IJCSI International Journal of Computer Science Issues, Volume 15, Issue 2, March 2018
ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784
www.IJCSI.org https://doi.org/10.5281/zenodo.1227767

N-GRAM Based Semantic Enhanced Model for Product Information Retrieval

Mang'are F. Nyamisa[1], Prof. Waweru Mwangi[2], Dr. Wilson Cheruiyot[3]

[1]School of Computing and Information Technology
Nairobi, Kenya

[2]School of Computing and Information Technology
Nairobi, Kenya

[3]School of Computing and Information Technology
Nairobi, Kenya

Abstract

The current information retrieval mechanisms are based on models such as Boolean model, extended Boolean model, vector space model, and probabilistic model and language models. However, these models fall short of expectations, leading to misunderstanding of the user query and therefore the information retrieved fail to meet user expectations. In this paper, a novel search technique is proposed as the possible solution to the problems inherent in the current information retrieval models. To achieve this objective, an experimental research design was utilized. This new technique is based on the concept of N-gram coupled with a conceptual search in which information is searched based on meaning instead of matching of keywords. Thereafter, N-grams are employed to tokenize and display the retrieved information. To validate the proposed approach, a number of experimentations were carried out based on the current search criteria as well as the N-gram based semantic search. The results obtained demonstrated that the proposed information retrieval technique outperformed the current models in terms of precision, recall and F-measure. As such, this model proves significant in the information retrieval process as it accomplishes search by meaning instead of keyword based searching.

Keywords: *Semantic search, N-gram, information retrieval, search engine.*

1. Introduction

An N-gram is a token consisting of a series of characters or words. The simplest n-gram is the unigram, where $n = 1$. According to [1], n is a fixed number, highly dependent on the particular corpus of documents and the queries made against that corpus. [2] defined an n-gram as a sequence of n words: a 1-gram (unigram), a 2-gram (or bigram) is a two-word sequence, and a 3-word (or trigram) is a three-word sequence of words. In n-grams, initially the general pre-processing is carried out. Then the document is divided into N-grams or N-shingles. This refers to a sequence of consecutive words of size `N', where `N' is user specified.

In their study, [3] employed N-grams of several different lengths simultaneously by appending blanks to the beginning and ending of the string. This helped in the matching the beginning-of-word and ending-of-word situations. In addition, they employed underscore character ("_") to represent blanks to decompose the word "TEXT" into different N-grams as follows:

bi-grams: _T, TE, EX, XT, T_

tri-grams: _TE, TEX, EXT, XT_, T_ _

quad-grams: _TEX, TEXT, EXT_, XT_ _, T_ _ _

In their work, [4] discuss that n-gram models are widely used in statistical natural language processing. Additionally, in speech recognition, phonemes and sequences of phonemes are modeled using a n-gram distribution. For parsing, words are modeled such that each n-gram is composed of n words.

2. Related Work

In this section, a review of N-gram, semantics and the current information retrieval models is provided as discussed in the sub-sections that follow.

2.1 N-Gram Based Language Modeling

In general, a string of length k, padded with blanks, will have k+1 bi-grams, k+1 tri-grams, k+1 quad-grams. According to [5], N-gram-based matching has had some success in dealing with noisy ASCII input in other problem domains, such as in interpreting postal addresses, text retrieval and in a wide variety of other natural language processing applications.

In his study, [6] assumed that a language has T word types in its lexicon, and then sought to establish how

likely is word x to follow word y. The task of predicting the next word can be stated as attempting to estimate the probability distribution function *P:*

$$P(w_n|w_{1,\ldots\ldots}w_{n-1}) \quad (1)$$

This is essentially the probability of word W_n given some history $W_{1,\ldots\ldots}W_{n-1}$. In other words, a classification of the previous history words (or context) is used to predict the next word. On the basis of having looked at a lot of text, it can be determined which words tend to follow other words.

In N-gram models, the idea is to model sequences using the statistical properties of N-Grams. According to [7] the Shannon idea is that given a sequence of letters, one can establish the likelihood of the next letter. This can be achieved using training data, from which one can derive a probability distribution for the next letter given a history of size n. in this way, the N-gram model becomes a Markov chain.

In N-grams, the Markov assumption is utilized [8]. This assumption states that the probability of the next word depends only on the previous k words. This can be mathematically represented as follows:

$$P(w_n|w_1 \ldots\ldots\ldots\ldots\ldots w_{n-1} = P(w_n|w_{n-k}w_{n-k+1} \ldots\ldots\ldots w_{n-1}) \quad (2)$$

Where w_{n-k+1} is the (k+1)-gram of the K[th] order Markov approximation

Using this modeling, the common N-grams become:

$$Unigram: P(w_1w_2 \ldots\ldots\ldots w_n) = P(w_1)P(w_2) \ldots\ldots\ldots P(w_n) \quad (3)$$

$$Bigram: P(w_1w_2 \ldots\ldots\ldots w_n) = P(w_1)P(w_2|w_1) \ldots\ldots\ldots P(w_n|w_{n-1}) \quad (4)$$

$$Trigram: P(w_1w_2 \ldots\ldots\ldots w_n) = P(w_1)P(w_2|w_1) \ldots\ldots\ldots P(w_n|w_{n-2}|w_{n-1}) \quad (5)$$

In general, using N-gram models, given P(w_1,w_2, ...,w_n), utilizing the Chain Rule, the joint probability, such as P(w_1,w_2,w_3) can be decomposed as follows:

$$P(w_1, w_2, \ldots\ldots w_n) = P(w_1|w_2, w_3, \ldots\ldots w_n)P(w_2|w_{3\,,\ldots\ldots}w_n) \ldots) \quad (6)$$

$$P(w_{n-1}|w_n)P(w_n) \quad (7)$$

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_1^{k-1}) \quad (8)$$

[9] point out that the key benefit of N-gram-based matching is that since every string is decomposed into small parts, any errors that are present tend to affect only a limited number of those parts, leaving the remainder intact. If N-grams that are common to two strings are counted; a measure of their similarity that is resistant to a wide variety of textual errors is obtained.

According to [10] an *n*-gram of size one is referred to as a unigram; that of size two is a ; while that of size three is a trigram. Larger sizes are sometimes referred to by the value of *n* in modern language, such as four-gram and five-gram. [11] explain that *n*-gram models find applications in areas such as probability, communication theory, computational linguistics (such as statistical natural language processing), computational biology (such as biological sequence analysis), and data compression.

Given the n-gram representations of the two documents d_1 and d_2, $S_n(d_1)$ and $S_n(d_2)$, the Jaccard coefficient, as expressed in equation (viii) , can be used to compute the similarity of the two documents:

$$sim(d_1, d_2) = \frac{|S_n(d_1) \cap S_n(d_2)|}{|S_n(d_1) \cup S_n(d_2)|} \quad (9)$$

A threshold is used to determine whether d_1 and d_2 are likely to be duplicates of each other. For a particular application, the window size *n* and the similarity threshold are chosen through experiments.

[12] described n-gram language model as a probability model used to predict the probability of a n-gram in a language. More specifically, it predicts the next word after a sequence of words. By using these probabilities, one can calculate the probability that a document d belongs to a certain class c given by Eq. (10):

$$P(c|d) = \prod_{i=1}^{n} P(w_i|w_{i-n+1}^{i-1}) \quad (10)$$

$$P(w_i|w_{i-n+1}^{i-1}) = \frac{freq(w_{i-n+1}^{i-1}w_i)}{freq(w_{i-n+1}^{i-1})} \quad (11)$$

Since training data is finite, it is very likely that some n-grams will occur in a document are not seen in the training data. These n-grams will cause n-gram model in Equation 2.6 to assign probability of zero to the document. These zero probability n-grams need to be changed to ensure an accurate probability to be calculated. This is done by smoothing. Smoothing gives probabilities to n-grams of which the smaller n-grams have occurred in the training data [12].

2.2 Semantics

The field of semantics studies the relationship between signifiers such as words, phrases, signs and symbols, and what they stand for [13]. A number of theories exist in this field and they include formal semantics, truth-conditional semantics, conceptual semantics, lexical semantics and computational semantics. In formal semantics, coverage of a range of approaches to the study of meaning, from model-theoretic to proof-theoretic semantics is provided while in truth conditional semantics, the idea is that given that a sentence is a representation that has a predicative structure, its representational content determines the conditions under which the representation is true. On the other hand, conceptual semantics provide simple and general tools for describing observations in reality. In lexical semantics, the focus is on word denotations and it

presumes that the connotation of a given word can be wholly mirrored by the context in which it is used [14]. On its part, computation semantics, emphasis is placed on process of automating the construction and reasoning with connotation depictions of natural language expressions.

2.3 Information Retrieval models

According to [15], the information retrieval strategies transform documents into suitable representations so that relevant documents can be retrieved effectively. Informational retrieval models are grouped according to dimensions, the mathematical basis (first dimension) and the properties of the model (second dimension). The first dimension models can further be classified into four categories namely: set-theoretic, algebraic models, probabilistic models and feature-based retrieval models. In their study, [16] proposed a new computational approach for tracking and detecting statistically significant linguistic shifts in the meaning and usage of words.

3. Methodology

This paper employed an experimental research design to achieve the laid down objectives. A number of experiments were carried out on the current information retrieval models and the proposed N-gram based semantic search information retrieval. Python programming language, due to its support for natural language processing was chosen as an appropriate programming language for the proposed information retrieval.

In this paper, N-gram semantic search is proposed and implemented as a possible solution to the shortcomings of the shortcomings of the keyword-based searching process. The information retrieval here is composed of four components namely the user interface, the semantic engine, working memory and the knowledge base as illustrated in Figure 1. In this new information search, the retrieval process starts by having the user input the facts, questions or queries on the user interface. These details are then passed to the semantic engine which first establishes whether the entered query is in the knowledge base. If the query is in the knowledge base, the semantic engine carries out associative chaining to all related terminologies to the one input by the user. Afterwards, semantic engine contacts the knowledge base containing all the product information and opens all these documents. In order to make the retrieval process faster and accurate, the N-gram concept is employed to tokenize the content to be retrieved.

This requires that the total length of the entire content be established so that the value of *n* in N-gram can be

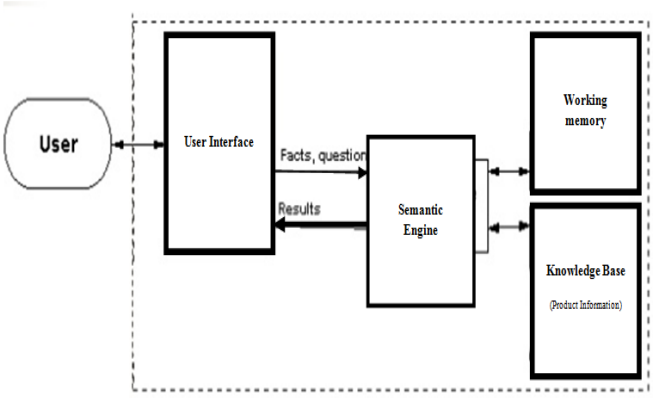

Fig.1: Proposed N-gram Based Semantic Search

initiated to a value *n-1* to prevent buffer overflows. The *'For…loops..'* are then employed to increment the value of *n* from zero to *n-1*, displaying the equivalent grams on the user interface as shown in Figure 2.

```
SearchCriteria= raw_input("PLANES: ENTER YOUR SEMANTICS HERE: ");

N = len(category_1)-1

grams = [category_1[i:i+N] for i in xrange(len(category_1)-N)]
#
#########################################################################
#
if SearchCriteria in category_1:
 tf1=category_1.count(SearchCriteria)
 print "[********************* :MATCHES FOUND: **************************]", [tf1]
 print"-----------------------------------------------------------------";
 print"[****** MATCHING SEMANTICS FOUND:-*******]",
 print [max(grams)]
```

Fig.2: N-gram Tokenization

Upon reaching the value *n-1*, the end of the content would have been reached and the algorithm stops and at this point, all the relevant document content being displayed on the user interface. Figure 3 provides a conceptual interaction model among the information retrieval participants. As shown here, the inference engine contact the knowledge base to find out whether the user query is in its database and if this is the case, the inference engine performs linkages among the semantically related queries and retrieves the relevant documents. In this information retrieval algorithm, the only time that relevant information would not be retrieved is when the user query cannot be found in the knowledge base.

On the same breadth, the only chance of returning information for only one product is when such a product does not have other semantically related products. To validate the developed information retrieval algorithm, the conventional metrics for information retrieval were employed, namely precision, recall and F-measure.

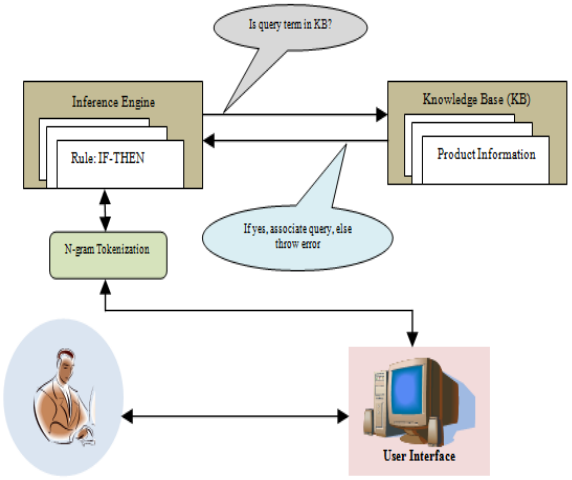

Fig.3: N-Gram Based Information Retrieval Concept

While precision measures how accurately the system picks the related documents among all documents, recall defines the percentage of correctly classified documents among all documents belonging to that category. On its part, F-measure combines precision and recall, taking their harmonic mean and is high when both precision and recall are high. Figure 4 gives a snippet of the evaluation process for four products namely aeroplane, aircraft, plane and airliner.

```
RawList=[TextAeroplane,TextAircraft,TextPlane,TextAirliner]
ConvtList=','.join(RawList);
test1=ConvtList.count("aeroplane")
test2=ConvtList.count("aircraft")
test3=ConvtList.count("airliner")
test4=ConvtList.count("plane")
############################################################
#
predict1=category_1.count("aeroplane")
predict2=category_1.count("aircraft")
predict3=category_1.count("airliner")
predict4=category_1.count("plane")
 #
############################################################
#
 predicted = [test1,test2,test3,test4]
y_test = [predict1,predict2,predict3,predict4]
precision, recall, fscore, support = score(y_test, predicted)
#
############################################################
#
print('Precision: {}'.format(precision*100))
print('Recall: {}'.format(recall*100))
print('F-score: {}'.format(fscore*100))
```

Fig.4: Information Retrieval Evaluation Snippet

All these metrics require the determination of the frequency of these products in the entire document collection and this is accomplished by the first part of this snippet. The second part establishes the frequency of these terms in their individual categories. The rest of the code computes the recall, precision and F-measure for each of these products.

4. Results and Discussions

In this section, the proposed information retrieval algorithm based on N-gram basedwith semantic search is compared with the current keyword base search. To make the comparisons easier, he same query is utilized. As an illustration, the first query *'aeroplane'* was supplied to the current information search process. Figure 5 shows the outcome obtained.

```
------------------------------------------------------------------
ENTER YOUR KEYWORD HERE: aeroplane
------------------------------------------------------------------
----aeroplane:-  is a powered, fixed-wing aircraft that is propelled forward by
 thrust from a jet engine or propeller. airplane come in a variety of sizes
, shapes, and wing configurations. The broad spectrum of uses for airplane.

------------------------------------------------------------------
```

Fig.5: Current Keyword Based Information Retrieval

As this figure shows, only information about *aeroplane* was displayed on the user interface. Therefore, although *aeroplane* is semantically related to airliner, aircraft and plane, the user is denied an opportunity of getting this information. This means that to get information about the other terms, each one of them must be entered as a query on the user interface to yield the information on Figure 6.

```
ENTER YOUR KEYWORD HERE: aeroplane
------------------------------------------------------------------
----aeroplane:-  is a powered, fixed-wing aircraft that is propelled forward by
 thrust from a jet engine or propeller. airplane come in a variety of sizes
, shapes, and wing configurations. The broad spectrum of uses for airplane.

------------------------------------------------------------------
------------------------------------------------------------------
>>> runfile(r'F:\Thesis\Masters\Fry\Programming\Latest-Prototypes\Current_Retrieval.py', wdir=r'
F:\Thesis\Masters\Fry\Programming\Latest-Prototypes')
------------------------------------------------------------------
ENTER YOUR KEYWORD HERE: plane
------------------------------------------------------------------
plane:- an airplane for transporting passengers and cargo.A plane is typically
defined as an aircraft intended for carrying multiple passengers or cargo
in commercial service. The largest airliners are wide-body jets.

------------------------------------------------------------------
------------------------------------------------------------------
>>> runfile(r'F:\Thesis\Masters\Fry\Programming\Latest-Prototypes\Current_Retrieval.py', wdir=r'
F:\Thesis\Masters\Fry\Programming\Latest-Prototypes')
------------------------------------------------------------------
ENTER YOUR KEYWORD HERE: aircraft
------------------------------------------------------------------
aircraft:-  is a machine that is able to fly by gaining support from the air.
 It counters the force of gravity by using either static lift or by using
the dynamic lift of an airfoil, or in a few cases the downward thrust
from jet engines.

------------------------------------------------------------------
------------------------------------------------------------------
>>> runfile(r'F:\Thesis\Masters\Fry\Programming\Latest-Prototypes\Current_Retrieval.py', wdir=r'
F:\Thesis\Masters\Fry\Programming\Latest-Prototypes')
------------------------------------------------------------------
ENTER YOUR KEYWORD HERE: airliner
------------------------------------------------------------------
airliner:- is a type of aircraft for transporting passengers and air cargo.
 Such aircraft are most often operated by airlines. Although the definition
 of an airliner can vary from country to country.

------------------------------------------------------------------
```

Fig.6: Repeated Retrieval of Keywords

As this figure demonstrates, the search process is repeated four times. The challenge of this is that it is slow as the user must now search for all documents one by one to get the required information. In addition, the user must know all the synonyms of a particular query beforehand in order to get all the required information. In addition, the users may easily cause network congestions while requesting for the same data set that would have been otherwise retrieved together and displayed on the user interface. The source of these problems could be traced to the source code, where as shown in Figure 7, this information retrieval basically implements keyword based search.

```
keyword= raw_input("ENTER YOUR KEYWORD HERE: ");
print"------------------------------------------------------------------"
UsableSearchCriteria=keyword+'.txt'
 #
############################################################
#
if UsableSearchCriteria==aeroplane.name:
 print TextAeroplane
 print"------------------------------------------------------------------"
 #
############################################################
#
if UsableSearchCriteria==aircraft.name:
 print TextAircraft
# print "Term Frequency in first file:-------", [tf1]
 print"------------------------------------------------------------------"
 #
############################################################
 #
if UsableSearchCriteria==plane.name:
 # tf1=TextAircraft.count(str)
 print TextPlane
 print"------------------------------------------------------------------"
 #
############################################################
 #
if UsableSearchCriteria==airliner.name:
 # tf1=TextAircraft.count(str)
 print TextAirliner
 print"------------------------------------------------------------------"
```

Fig.7: Keyword-Based Search

This figure illustrates that once the user has input the query, the information retrieval algorithm compares this query with all the file names in its database. When a match is found, the content of this file is displayed on the user interface by use of the *'Print'* command. Therefore for the case of aeroplane, *'print TextAeroplane'* was executed, for aircraft, *'print TextAircraft'* was executed, for plane, *'print TextPlane'* executed and lastly for airliner, *'print TextAirliner'* executed. The only advantage of this kind of information retrieval is that the information is retrieved very fast, since only information concerning one keyword is retrieved at a time. An improvement to this retrieval process would be to carry out semantic associations among the query synonyms and display the information about all these synonyms at the same time on the user interface. To maintain the retrieval speed, N-grams could be employed to tokenize the content to be retrieved. This is exactly what this paper sought to carry out.

In this paper, a semantic engine was introduced to carry out the required associations among the query synonyms. The knowledge base contained all the information of all the products. Here, before any query information could be rendered, the query's synonyms are first sought in the knowledge base and linked to the user query as shown in Figure 8. Afterwards, all the information about the linked documents is displayed at a go on the user interface.

```
UsableSearchCriteria=SearchCriteria+'.txt'
if UsableSearchCriteria==aeroplane.name:
  RawList=[TextAeroplane,TextAircraft,TextPlane,TextAirliner]
  ConvtList=']\n------:'.join(RawList);
  print '\n'
  print ConvtList
  print '\n'
```

Fig.8: Semantic Engine Associative Chaining

The first line takes the user query and appends *'.txt'* to it to convert it into text format that can be handled by python. The second line applies N-gram basedto detect the corresponding file name in the knowledge base. The third line performs associative chaining of the text formats of *aeroplane, aircraft, plane* and *airliner* and stores this chain in a variable, *'RawList'*. The fourth line converts this list into string and retrieves all the files in this string. The fifth line dictates that each of these files should be displayed on a new line and the sixth line displays the contents of these files in the user interface. This process is repeated for the rest of the products in the semantic engine.

To put this N-gram based semantic search into test, the user input the query 'aircraft' at the user interface and executed the semantic engine. The outcome obtained is displayed in Figure 9. As shown in the fifth line, the semantic engine reported to have found matches: *'aeroplane, aircraft, airline and plane'*. The rest of the output was the specific information of these files, starting with aeroplane and ending with airliner. It can be observed that although this is the same content that was retrieved by keyword based search, here no repeated entry of separate keywords was required. This is illustrated by the first line where only *'aircraft'* was input, but because of associative chaining in the semantic engine, information about the synonyms of query 'aircraft' are retrieved as well. This saves time and network resources by requesting for information only once instead of four times as was the case for the keyword-based approach.

```
[----------------------STARTING SEMANTIC SEARCH-----------------]
PLANES: ENTER YOUR SEMANTICS HERE: aircraft
[********************** :MATCHES FOUND: **************************] [1]
-----------------------------------------------------------------
[****** MATCHING SEMANTICS FOUND:-*******] ['aeroplane aircraft airliner plane']


----aeroplane:-  is a powered, fixed-wing aircraft that is propelled forward by
 thrust from a jet engine or propeller. airplane come in a variety of sizes
, shapes, and wing configurations. The broad spectrum of uses for airplane.
]
------:aircraft:-  is a machine that is able to fly by gaining support from the air.
 It counters the force of gravity by using either static lift or by using
the dynamic lift of an airfoil, or in a few cases the downward thrust
from jet engines.
]
------:plane:- an airplane for transporting passengers and cargo.A plane is typically
defined as an aircraft intended for carrying multiple passengers or cargo
in commercial service. The largest airliners are wide-body jets.
]
------:airliner:- is a type of aircraft for transporting passengers and air cargo.
 Such aircraft are most often operated by airlines. Although the definition
 of an airliner can vary from country to country.
```

Fig.9: N-gram Based with Semantic Search Output

4.1 N-gram and Quantity of Output Information

Another advantage of this N-gram based with semantic search is its utilization of N-grams, which can be used to control the content displayed. This was achieved by varying the value of n in the N-grams. In Figure 10, the value of n has been set to 12.

```
from nltk import ngrams
ProductDetails= open("aeroplane.txt", "r+")
TextProductDetails=open('aeroplane.txt').read();

document = TextProductDetails
n = 12
twelve_grams = ngrams(document.split(), n)
for grams in twelve_grams:
  print grams
```

Fig.10: Semantic Engine N-gram =12

This snippet opens a file *'aeroplane'* in read mode and assigns its content to a variable, *'document'*. Since the value of N-grams has been set to *n=12*, it is expected that the output content will be displayed in batches of 12 words or phrases as shown in Figure 11.

```
('----aeroplane:-', 'is', 'a', 'powered,', 'fixed-wing', 'aircraft', 'that', 'is', 'propelled', 'forward', 'by', 'thrust')
('is', 'a', 'powered,', 'fixed-wing', 'aircraft', 'that', 'is', 'propelled', 'forward', 'by', 'thrust', 'from')
('a', 'powered,', 'fixed-wing', 'aircraft', 'that', 'is', 'propelled', 'forward', 'by', 'thrust', 'from', 'a')
('powered,', 'fixed-wing', 'aircraft', 'that', 'is', 'propelled', 'forward', 'by', 'thrust', 'from', 'a', 'jet')
('fixed-wing', 'aircraft', 'that', 'is', 'propelled', 'forward', 'by', 'thrust', 'from', 'a', 'jet', 'engine')
('aircraft', 'that', 'is', 'propelled', 'forward', 'by', 'thrust', 'from', 'a', 'jet', 'engine', 'or')
('that', 'is', 'propelled', 'forward', 'by', 'thrust', 'from', 'a', 'jet', 'engine', 'or', 'propeller.')
('is', 'propelled', 'forward', 'by', 'thrust', 'from', 'a', 'jet', 'engine', 'or', 'propeller.', 'airplane')
('propelled', 'forward', 'by', 'thrust', 'from', 'a', 'jet', 'engine', 'or', 'propeller.', 'airplane', 'come')
('forward', 'by', 'thrust', 'from', 'a', 'jet', 'engine', 'or', 'propeller.', 'airplane', 'come', 'in')
('by', 'thrust', 'from', 'a', 'jet', 'engine', 'or', 'propeller.', 'airplane', 'come', 'in', 'a')
('thrust', 'from', 'a', 'jet', 'engine', 'or', 'propeller.', 'airplane', 'come', 'in', 'a', 'variety')
('from', 'a', 'jet', 'engine', 'or', 'propeller.', 'airplane', 'come', 'in', 'a', 'variety', 'of')
('a', 'jet', 'engine', 'or', 'propeller.', 'airplane', 'come', 'in', 'a', 'variety', 'of', 'sizes')
('jet', 'engine', 'or', 'propeller.', 'airplane', 'come', 'in', 'a', 'variety', 'of', 'sizes', ',')
('engine', 'or', 'propeller.', 'airplane', 'come', 'in', 'a', 'variety', 'of', 'sizes', ',', 'shapes,')
('or', 'propeller.', 'airplane', 'come', 'in', 'a', 'variety', 'of', 'sizes', ',', 'shapes,', 'and')
('propeller.', 'airplane', 'come', 'in', 'a', 'variety', 'of', 'sizes', ',', 'shapes,', 'and', 'wing')
('airplane', 'come', 'in', 'a', 'variety', 'of', 'sizes', ',', 'shapes,', 'and', 'wing', 'configurations.')
('come', 'in', 'a', 'variety', 'of', 'sizes', ',', 'shapes,', 'and', 'wing', 'configurations.', 'The')
('in', 'a', 'variety', 'of', 'sizes', ',', 'shapes,', 'and', 'wing', 'configurations.', 'The', 'broad')
('a', 'variety', 'of', 'sizes', ',', 'shapes,', 'and', 'wing', 'configurations.', 'The', 'broad', 'spectrum')
('variety', 'of', 'sizes', ',', 'shapes,', 'and', 'wing', 'configurations.', 'The', 'broad', 'spectrum', 'of')
('of', 'sizes', ',', 'shapes,', 'and', 'wing', 'configurations.', 'The', 'broad', 'spectrum', 'of', 'uses')
('sizes', ',', 'shapes,', 'and', 'wing', 'configurations.', 'The', 'broad', 'spectrum', 'of', 'uses', 'for')
(',', 'shapes,', 'and', 'wing', 'configurations.', 'The', 'broad', 'spectrum', 'of', 'uses', 'for', 'airplane.')
```

Fig.11: N-gram=12 Output

Considering the first line, we have *''----aeroplane:-' 'is', 'a', 'powered,', 'fixed-wing', 'aircraft', 'that', 'is', 'propelled', 'forward', 'by', 'thrust''*, which gives a total count of twelve (12), hence the name *twelve-grams.*

The rest of the output obeys this *'twelve-grams'* information display. The total output lines are twenty six (26). The value of *n* was then reduced to 9 to yield nine-grams as shown in Figure 12.

Fig.12: N-gram=9 Output

This figure confirms that now the output is in a batch of nine words or phrases. For instance, considering the last line, we have *''wing', 'configurations.', 'The', 'broad', 'spectrum', 'of', 'uses', 'for', 'airplane.''*, which gives a total count of nine (9), hence the name *nine-grams*. The total lines in the output now has shoot up to twenty nine (29). To derive a reduction patter, two more values of *n* were considered (*n=6* and *n=3*). The outputs are as shown in Figure 13.

Fig. 13: (a) N-gram=6 Output (b) N-gram=3 Output

We now derive the general pattern for *n* and output lines displayed, which essentially represent the amount of information displayed. Table 1 below gives the variation of output lines at different values of *n.*

Table 1: Variation of *n* against Output lines

Value of *n* in N-gram	Output lines
37	3
12	26
10	28
9	29
7	31
6	32
4	34
3	35
1	37

This table shows that as the value of *n* increases, the output lines on the user interface reduces and vice versa. To get a clear picture o the relationship suggested here, a graph of *n* versus output lines was plotted in a statistical package as shown in Figure 14. As this figure shows, this is nearly a straight line graph cutting the X-axis at point 12 and the Y-axis at point 35, which were maximum N-grams and output lines respectively. To obtain the gradient of this line, the values of the changes in Y values and changes in X- values were determined. Afterwards, the equation of the straight line, $y = mx + c$ was utilized to compute the predictions of output lines for any given value of *n* in N-gram. Here, m is the gradient, which was computed by dividing the changes in Y by the changes in X, as shown in Figure 15, the

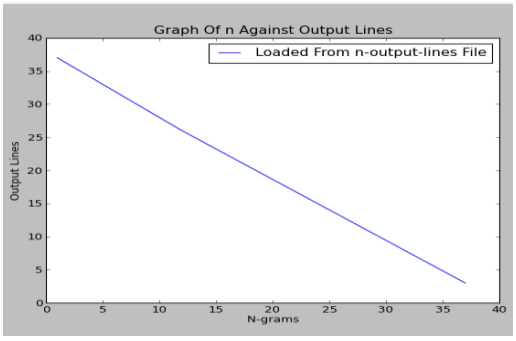

Fig.14: Variations of Output against N-gram

changes in Y, ΔY was 16 and the changes in X, ΔX was 16. As such, the gradient was -1. Since this line does not cross the Y –axis, the value of Y intercept c can be determined by extrapolation. Doing this yielded the Y-intercept value of approximately 39 as per Figure 15. Putting all these values together, we have:

$$y = 39 - x \qquad (12)$$

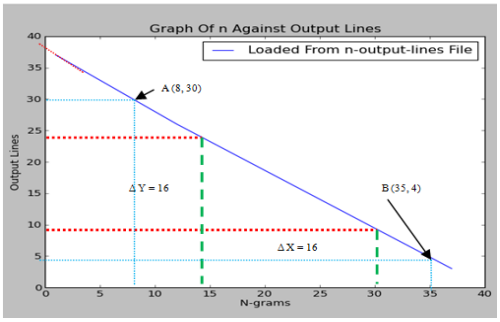

Fig.15: Results Validation

To validate these results, two points A (8, 30) and B (35, 4) were considered. Taking point A, the value of output lines could be approximates as follows: $y = 39 - 8$. This gives a value of 31, which is a close approximation to 30, the value of Y at point A. Similarly, for point B, $y = 39 - 35$. This yields a value of 4, which is exactly the value of Y at point B. We now take any value of *n*, say 15 and try to figure out the number of output lines that will be displayed on the user interface.

When the semantic engine was run with *n* in N-gram initialized to 15, the outcome in Figure 16 was obtained.

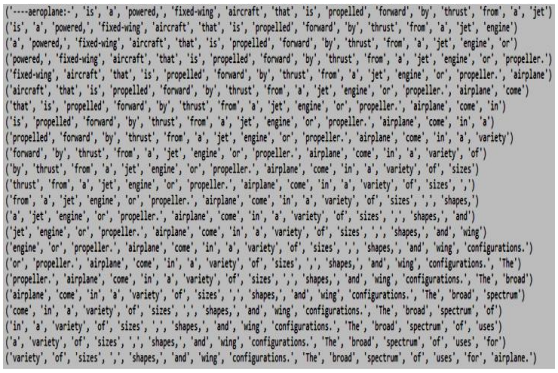

Fig.16: Output for N-gram = 15

This figure shows that the total output lines are 23. From the approximation equation of $= 39 - x$, the number of lines would be: $y = 39 - 15$. This yields a value of 24, which is a close approximation of the observed output lines of 23. These are very crucial observations as it implies that the developed retrieval model could be used to adjust the retrieved content to fit different screen sizes. For instance, if the screen is wide but short in height, then the value of N-grams can be set to a high number to yield few lines of output with content occupying more of the horizontal space than vertical space. On the other hand, if the screen is long in height but short in width, the value of N-grams can be set to a small value to yield many lines occupying less horizontal space.

4.2 Developed Model Parametric Evaluation

Three metrics, precision, recall and F-Measure were employed to assess the developed N-gram based with semantic search information retrieval algorithm. When the snippet of Figure 8 was run, the output in Figure 17 was obtained. These results demonstrate that the probability that the retrieved documents were relevant was a hundred percent, implying that the precision of the developed N-gram based with semantic search was highly accurate. On the other hand, there was a 33.33% probability that relevant documents are retrieved in this search. In addition, the value of the F-measure was 50%, implying that the weighted harmonic mean of the precision and recall is 50%.

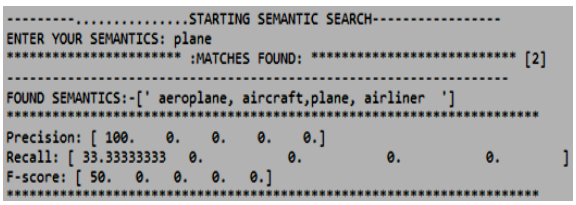

Fig.17: Model Evaluation Output

The next task was the empirical proof of the relationship between precision and recall, which will help in the computation of the F-measure. Theoretically, F-measure can be represented as the average of recall and precision given by Eq. (13):

$$F - measure = \frac{2*precision*recall}{precision+recall} \qquad (13)$$

Substituting for recall and precision in Figure 24, we obtain:

$$F - measure = \frac{2*100*33.33}{100+33.333} \qquad (14)$$

$$= 49.99$$

It is observed that the value 49.99 is approximately 50, the value obtained experimentally for F-measure. As such, the theoretical values agree to a large extent to the experimental values. Since this study sought to develop an information retrieval model that will ensure that there are reduced chances of users missing relevant information when they are querying the information sources by use of search engines, the precision value of 100% is a strong indication that the retrieved documents for this model are highly relevant.

4.3 Model Performance and Document Size

In real life situations, users would be willing to read more documents if it would increase the percentage of the viewed set that were relevant. It was therefore necessary to investigate the variation of precision at various values of recall. Table 2 that follows gives the outcome obtained. Theoretically, if more documents are retrieved, then recall is improved. This means that in situations where all documents are retrieved, then recall will be unity (1) or 100%. However, when fewer documents are retrieved, then precision is improved, but recall is reduced. Consequently, there always some trade-off between precision and recall. The values in Table 2 were used as input to the statistical package to come up with a graph shown in Figure 18.

Table 2: Variation of Precision with Recall

Recall	Precision
0	1
0.1	0.67
0.2	0.63
0.3	0.55
0.4	0.45
0.5	0.41
0.6	0.36
0.7	0.29
0.8	0.13
0.9	0.1
1	0.08

This figure shows that the graph of precision against recall is a decay curve. Although it is not a smooth decay, it gives an approximation of the correlation between these two parameters.

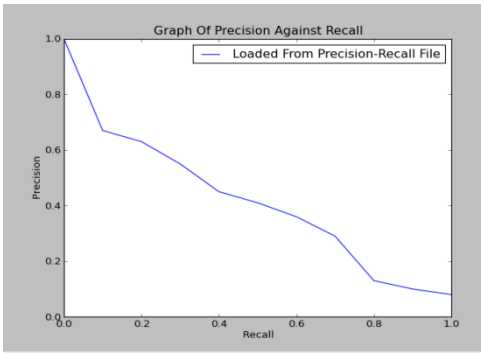

Fig.18: Precision-Recall Graph

In essence, the closer the curve to the (1, 1) point, the better the information retrieval system performance. Taking into consideration the area under this curve, then the closer this area is to one unity, the better the information retrieval system. Figure 19 shows the Precision-Recall graph for an ideal information retrieval system.

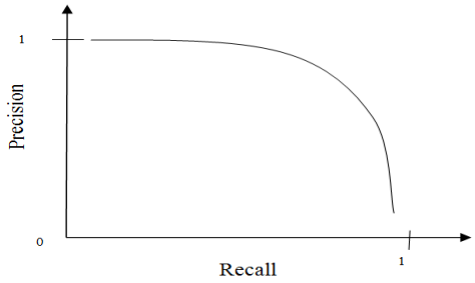

Fig. 19: Perfect Information Retrieval Precision-Recall Graph

Here, the area under this curve is nearly unity and therefore provides a depiction for a perfect information retrieval system. Figure 18 shows that precision and recall are inversely proportional, which can mathematically written as:

$$precision \propto \frac{1}{recall} \qquad (15)$$

where $\propto$ is the proportionality constant. The next phase of this paper was the empirical derivation of this proportionality constant. Comparing the graph of Figure 20 with that of a $y = \frac{1}{x^2}$ shown in Figure 24:

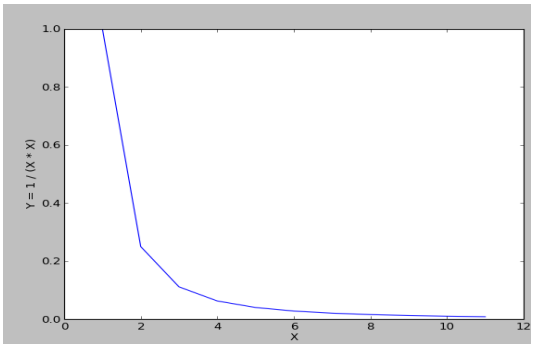

Fig.20: Proportionality Constant Determination

It was noted that these two graphs are fairly similar. Hence the value of the proportionality constant was $\frac{1}{recall}$ and therefore substituting this value in equation (iv), the value of precision can be obtained from recall as shown in equation (v):

$$precision = \frac{1}{(recall)^2} \qquad (16)$$

As such, precision for the N-gram based with semantic search information retrieval model was approximately the inverse of the square of recall.

5. Conclusion and Recommendations

This paper sought to develop a high precision N-gram based with semantic search information retrieval model and to experimentally compare its performance with the current keyword-based information retrieval. The results obtained have clearly indicated that the developed information retrieval model is faster and has high precision. To validate the results, this paper has derived two types of relationships, the first one being that of N-gram and quantity of output information, while the second one was that of precision and recall. In the first case, it has been demonstrated in this paper that it is possible to predict the amount of output information given the value of N-grams. In the second case, it is possible to predict the value of precision given any value of recall. This model is therefore recommended for usage in search engines for accurate information retrieval.

REFERENCES

[1] H. Haibin and C. Shipin, "Design and Implementation of Retrieval System of Network Learning Resources Based on Semantic Web", International Journal of u- and e- Service, Science and Technology, Vol.8, No.5, 2015, pp.241-252.

[2] D. Jurafsky J. Martin , N. Peter, & R. Stuart, "Speech and Language Processing", Springer Science and Business Media, 2014.

[3] B. William and M. John, "N-Gram-Based Text Categorization", Environmental Research Institute of Michigan, 2016.

[4] *K.* Wołk and K., Marasek, "Polish-English Speech Statistical Machine Translation Systems for the IWSLT", in proceedings of the 11th International Workshop on Spoken Language Translation, Tahoe Lake, 2014, USA.

[5]G. Grigori , "Syntactic Dependency-Based n-grams in Rule Based Automatic English as Second Language Grammar Correction", International Journal of Computational Linguistics and Applications, Vol. 4 (2), 2013, pp. 169–188.

[6] M. Adam, "N-Gram Language Models", 2016.

[7]K. Noriaki, "N-gram over Context", 2016.

[8] J. Yuan, F. Gao, Q. Ho, W. Dai, J. Wei, X. Zheng, P. Xing, T. Liu and W. Ma, " Lightlda: Big topic models on modest computer clusters", In WWW, 2015, pp. 1351-1361.

[9] S. Jameel and W. Lam, "An unsupervised topic segmentation model incorporating word order", *In SIGIR*, 2013, pp. 203-312.

[10] Kawamae N. (2012). Identifying sentiments over n-gram. *In WWW*. (pp. 541-542).

[11] J. Daniel and H. James Martin, "N-Grams: Speech and Language Processing", 2014.

[12] J.K. Van Dam, and V. Zaytsev, "Software Language Identification with Natural Language Classifiers", In 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), 2016, Vol.1, pp. 624-628.

[13] H. Haibin and C. Shipin, "Design and Implementation of Retrieval System of Network Learning Resources Based on Semantic Web", International Journal of u- and e- Service, Science and Technology, Vol.8, No.5, 2015, pp.241-252.

[14] J. Botha, and P. Blunsom, "Compositional Morphology for Word Representations and Language Modelling", In Proceedings of ICML, 2014.

[15] P. Sojka and H. Schütze, "Introduction to Information Retrieval", Faculty of Informatics, Masaryk University, 2015.

[16] V. Kulkarni, A. Rami, B. Perozzi and S. Skiena, "Statistically Significant Detection of Linguistic Change", 2015.

Mang'are Fridah Nyamisa: Is a student holding a bachelors degree in mathematics and computer science from Jomo Kenyattta University of Science and Technology (JKUAT). She is currently pursuing her masters degree in computer systems from the same institution.

Prof. W. Mwangi: Is a Professor in the School of Computing and Information Technology, Jomo Kenyatta University of Agriculture and technology (JKUAT). His areas of expertise include Computer Security and Reliability, Artificial Intelligence and Information Systems (Business Informatics).

Dr. Wilson. Cheruiyot: He is a lecturer, Institute of Computer Science and Information Technology, Jomo Kenyatta University of Agriculture and Technology (JKUAT). His research interests include Artificial Intelligent (Agents Applications in Automation and Data Mining, Evolutionary Computing, AI in Multimedia Applications).