the calculation of linear interpolation. Instead of adding one to the probability calculation, as is well known in Laplace's smoothing (2), linear interpolation, recursively, calculates the MLE of order two (*bi-grams*) (3), up to the order zero (*zero-grams*)(4).

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

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

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

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 always good to specify that, like the iterative process of steps s, both λ parameters also follow the same reasoning. The idea is to assign a range of numbers $\lambda = \{0.1, 0.2, \ldots, 1\}$ to both values. In both smoothing techniques, the *perplexity* level existing between the query word set $W = \{w_1, w_2, \ldots, w_N\}$ and the language model of the target document present in the ranking of relevant documents returned by the tf-idf model will be calculated(4) [3].

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

After obtaining the best ranking, the next step is based on building a *term-term matrix* [3], where the terms in question are both those of the query and those belonging to the language model of the entire collection of relevant documents present in the ranking. Within this matrix, the numbers of co-occurrences between all terms will be reported. On this type of matrix it is possible to apply the calculation of *Positive Pointwise Mutual Information (PPMI)*. PPMI draws on the intuition that the best way to weigh the association between two words is to ask how much more the two words co-occur in our corpus than we would have a priori expected them to appear by chance. This measure derives from the calculation of the standard PMI which represents a measure of frequency between two events x and y, compared to what we would expect if they were independent (6).

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

The ratio gives us an estimate of how much more the two words co-occur than we expect by chance. PMI values can be positive, negative or infinite. Negative values, which imply that events occur less often than we would expect by chance, tend to be unreliable when we have documents consisting

of few terms, as in our case. To solve this problem, the calculation of the PPMI is used which replaces negative values with zero(7).

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

But the question is: why is the PPMI calculation used? The adoption of this is useful for being able to calculate the similarity between words, i.e. their synonymy, search for paraphrases, keep track of the change in meaning of words and to automatically discover the meanings of words in different corpora. To find the words most similar to those in the query, the cosine similarity is calculated on the first ten word vectors that have the highest positive PPMI values. Eventually, each token in the query will have a maximum of ten expansion terms. Thanks to this, we are already seeing how query expansion can be done. Moving on, there is a problem to solve before we can calculate the similarity of the cosine: the *sparsity* of the matrix. In order to obtain a good similarity, relatively low at the computational level, another concept has been implemented: *Singular Value Decomposition (SVD).* The idea of applying SVD on a term-term matrix was proposed by [11]. Switching from sparse to dense vectors allows for better similarity comparisons. The SVD allows to decompose the term-term matrix (A), of dimensions txd, into three matrices [12] (8):

$$A = USV^t \tag{8}$$

Where:

1. U: matrix of dimension txm where the columns represent the left singular vectors of matrix A;

2. S: diagonal matrix of dimension mxm, containing the singular values of matrix A;

3. V^T: transposed matrix, of dimensions mxd, where the columns represent the right singular vectors of matrix A.

Thanks to the product of the matrix U with the matrix S, a new matrix $\mathcal{D}$ (9), of dimension txm, is formed containing all the terms that will be compared, through cosine similarity, with the query terms present in the matrix $\mathcal{T}$ (10) given by the product of S and V^T .

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

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

Having all the possible words available to be added to the query tokens, the time has come to generate all the possible queries. The number of queries produced will depend on the number of words present in the initial query and on any words obtained from the calculation of the cosine similarity. Having set a maximum limit of ten words per token, the total of the quries generated is given by the product (11) of the number of possible pairs of terms:

$$\#Queries = \prod_{i}^{q} (\#new\_tokens_i) \tag{11}$$

where i represents the single word taken in the query and q represents the total of the words present in the starting query. Being new queries, these will be subjected to the same procedure as the original query, that is, new LMs will be produced and with them new rankings of documents that will contain the target document in the first positions. Finally, the best query, or queries, will be chosen based on the level of perplexity reached.

Experimental Results

The dataset used for the experiments is the famous *Recipes1M+* [7], 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 study. The information about the line distributions for each recipe indicates that the instruction field contains a higher number than the information contained in the ingredients field. To have good performance in finding, we preferred to choose the "instructions" field (Fig. 1). The first ranking of relevant documents, generated by the tf-idf method, was evaluated following two different approaches that could best be linked with the principle explained in [10]. Specifically, each category of a recipe is considered as an entity associated with a document. Both approaches rely on being able to compare the categories of relevant documents with the category of the target document. Since the categories are not present in the dataset, the two proposed approaches deal with being able to "extract" these categories directly from the web. The first approach uses an API, called *Scrape Schema Recipe*, which takes the link associated with the recipe website, present in the dataset, and scrapes the

corresponding category. Unfortunately not all the links are still existing, therefore the evaluation of the ranking takes place following three types of methods:

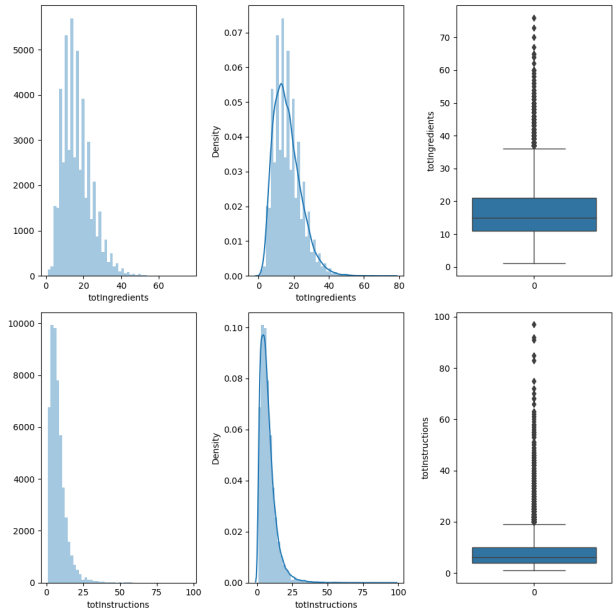

Figure 1. Distributions of lines per ingredients and instructions.

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.

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, it was 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

Queries	Scrape Schema Recipe			USDA	Mixed (Scrape+USDA)
	Overestimate	Underestimate	Discarded		
I	0.7520	0.3734	0.5958	0.9817	0.9947
II	0.9309	0.6780	0.9054	1.0	1.0
III	0.7458	0.2746	0.5411	0.9797	0.9982
IV	0.8939	0.5320	0.8433	0.9612	0.9870
V	0.8335	0.5033	0.7544	0.9940	0.9940
Average	0.8312	0.4722	0.7280	0.9833	**0.99478**

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

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 interpolated recall, visible in (Fig. 2), demonstrate that the mixed approach is the best. Moving on

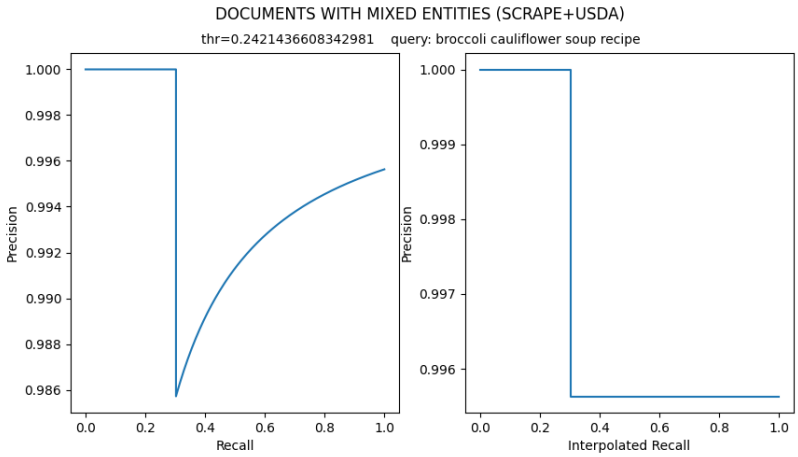

Figure 2. Performance in terms of Precion, Recall and Interpolated Recall of mixed approach (Scrape+USDA)

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 project core (Fig. 3). 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.

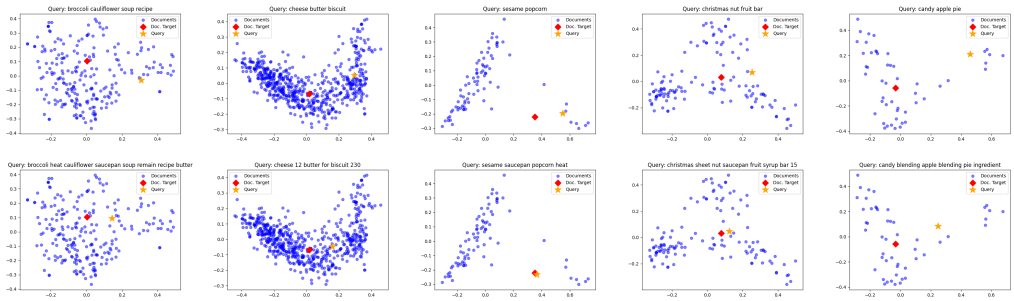

Figure 3. Distance within the PCA between the original query (top) and the expanded query (bottom).

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 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. 4). We

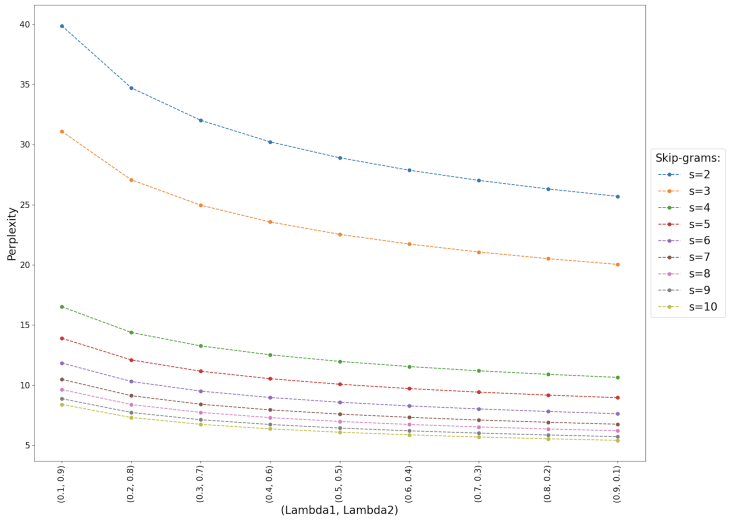

Figure 4. Variation of the perplexity value based on the change of the step s and of the λ_1 and λ_2 values of interpolated smoothing.

have come to the point of asking: How can the whole system be evaluated? To answer this question, four different libraries able to tokenize documents and queries were used: **Spacy**, **Gensim**, **Nltk** and **Keras**. Each of these

produces different results in terms of perplexity, number of queries generated, sparsity index (Tab. 3) and rankings(Tab. 2). As can be seen from the data, each method has a different benefit. What each method has in common is the positioning of the target document in the first position of the ranking.

Method	**I**		**II**		**III**		**IV**		**V**	
	T	**P**	**T**	**P**	**T**	**P**	**T**	**P**	**T**	**P**
Keras	228	0	623	0	82	0	126	0	51	0
Nltk	228	0	623	0	82	0	126	0	51	0
Spacy	221	0	647	0	81	0	141	0	72	0
Gensim	240	0	652	37	84	0	113	0	37	0

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

Method	**#Queries**	**Avg. Sparsity Reduction**	**Time**
Keras	14.900	0.721	***Low***
Nltk	14.900	0.717	***Low***
Spacy	19.010	**0.741**	*High*
Gensim	**22.090**	0.730	*Medium*

Table 3: Performance in terms of number of queries (#Queries), Average Sparsity reduction with the SVD method and computation time.

Looking at the results obtained with an additional five queries (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 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 you can see, 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.

Queries	Keras		Nltk		Spacy		Gensim	
	D	P	D	P	D	P	D	P
I	0.14	16.41	0.14	16.59	0.12	34.89	0.14	90.17
II	228	17.91	0.11	22.37	0.2	11.57	0.15	53.44
III	0.02	5.41	0.02	5.41	0.07	5.15	0.01	5.03
IV	0.05	12.39	0.05	12.45	0.09	12.44	0.01	10.83
V	0.31	23.04	0.31	23.22	0.4	42.25	0.27	10.36
VI	0.22	6.39	0.22	6.39	0.24	3.93	0.21	4.02
VII	0.12	1429.33	0.12	1521.33	0.24	6.03	0.1	3.43
VIII	0.39	13.8	0.39	13.81	0.46	13.65	0.27	8.11
IX	0.56	5.3	0.56	5.3	0.6	4.7	0.41	3.9
X	0	4.29	0	4.29	0.06	5.12	0	4.28

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

Concluding Remarks

Coming to the conclusions, we can say that the generation of new queries happens correctly and that the system that obtains the best performance is Gensim. However, there are some points that need to be improved, such as determining a method that sets a threshold useful for generating each ranking of relevant documents. During the project, the thresholds returned by the '*precision_recall_curve*' method of the sklearn library were taken into consideration. Unfortunately, not all of these had a useful score to be able to take into consideration the target document. It was therefore decided to take into consideration the initial threshold, referred to the weight, which the method of cosine similarity assigned to the target document. In addition, to speed up searches, an ad hoc threshold was set to prevent the ranking of relevant documents from containing all the documents in the collection. As for the use of the different parsers, as a future development it would be interesting to develop a hybrid system, capable of generating a high number of queries with low perplexity with the target document. In addition to this, it would be interesting to adopt a method capable of expanding the original query with the terms present in a vocabulary such as Wordnet instead of using the terms already present in the entire corpus of documents.

References

[1] A.Gutkin. *Log-Linear Interpolation of Language Models*. PhD thesis, 11 2000.

[2] C.D.Manning, P.Raghavan, and H.Schütze. *Introduction to Information Retrieval*, chapter 12. Cambridge University Press, 2008.

[3] D.Jurafsky and J.H.. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, volume 3. 2008.

[4] G.Alexander. *Log-Linear Interpolation of Language Models*. PhD thesis, 2000.

[5] H.Zaragoza, D.Hiemstra, and M.Tipping. Bayesian extension to the language model for ad hoc information retrieval. page 4–9, 2003.

[6] J.Lafferty and C.Zhai. Document language models, query models, and risk minimization for information retrieval. page 111–119, 2001.

[7] J.Marin, A.Biswas, F.Ofli, N.Hynes, A.Salvador, Y.Aytar, I.Weber, and A.Torralba. Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019.

[8] J.M.Ponte and W.B.Croft. A language modeling approach to information retrieval. page 275–281, 1998.

[9] R.Cummins and C.O'Riordan. Evolving co-occurrence based query expansion schemes in information retrieval using genetic programming. pages 137–146, 2005.

[10] M. R.Parvez, S.Chakraborty, B.Ray, and K-W.Chang. Building language models for text with named entities. pages 373–2383, 2018.

[11] H. Schütze. Dimensions of meaning. In *Supercomputing '92*, 1992.

[12] S.Deerwester, S.T.Dumais, G.W.Furnas, T.K.Landauer, and R.Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science 41*, pages 391–407, 1990.

[13] S.E.Robertson and K.Spärck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, page 129—46, 1976.

[14] S.F.Chen and J.Goodman. An empirical study of smoothing techniques for language modeling. page 310–318, 1996.

[15] V.Lavrenko and W.B.Croft. *Relevance Models in Information Retrieval*, pages 11–56. Springer Netherlands, 2003.

[16] W.B.Croft and D.J.Harper. Using probabilistic models of document retrieval without relevance information. *Journal of Documentation*, pages 285–295, 1979.