Journal of Information Technology Research
Volume 15 • Issue 1

Finding Relevant Documents in a Search Engine Using N-Grams Model and Reinforcement Learning

Amine El Hadi, Faculty of Sciences and Technics, Sultan Moulay Slimane University, Morocco*
Youness Madani, Faculty of Sciences and Technics, Sultan Moulay Slimane University, Morocco
https://orcid.org/0000-0001-9363-7147
Rachid El Ayachi, Laboratory TIAD, Faculty of Sciences and Technology, Sultan Moulay Slimane University, Morocco
Mohamed Erritali, Laboratory TIAD, Faculty of Sciences and Technology, Sultan Moulay Slimane University, Morocco
https://orcid.org/0000-0002-1672-8085

ABSTRACT

The field of information retrieval (IR) is an important area in computer science. This domain helps us to find information that we are interested in from an important volume of information. A search engine is the best example of the application of information retrieval to get the most relevant results. In this paper, the authors propose a new recommendation approach for recommending relevant documents to a search engine's users. In this work, they proposed a new approach for calculating the similarity between a user query and a list of documents in a search engine. The proposed method uses a new reinforcement learning algorithm based on n-grams model (i.e., a sub-sequence of n constructed elements from a given sequence) and a similarity measure. Results show that the method outperforms some methods from the literature with a high value of accuracy.

KEYWORDS

N-Grams Model, Query Reformulation, Reinforcement Learning, Search Engine, Semantic Similarity

1. INTRODUCTION

Since the most recent decade, search engines have become an essential tool in our regular daily life. At the point when we search for an information, we frequently go to our preferred search engine and look at the returned pages, users of web search tools calmly hope to get precise and close prompt responses to questions and demands, only by entering a short query — a couple of words — into a text box and tapping on a search button. The system of these search engines return a list of web pages containing the terms in the query, this list is constructed by identify eliminate duplicate and redundant pages, compute a score of each page to rank the list of web pages, then return the top results of the list, but these results do not always return the desired information. This is a major issue in

DOI: 10.4018/JITR.299930 *Corresponding Author

information retrieval. Information retrieval researchers have focused on a few key issues, one of these issues is relevance. A document is considered relevant to a given query if its contents (completely or partially) satisfy the information need represented by the query, though this sounds simple, there are many factors that come in a person's decision as to whether a specific document is relevant. These factors should be taken into consideration when designing algorithms for comparing text and ranking documents to get the best result in a search engine. There are many fields that help us to get relevant documents; one of them is query reformulation. Users of search engines usually have interaction in a process of query formulation and reformulation if they are unsatisfied with the results from the initial query so as to meet their information desires. An initial query is given and then updated in light of the results obtained until the user get a set of relevant documents and get satisfied. This process is known as query reformulation. There are a minimum of 2 reasons why query reformulation occurs.

Firstly, the user could have a quite specific information would like in mind but is unsure how to express that require in the query language. Secondly, the user's information want might alter as a consequence of examining the search results. For example a user search for the query "quilting" then get a list of quilting stores, then decide to update their information to get stores in their location. Text similarity measures play associate degree progressively important role in text connected research and applications in tasks like information retrieval, text classification, document cluster, topic detection, topic following, questions generation, question answering, essay scoring, short answer scoring, machine translation, text report and others. Finding similarity between words may be a fundamental part of text similarity that is then used as a primary stage for sentence, paragraph and document similarities. Words will be similar in 2 ways in which lexically and semantically. Words are similar lexically if they have the same character sequence. Words are similar semantically if they have the same thing, same means, used in the same way, and one is a type of the other.

In this paper, we propose a new approach to improve the results of a search engine based on reinforcement learning. Reinforcement learning (RL) (Sutton & Barto, 2013) is a learning technique in which an agent learns from the interactions with the environment by trial and-error, Reinforcement learning problems involve learning what to do—how to map situations to actions—so as to maximize a numerical reward signal. In an essential way, they are closed-loop problems because the learning system's actions influence its later inputs. Moreover, the user is not told which actions to take, as in many forms of machine learning, but instead must discover which actions yield the most reward by trying them out. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards. These three characteristics—being closed-loop in an essential way, not having direct instructions as to what actions to take, and where the consequences of actions, including reward signals, play out over extended time periods—are the three most important distinguishing features of reinforcement learning problems.

The action in our system corresponds to changing the n in n-gram for calculating the similarity between a query and documents in our corpus, and the reward is the accuracy of the result obtained, all that is with the use of the semantic similarity and the N-grams model.

Our work will be divided into parts, the first concerning the Natural Language Processing (NLP) Methods, by applying different text preprocessing methods, such as removing stop words from documents, and get the stem of words, get the n-grams for documents, after the preprocessing stage we will calculate the semantic similarity between documents in our corpus and query to get in the last step the most relevant documents. In the step of calculation of semantic similarity, we apply the RL algorithm to find the optimal path the user must follow for finding an optimal result.

The rest of this paper is organized as follows: Section 2 presents a literature review, in Section 3 we will describe our Research methodology (the different text preprocessing methods exist in the literature, how we divide documents into n-grams, the calculation of the semantic similarity between documents using reinforcement learning and finally how we obtain the relevant documents). Section 4 presents the experimental results and finally, in Section 5 the conclusion and the perspectives.

2. LITERATURE REVIEW

Several definitions of information retrieval have been defined describe searching as a process of storage, organization, representation, and access to information by the user.

To provide the right results to users, many studies have been presented on measuring document similarity in recent years for facilitating the search for information in complex information systems such as a search engine.

Authors in (Cho, Lebanoff, Foroosh & Liu, 2019) proposed a new approach inspired by the determinantal point process for multi-document summarization. This approach includes a diversity measure assessing the redundancy between sentences, and a quality measure that indicates the importance of sentences. DPP extracts a set of summary sentences that are both representatives of the document set and remain diverse; they measure redundancy between a pair of sentences based on surface form and semantic information. Authors exploit capsule networks for determining sentence similarity for summarization purpose and shows the performance of this approach. Researchers in the article presented in (Gao, Zhang & Chen, 2015) presented a new approach for semantic similarity measure which is based on edge-counting and information content theory, the proposed measure nonlinearly transforms the weighted shortest path length between the compared concepts to achieve the semantic similarity results. The method is subcategorized into three strategies: strategy 1 weights all the edges along the shortest path between the compared words, strategy 2 uses depth of words to weight the shortest path length, and strategy 3 utilizes IC of words to weight the shortest path length. The Experimental Results show the efficiency of this method compared to some methods from the literature. Authors of (Cai, Zhang, Lu & Che, 2018) present a hybrid WordNet-based approach for concept semantic similarity measurement based on the information content-weighted shortest path to measuring concept semantic similarity, which leverages the information content(IC) of concepts to weight the shortest path distance between concepts. Authors of this paper develop an intrinsic IC model by taking various semantic properties and structures of WordNet into consideration to improve the performance of IC computation. Finally, they summarize and classify the technical characteristics of previous WordNet-based approaches. The Experimental Results show that the results of the proposed hybrid method based on IC-weighted path distance in WordNet are more correlated with the human judgment of similarity in terms of the correlation coefficient.

Many studies have been focused on query reformulation in recent years for facilitating the search for information in complex information systems such as a search engine to get the best results.

Nogueira and Cho in (Nogueira & Cho, 2017) present a query reformulation approach based on a neural network that rewrites a query to boost the number of relevant documents returned in the search engine. The authors train the neural network with reinforcement learning. The actions relate to choosing terms to get a new reformulated query, and the reward is the recall of the document. The results show that the approach of the reformulation works better than using the original query alone to get the relevant documents, and they found that the supervised methods have a better performance than the unsupervised ones, but perform worse than Reinforcement Learning-based models.

Narasimhan et al. (Narasimhan, Karthik, Yala & Barzilay, 2016) used a reinforcement learning-based approach to reformulate query, it is a closet work to the previous article. A key difference is that in this work the reformulation component uses domain-specific template queries. Otherwise in (Nogueira & Cho, 2017), assumes open-domain queries.

Another work that uses query reformulation is that presented in (Azilawati, Abu Bakar & Azman, 2016), in this article authors introduced a new hybrid approach based on the combination of ontology terms and keywords from the query to reformulate new queries. Experimental results show that the proposed hybrid method returns the best results than the use of the keywords extracted from the query or the ontology terms alone.

Authors in (Kamlesh, Pinal & Parth Shah, 2014) proposed a new approach to get the best relevant links in a search engine through User Profiling and Query Reformulation, the researchers propose a

framework that identifies relevant search term for a particular user from previous search history by analyzing web log file maintained in the server, then removes the ambiguities from the original query by appending some useful keywords, the proposed methodology identify user interest on retrieved results by combing the user interest value generated using the Vector Space Model (VSM) and the actual rank of that link, finally this method suggest some keyword to get the best results from the original query. The experimental results of this approach show the effectiveness of the proposed search engine.

Several articles use Reinforcement Learning (RL) in the domain of search engines, as we can found in (Derhami, Khodadadian, Ghasemzadeh & Bidoki, 2013), the authors of this article proposed RL Rank algorithm which is a new connectivity-based method for ranking web pages, This proposed method considers the rank determination of a page as an RL problem where each web page is considered as a state and value function of the state is used to determine the score of that state. The actions are the transition from the current page to the next page, and the reward corresponds to the number of out links from the current page. The researchers propose a hybrid approach using a combination of BM25 (Robertson & Walker, 1994) as a content-based algorithm and RL Rank which give significant results in the experimental results.

The most proposed approach presented earlier use individual methods for finding similar documents to a user query. Our approach take advantages of an hybrid method (that combines between query reformulation, reinforcement learning and semantics) to improve the results of our system and to find the most relevant documents (minimize the error rate)

3. RESEARCH METHODOLOGY

In this section, we going to introduce the various steps of our work and the techniques used in our proposed approach. As we have presented before the point of our work is to get the most relevant documents for a query in a web search engine using reinforcement-learning algorithms based on n-grams model and semantic similarity.

Our system (RL approach) is composed of different steps, we will start by the preparation of the query and documents in our corpus (remove stop-words, get stem of words), then get n-grams word (the notion of query reformulation) and finally calculate similarity using a hybrid approach that uses a syntactic similarity and semantic similarity (El hadi, Madani, El ayachi & Erritali, 2019).

3.1 Construction of the Corpus

In this paper, we are collecting data from ezinearticles.com and divide them into different categories, we collect 10 articles for each category from the website to construct our corpus and each article is inserted in a document.

3.2 Text Preprocessing Method

After the steps of the collection of articles and inserted them in documents, we have constructed our data-set, but before the classification and the application of our proposal, we must make some text pre-processing methods to prepare the documents and the query for the classification and to delete the noise exist in them. Several works from the literature demonstrate that the application of the text pre-processing methods on the text improve the quality of the classification, in our work we apply some text pre-processing methods such as:

- **Tokenization:** Which is the phase of splitting the text into terms or tokens by removing white spaces, commas and other symbols etc. This step is very important in our work because we focus on individual words.

- **Removing Stop-word:** There is a kind of word called stopword. They are words of common function in a sentence, such as 'a', 'the', ',', 'to', 'at', etc. These words seem useless for the calculating of the similarity between documents, for that reason they should be deleted.
- **Removing numbers:** In general, numbers have no use when measuring similarity and are removed from documents to refine the document's content.
- **Removing Punctuation:** We don't need pits as characteristics, this are only symbols for separate sentences and words so we delete them from documents.

3.3 Stemming

Stemming is utilized in many information retrieval (IR) systems to decrease variation word structures to regular roots. It is one of the least complex applications of natural language processing to IR. In fact, it is very important in most of the information retrieval systems. The main purpose of stemming is to reduce different grammatical forms/word forms of a word (noun, adjective, verb, adverb, etc.) to its root form. It would do this by using context and grammar to determine the original form of the word, this can be used in many domains like text searching, and the best example of this is web search engines. Searching for the root of a word gives a wider search than trying to find an exact match. Stemming helps in mapping grammatical variations of a word to instances of the same term. In table 1, we present examples of different words with their stemming.

3.4 N-gram Model

N-gram based techniques are one of the most important concepts in natural language processing (NLP) and its applications. Traditional n-grams are sequences of elements as they appear in texts. These components can be words, POS tags, stop-words, characters, or any other elements as they encounter one afterward another in texts. By convention, "n" corresponds to the number of elements in a sequence.

Table 2 shows that for a string the n-gram model divides the word into "n" character with not losing any combination of two successive characters. We can apply the n-gram technique in a text that contains a list of words, table 3 shows on the other hand, that the n-gram does the same thing as the string but this time with words. Finally, we got a list of n-grams that will help us in our work.

Table 1. Example of stem of words

Original word	Stemmed word
close closed closely closing	close
amuse amused amusement amusements amusing	amus

Table 2. Example of n-gram of a word = "Document"

N-gram	Example
2-grams	do, oc, cu, me, en, nt
3-grams	doc, ocu, cum, ume, men, ent
4-grams	docu, ocum, cume, umen, ment

Table 3. Example of n-gram of a text = "I want to eat Chinese food"

N-gram	Example
2-grams	I want, want to, to eat, eat chinese, chinese food
3-grams	I want to, want to eat, to eat chinese, eat chinese food
4-grams	I want to eat, want to eat chinese, to eat chinese food

In many domains, techniques based on n-grams gave very good results. For instance, in natural language processing, n-grams can be used to distinguish between documents written in different languages in multi-lingual collections and to gage topical similarity between documents in the same language [18, 24], but also in some other problems. In this field, n-grams show some of its good features:

- **Robustness:** relatively insensitive to spelling variations/errors;
- **Completeness:** token alphabet known in advance;
- **Domain independence:** language and topic independent;
- **Efficiency:** one-pass processing; and
- **Simplicity:** no linguistic knowledge is required.

On the other hand, the problem, which can appear in using n-grams, is the exponential explosion. Many algorithms with n-grams are computationally too expensive even for n = 5 or n = 6, with larger n the n-gram cardinality grows exponentially.

3.5 Semantic Similarity

In this subsection, we will present the semantic similarity approach used in this paper and how we exploit it to make our new approach (how we use it in our RL approach), for that, we use the notions of the information retrieval systems (IRS) and some semantic similarity measures.

In this paper, we use our hybrid semantic similarity approach presented in (El hadi & al, 2019), it is a method that uses cosine similarity as a syntactic similarity, and Leacock and Chodorow measure as semantic similarity.

3.5.1 Cosine Measure

The cosine similarity calculates the similarity between two n-dimensional vectors by determining the cosine of the angle between them. This metric is frequently used in text mining. It uses the complete vector representation, that is to say, the objects' frequency (words). Two documents are similar if their vectors are combined. If two objects are not similar, their vectors form an angle (X, Y) whose Cosine represents the similarity value. The formula 1 is defined by the ratio of the scalar product of vectors X and Y and the product of the norm of x and y. The formula is defined by the ratio of the scalar product of vectors x and y and the product of the norm of x and y:

$$Sim_{Cosine}\left(X,Y\right) = \frac{\sum_{i=1}^{n} x \times y}{\sqrt{\left(\sum_{i=1}^{n} x^2 \times \sqrt{\left(\sum_{i=1}^{n} y^2\right)}\right)}} \quad (1)$$

Each element of the vectors X and Y present the rating of a concept (0 or 1).

3.5.2 Construction of Vectors

In this paper, to construct the vectors, instead of using the number of times the n-grams appear in a document d (frequency), we are developing a new method for calculating the frequency of an n-gram in a document. Let D1 and D2 two documents in a corpus with:

D1 = "Car was cleaned by Jack"
D2 = "Jack drives his car well"

For example, a bi-gram (2-gram) approach will create a vocabulary set like the following:

S1 = ["car was", "was cleaned", "cleaned by", "by jack"]
S2 = ["jack drives", "drives his", "his car", "car well"]

Therefore, after getting S1 and S2 we will construct our complete bi-gram corpus, which give us:

S = ["car was", "jack drives", "was cleaned", "drives his", "cleaned by", "his car", "by jack", "car well"]

The classical method of the vector representation consists of finding the bi-gram represented in S1 and S2 in our complete corpus S, if we find the bi-gram we will represent it in V_i by 1, otherwise we will represent it by 0.

The application of the classical method of the sentences will give us the following result:

V1 = [1 0 1 0 1 0 1 0]
V2 = [0 1 0 1 0 1 0 1]

In the developed method, to calculate the frequency of n-gram, we will calculate the frequency of each word in n-gram in all corpus then get the average of all the frequencies calculated:

$$freq(W) = \frac{\sum_{i=1}^{n} \frac{f_c(w)}{s}}{n} \tag{2}$$

where:

- $f_c(w)$: This is the number of times certain words (w) appear in the document.
- s: the number of words in the full document.
- n: the number of gram used.

After applied the proposed method we get for the previous example:

Vc1 = [0.18 0.06 0.25 0 0.25 0.06 0.18 0.06]
Vc2 = [0.12 0.18 0 0.25 0 0.25 0.06 0.18]

3.5.3 Leacock and Chodorow Measure

This method presented by (Leacock & Chodorow, 1994), which combines the counting of the arcs method and the informational contents method. The proposed measure by Leacock and Chodorowis based on the M is the length of the shortest path between two synsets of Wordnet. The formula 3 define this technique (Leacock & Chodorow, 1994):

$$Sim_{LC}\left(X,Y\right) = -\log\left(\frac{cd\left(X,Y\right)}{2\times M}\right) \tag{3}$$

M is the length of the longest path, which separates the concept root, of ontology, of the concept more in the bottom. We indicate that *cd(X,Y)* is the length of the shortest path, which separates X and Y.

3.5.4 Hybrid Used Approach

In this paper, we will use our hybrid approach presented in (El hadi & al, 2019) that combines cosine similarity and Leacock and Chodorow similarity, and presented with the following formula:

$$Sim\left(q,d\right) = \frac{1}{2}\times\left(\frac{\sum_{i=1}^{n} q\times d}{\sqrt{\left(\sum_{i=1}^{n} q^2\right)}\times\sqrt{\left(\sum_{i=1}^{n} d^2\right)}} + Sim_{LC}\left(q,d\right)\right) \tag{4}$$

3.6 Big Data

One of the problems of searching the most pertinent document is the time necessary to have the desired result if we work on a very voluminous corpus, a very interesting solution is the use of Big Data to parallelize the work that is to say distributed it between several machines, using the framework Hadoop with the MapReduce programming model and HDFS as a distributed file system.

3.6.1 Framework Hadoop

Framework Hadoop Apache's Hadoop is an open-source implementation of Google's Map/Reduce framework. It enables distributed, dataintensive and parallel applications by decomposing a massive job into smaller tasks and a massive data set into smaller partitions such that each task processes a different partition in parallel. The main abstractions are: 1. MAP tasks that process the partitions of a data set using key/value pairs to generate a set of intermediate results and 2. REDUCE tasks that merge all intermediate values associated with the same intermediate key. Hadoop uses the Hadoop Distributed File System (HDFS), which is an implementation of the Google File System (GFS) for storing data (Karun & Chitharanjan, 2013).

3.6.2 Hadoop Distributed File System HDFS

The Hadoop Distributed File System (HDFS) (Carlberger, Dalianis, Hassel & Knutsson, 2001) is an open-source distributed storage system inspired by the design of GFS. HDFS is initially developed for the Hadoop MapReduce computation engine. With the rapid advancement of the Apache Hadoop project, HDFS now serves a thriving ecosystem of open-source big data technologies, such as Giraph, Spark, Pig, Hive, and HBase. The performance of HDFS is crucial to all these components built on top of it (Wu & Hong, 2015).

HDFS cluster has master/slave architecture with a single Name Node as the master server, which manages the file system namespace and regulates access to files by clients. The slaves are a number of Data Nodes. Usually, one per node in the cluster, which manage storage attached to the nodes that they run on. HDFS provides a file system namespace and allows user data to be stored in files (Karun & Chitharanjan, 2013). A file is partitioned into one or more blocks and these blocks are stored in a set of Data Nodes. The file system namespace operations like opening, closing, and renaming of files and directories are done by the Name Node. It also decides the mapping of blocks to Data Nodes.

3.6.3 MapReduce

The MapReduce framework has two basic operations on the data i.e. Mapper and Reducer function:

- **Mapper:** It is applied to all input key-value pairs, which generate many intermediate key-value pairs. Mapper next sorts the set of key-value pairs by their keys. The combiner can emit any number of key-value pairs, but the keys and values must be of the same type as the mapper output then partitioners are responsible for dividing the intermediate keyspace and assigning intermediate key-value pairs to reducers (Verma, Mansuri & Jain, 2016).
- **Reducer:** The Reducer's first job is to gather all of its input from the various Mapper output partitions. The sort is functioning here, the shuffle and sort occur simultaneously from different groups. After Shuffle and Sort reduce work, the output of the reduce task is returned on file system (Verma & al, 2016).

3.7 Proposed Method

In this subsection, we describe our proposed approach based on semantic similarity and reinforcement learning. Our new reinforcement learning approach is based on n-gram and the semantic similarity for finding the optimal result. Our idea is that each n-gram has his own most relevant document, so we need to adapt the learning process based on the n-gram model (find the optimal path that the user must follow to reach the goal optimally).

- **Reinforcement learning (RL):** Reinforcement learning (RL) is an intelligent technique with the ability to learn from interaction with the environment. It learns from trial and error and generally does not need any training data or a user model. At the beginning of the learning process, the RL agent does not have any knowledge about the actions it should take. After a while, the agent learns which actions yield the maximum reward. The problem consists of an agent with its various states S and a set of actions per state A. The agent can move from one state to another by performing some action a. the next state gives a reward to the agent. The goal of the agent is to maximize the total reward. If we compare all that with our proposed RL approach, the user is the agent, and the environment is our search engine platform with a query.

The ultimate goal for the agent is to find the optimal path to reach the final goal, which is returning the most relevant documents to the query. The initial state of the agent in our RL approach is the introduction of the query by the user and by doing all the steps presented in figure 2, and the objective (final state) is to return the most relevant documents, which are at the top of the list after calculating the recall. In our proposed RL system, the states are the list of documents obtained after choosing an action. In each state, the agent can take different actions to move from one state to another to reach the final goal. The actions of our approach are the different n-grams (1-gram, 2-grams, 3-grams...). After that the agent chooses an action, his state will be changed, that is the list of his result will be changed, and the reward produced is obtained by calculating the accuracy of the result after calculating the semantic similarity between the query and documents in our corpus.

Figure 1. Agent–environment interaction

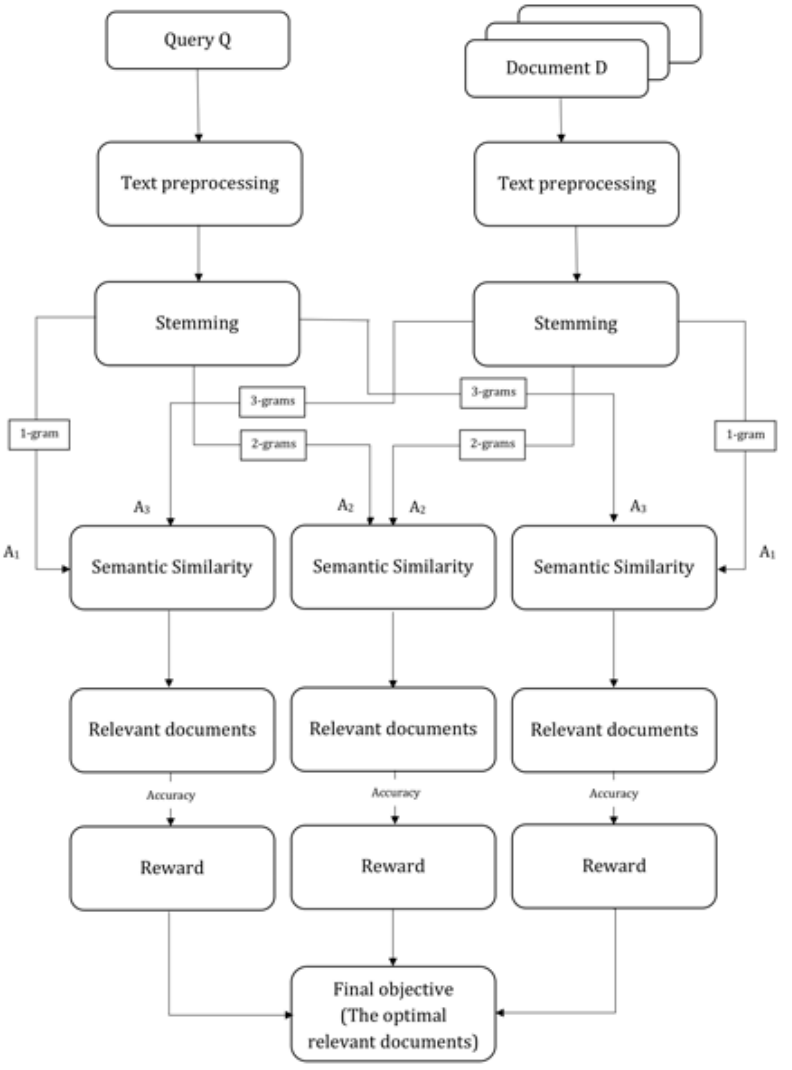
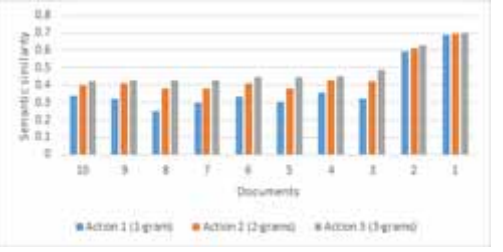


Figure 2. Our proposed approach system



In each state, the goal is to choose the action that maximizes the semantic similarity to get the most relevant document with the n-gram.

Figure 1 shows how the agent interacts with the environment. In each step and after the agent take an action, his state will be changed, which means that his list of the relevant documents will be changed. For finding the list of the relevant documents, we used our semantic similarity approach presented earlier, and by calculating the accuracy of the result obtained, we find the reward produced by taking an action.

In our method, the agent will take actions (1-gram, 2-grams, 3- grams) and the environment will reward him (accuracy) based on the list of the documents obtained (hybrid semantic similarity between the documents in the corpus and the query in the input of the search engine). The goal of our approach is to know what n to use in n-gram (what action to use) to maximize the hybrid semantic similarity between the query and the documents. Therefore, the agent will discover by himself the actions that give the highest rewards by trying them. A user experience using our approach is composed of 4 steps: a current state, take an action, next state (new state with new list of relevant document), and a reward (calculating the accuracy). The value function or the update function of our approach is the measure of accuracy for evaluating the result obtained; this measure consists in associating with each possible state an estimation of the value for the agent to be in this state.

To fully explain our approach, figure 2 will present how our system works based on RL approach.

According to figure 2, in each iteration of our system based on reinforcement learning, the agent choose n in n-gram and apply the different step presented in the previous sections, we start by text pre-processing to remove stop words, punctuations, URLs...etc; from the query and documents in our corpus, as presented in text Pre-processing method section, then we apply stemming to get the root words of different words, this step is important because we reduce complexity in our system and it will not affect the results because convert word to its stem will not lose the meaning of the word. After this step of stemming, the agent in our system will iterate the 3 actions (1-gram, 2-grams, and 3-grams), in the figure 2 A_1 is the action 1 (by choosing 1-gram) A_2 is the action 2 (by choosing 2-grams) and A_3 is the action 3 (by choosing 3-grams), the agent iterate all theses actions to define the n-grams of the query and each document in the corpus, for calculating the similarity, which will give us the most relevant documents. In the end, for each state (after choosing an action) we calculate the recall to get the reward. If we use for example 1-gram, 2-grams, and 3-grams, and we got in the recall 25%, 50% and 75% respectively, so the agent will choose the 3-grams, which is the best path to get the most relevant document to our initial query.

3.8 Our Work Steps and Algorithm

As presented earlier, our approach is based on reinforcement learning using several important steps. We will present them in a form of MapReduce algorithm, which is a specific framework for distributed processing management in a large number of documents, our mapper contains 2 steps, the first is to apply text processing to any document in our corpus to remove stop-words, punctuations, URLs...etc from text. The second is to get the stem of words in documents that help us to get better results in semantic similarity as we will see in our experimental results. Then we send the document id and the list of terms produced in the stemming step to the reducer. Our reducer does the necessary work to get the most relevant documents. The first step in the reducer is to apply the 2 functions in the mapper to our query, then get N-grams based on 'n' in the input (in this paper we will work with $n \hat{I} \{1,2,3\}$). And finally, the calculation of semantic similarity between query and documents based on n-grams model. Here the agent in our system needs to calculate semantic similarity in each action (1-gram, 2-grams, 3-grams) and get the reward (recall) returned by these actions to decide which actions return the most relevant documents to select this path and return the optimal result to the user.

Algorithm 1: Method Mapper

```
Input: Corpus
Output: Indexed documents for document ∈ Corpus do
    TPD ← Text Processing(document);
    STD ← Stemming(TPD);
    for term ∈ STD do
            write(Docid, term);
    end
end
```

Algorithm 2: Method Reducer

```
Input: Docid, List(term)
Output: Most relevant documents
  n ← 1 ;
  TPQ ← Text Processing(Query) ;
  STQ ← Stemming(TPQ) ;
  while n ≤ 3 do
    NGD ← Ngrams(List(term),n) ;
    NGQ ← Ngrams(STQ,n) ;
    Vector(d) ← indexing(NGD) ;
    Vector(q) ← indexing(NGQ) ;
    C ← Cosine(Vector(q),Vector(d)) ;
    SimLC ← SimilarityLC(NGQ,NGD) ;
    Sim ← ½ * (C + SimLC);
    SimDoc[n] ← getSimilarDocuments(n,List(Sim)) ;
    Accuracy[n] ← getAccuracy(SimDoc[n]);
    n++ ;
end
RelevantN ¬ Max(Accuracy) ;
Relevant_Documents(RelevantN) ;
```

- **Text Processing(document):** is a function that allows removing stop-word from the text in corpus and query.
- **Stemming(TPQ):** is a function that converts words in text returned by the first function in their stem to get better results.
- **Ngrams(STQ, n):** is the function that returns the n-grams based on n in the input.
- **Cosine(Vector(q), Vector(d)):** Calculate the cosine similarity between query and document in the corpus.
- **SimilarityLC(NGQ, NGD):** Calculate the similarity semantic using Leacock and Chodorow approach based on n-grams model.
- **getSimilarDocuments(n, List(Sim)):** is the function that return the relevant documents for n-grams using a threshold, which means returning the documents with a degree of similarity greater than or equal the threshold.
- **getAccuracy(SimDoc[n]):** is the function that return the accuracy of the relevant documents using n-gram.
- **Max(Accuracy):** a function that returns the maximum of accuracy in 1,2 and 3grams.
- **Relevant_Documents(RelevantN):** is the function that return the relevant documents based on n-gram with the max accuracy.

4. EXPERIMENTAL RESULTS

In this section, we going to present some experimental results of our work. To evaluate our new similarity approach based on the reinforcement learning, we conduct extensive performance study, the experiment is to run our MapReduce algorithm with a variable number of documents in the input (in the corpus stored in HDFS). We compute the semantic similarity measure between the query and all the documents in the corpus, and we going to compare the accuracy of the obtained result with other approaches from the literature.

Our first experiment consists of comparing the semantic similarity between the first 10 returned documents in our system. The agent in our system iterates the three actions (1-gram, 2-grams, and 3-grams). In each action, it calculates the semantic similarity, which produces a new state, and by calculating the reward (the accuracy of the result), it evaluates the result obtained by taking such action. Figure 3 shows the results obtained.

According to Fig. 3, we remark that the agent in our system(based on our dataset) finds for all documents that the semantic similarity of the third actions (3-grams) is better than the semantic similarity if we use the action 1 (1-gram) or the action 2 (2-grams) using our corpus. In this case, the agent will choose the action 3 to return the most relevant documents, but not always the action 3 is the one which we will use to get the most relevant document, by the use of another corpus or by changing the query we can find that the action 1 or 2 give us the best results, it is the purpose of our proposed approach based on reinforcement learning, that is to let the agent choose which action gives better results and return the most relevant documents to the user.

In the first experiment, we demonstrated that the semantic similarity based on action 3 (3-grams) is more efficient than the others (according to our query and dataset). In this second experiment, we will show how stemming is important for our approach to give us better results. Figure 4 shows the results obtained.

According to Fig. 4, we conclude that the use of stemming improves the semantic similarity score, for that we chose to work with stemming in our proposed approach.

In the next experiment, we compared our proposed approach based on reinforcement learning using n-grams with several approaches in the literature, such as the hybrid approach (El hadi & al, 2019) based on a syntactic and a semantic similarity that we have based in this paper to calculate semantic similarity, Wu & Palmer (Wu & Palmer 1994) and Resnik approach (Resnik 1999).

Figure 5 shows the results obtained.

According to Fig. 3 the agent in our system, choose the action 3 to define his path to return the most relevant document using our corpus, and as you can see in the figure 5, there is a huge difference in semantic similarity score between our proposed approach based on reinforcement learning and the hybrid approach (El hadi & al, 2019). The figure shows also that the use of reinforcement learning outperforms some other approaches of the literature.

Figure 3. Semantic similarity by n-grams

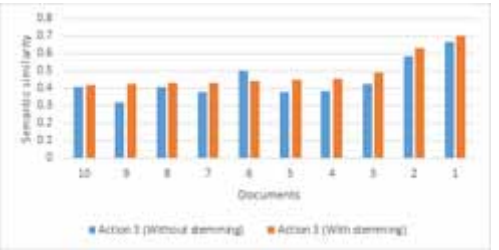


Figure 4. Semantic similarity without stemming

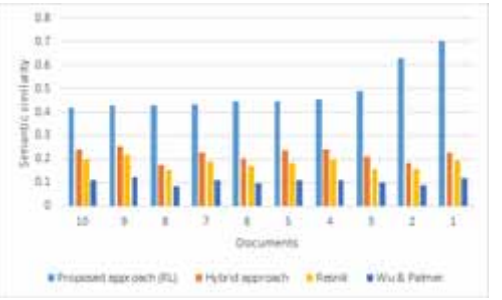


Figure 5. Semantic similarity of our Approach Compared to others approaches

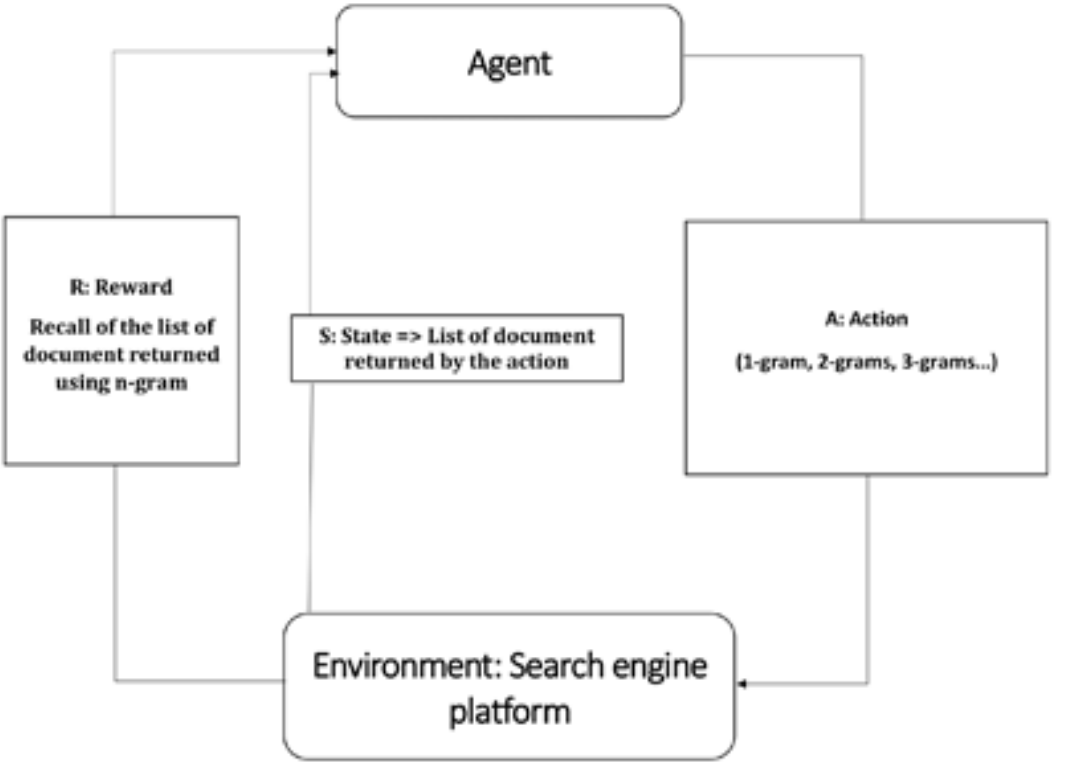


Table 4 shows the precision, recall, and F1 of our proposed approach based on reinforcement learning, it shows how our new approach is effective compared to our previous approach described in the literature.

5. CONCLUSION

In this work, we have presented a new approach based on reinforcement learning using n-grams. Our approach deals with the problem of getting the most relevant documents in a search engine. Our proposed method is based on a new reinforcement learning approach using n-grams for finding the

Table 4. Similarity of our Approach Compared to others

	Precision	Recall	F1
Proposed approach (RL)	60%	42.85%	49.99%
Hybrid approach	40%	28.57%	33.33%

optimal path with which the user of a search engine must follow to get the desired results. We have shown that the use of n-grams added positive value to the proposed approach.

The proposed semantic similarity method is developed using the Hadoop framework with the Hadoop Distributed File System (HDFS) and the MapReduce programming model.

We compared our approach with existing measures in the literature like the Hybrid approach, Wu and Palmer or Leacock and Chodorow. The experimental results show that using n-grams improves the semantic similarity score of our method and that our proposed approach outperforms some other methods from the literature.

Our next work will consist in proposing a new multilingual approach that use a new recommendation method in order to enhance the results of our search engine.

FUNDING AGENCY

The publisher has waived the Open Access Processing fee for this article.

REFERENCES

Azizan, A., Abu Bakar, Z., & Noah, S. A. (2016). Query reformulation using ontology and keyword for durian web search. In *Third International Conference on Information Retrieval and Knowledge Management (CAMP)*. IEEE. doi:10.1109/INFRKM.2016.7806342

Cai, Y., Zhang, Q., Lu, W., & Che, X. (2018). A hybrid approach for measuring semantic similarity based on IC-weighted path distance in WordNet. *Journal of Intelligent Information Systems*, *51*(1), 23–47. doi:10.1007/s10844-017-0479-y

Carlberger, J., Dalianis, H., Hassel, M., & Knutsson, O. (2001). Improving Precision in Information Retrieval for Swedish Using Stemming. *Proceedings of NODALIDA'01 13th Nordic Conference on Computational Linguistics*.

Cho, S., Lebanoff, L., Foroosh, H., & Liu, F. (2019). Improving the Similarity Measure of Determinantal Point Processes for Extractive Multi-Document Summarization. *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. doi:10.18653/v1/P19-1098

Derhami, V., Khodadadian, E., Ghasemzadeh, M., & Bidoki, A. M. Z. (2013). Applying reinforcement learning for web pages ranking algorithms. *Applied Soft Computing*, *13*(4), 1686–1692. doi:10.1016/j.asoc.2012.12.023

El hadi, A., Madani, Y., El Ayachi, R., & Erritali, M. (2019). A new semantic similarity approach for improving the results of an Arabic search engine. *Procedia Computer Science, 151*, 1170-1175. .10.1016/j.procs.2019.04.167

Gao, J. B., Zhang, B. W., & Chen, X. H. (2015). A WordNet-based semantic similarity measurement combining edge-counting and information content theory. *Engineering Applications of Artificial Intelligence*, *39*, 80–88. doi:10.1016/j.engappai.2014.11.009

Karun, K. A., & Chitharanjan, K. (2013). A Review on Hadoop-HDFS Infrastructure Extensions. *Proceedings of 2013 IEEE Conference on Information and Communication Technologies (ICT 2013)*. doi:10.1109/CICT.2013.6558077

Leacock & Chodorow. (1994). Filling in a sparse training space for word sense identification. *Ms. March*.

Makvana, K., Shah, P., & Shah, P. (2014). A novel approach to personalize web search through user profiling and query reformulation. In *International Conference on Data Mining and Intelligent Computing (ICDMIC)*. IEEE. doi:10.1109/ICDMIC.2014.6954221

Narasimhan, K., Yala, A., & Barzilay, R. (2016). Improving informationextraction by acquiring external evidence with reinforcement learning. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2355–2365. doi:10.18653/v1/D16-1261

Nogueira, R., & Cho, K. (2017). Task oriented query reformulation with reinforcement learning. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 585–594. doi:10.18653/v1/D17-1061

Resnik, P. (1999). Semantic similarity in a taxonomy: An information based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, *11*, 95–130.

Robertson, S. E., & Walker, S. (1994). Some simple effective approximations to the 2-poissonmodel for probabilistic weighted retrieval. *Proceedings of SIGIR Conference on Research and Development in Information Retrieval*, 232–241. doi:10.1007/978-1-4471-2099-

Sutton, R. S., & Barto, A. G. (2013). Reinforcement Learning: An Introduction. *Journal of Machine Learning Research*, *14*, 399–436.

Verma, A., Mansuri, A. H., & Jain, N. (2016). Big Data Management Processing withHadoop MapReduce and Spark Technology: A Comparison. *Proceedings of the 2016 Symposium on Colossal Data Analysis and Networking (CDAN)*.

Wu, J., & Hong, B. (2015). Multicast-based Replication for Hadoop. Proceedings of2015 IEEE SNPD 2015.

Wu, Z., & Palmer, M. (1994). Verb semantics and lexical selection. *Proceedings of the 32nd Annual Meeting of the Associations for Computational Linguistics*, 133-138.

Amine El Hadi obtained a master's degree in business intelligence from the faculty of science and technics, Beni Mellal, Morocco in 2017, and he is a Ph.D. Student in Sultan Moulay Slimane University, faculty of sciences and technics, Beni Mellal, Morocco. His current interests focuses on Big Data, information retrieval, machine learning, semantic web.

Youness Madani obtained a master's degree in business intelligence from the faculty of science and technics, Beni Mellal, Morocco in 2016, and a Ph.D. in Computer Sciences from Sultan Moulay Slimane University, faculty of sciences and technics, Beni Mellal, Morocco, in 2019. His current interests include developing new methods and algorithms related to data science, data mining, Big Data, information retrieval, machine learning, and e-learning.

Rachid El Ayachi obtained a degree in Master of Informatic Telecom and Multimedia (ITM) in 2006 from the Faculty of Sciences, Mohammed V University (Morocco) and a Ph.D. degree in computer science from the Faculty of Science and Technology, Sultan Molay Slimane University (Morocco). He is currently a member of laboratory TIAD and a professor at the Faculty of Science and Technology, Sultan Moulay Slimane University, Morocco. His research focuses on image processing, pattern recognition, machine learning, semantic web.

M. Erritali obtained a master's degree in business intelligence from the faculty of science and Techniques, Beni Mellal at Morocco in 2010 and a Ph.D. degree in Computer Sciences from the faculty of sciences, Mohamed V Agdal University, Rabat, Morocco in 2013. His current interests include developing specification and design techniques for use within Intelligent Network, data mining, information Retrieval, image processing and cryptography. He is currently a professor at the Faculty of Science and Techniques, University Sultan Moulay Slimane, and also a member of the TIAD laboratory.