(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 March 2014 (27.03.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/046620 A1

(51) International Patent Classification:
G06F 17/30 (2006.01)

(21) International Application Number:
PCT/SG2013/000411

(22) International Filing Date:
20 September 2013 (20.09.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/703,559 20 September 2012 (20.09.2012) US

(71) Applicant: **NATIONAL UNIVERSITY OF SINGAPORE** [SG/SG]; 21 Lower Kent Ridge Road, Singapore 119077 (SG).

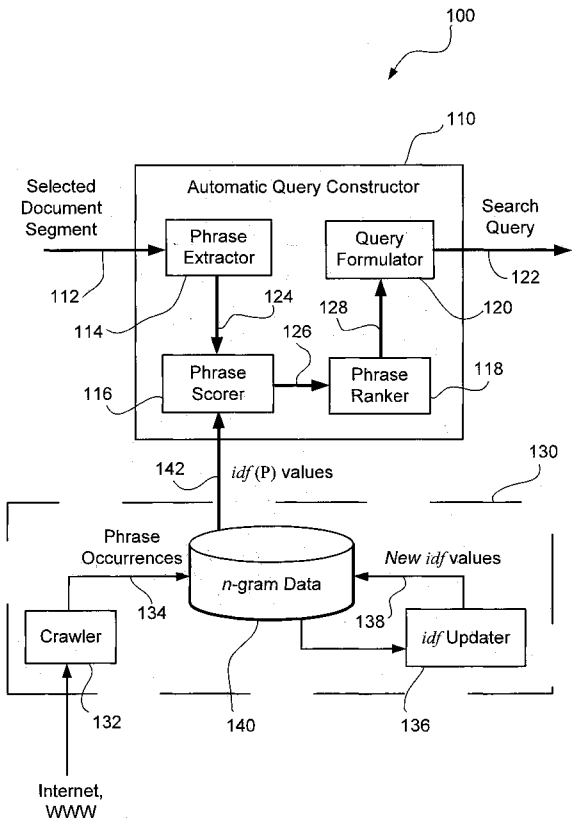
(72) Inventors: **DATTA, Anindya**; c/o National University of Singapore, School of Computing, Department of Information Systems, 21 Lower Kent Ridge Road, Singapore 119077 (SG). **KAJANAN, Sangaralingam**; c/o National University of Singapore, School of Computing, Department of Information Systems, 21 Lower Kent Ridge Road, Singapore 119077 (SG). **DUTTA, Kaushik**; c/o National University of Singapore, School of Computing, Department of Information Systems, 21 Lower Kent Ridge Road, Singapore 119077 (SG).

(74) Agent: **SPRUSON & FERGUSON (ASIA) PTE LTD**; P.O Box 1531, Robinson Road Post Office, Singapore 903031 (SG).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM,

[Continued on next page]

(54) Title: EFFICIENT AUTOMATIC SEARCH QUERY FORMULATION USING PHRASE-LEVEL ANALYSIS



(57) Abstract: Disclosed is a search query construction system (100) includes an n-gram database (140) of inverse document frequency data of phrases (*idf(P)*) in network locations and a phrase extractor (114) configured to extract candidate phrases (124) from an input document (112). A phrase scorer (116) is configured to determine a phrase frequency (*pf*) of a candidate phrase in the input document and, with at least the inverse document frequency data of phrases from the n-gram database, to form a score (*pfidf*) for each of the candidate phrases. A phrase ranker (118) operates to rank the candidate phrases according to the scores, and a query formulator (120) transforms a predetermined number of top ranked candidate phrases into a search query associated with the input document.

WO 2014/046620 A1

TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *of inventorship (Rule 4.17(iv))*

Published:

— *with international search report (Art. 21(3))*

EFFICIENT AUTOMATIC SEARCH QUERY FORMULATION USING PHRASE-LEVEL ANALYSIS

CROSS-REFERENCE TO RELATED APPLICATIONS

[001] This application claims the benefit under 35 U.S.C. §119 of the filing date of United States Patent Application No. 61/703559, filed September 20, 2012, hereby incorporated by reference in its entirety as if fully set forth herein.

TECHNICAL FIELD

[002] The present invention relates to data and information searches and, in particular to formulation of effective search queries that improve upon a typical human weakness in appreciating the essence of search desired to be performed.

BACKGROUND

[003] Searching for information of interest online represents one of the most common consumer activities on the computer and communication networks, including the Internet and the World Wide Web (the "web"). Google, Inc., the largest search engine provider in the search market, serves as an illustration of the size, scale and pervasiveness of search.

(i) Google™ processes upwards of one billion search queries *per day*;

(ii) Google™, the number one website by traffic statistics, accounts for 78.7% of search engine market share (as of December 2011) and approximately 6% to 7% of daily global internet traffic ;

(iii) Google™'s revenues for 2011 were $37.9 billion, with approximately $11 billion of these revenues attributable to search advertisement revenue.

[004] Further, the market for internet searching is large enough that even search engines with smaller worldwide market shares (e.g., Yahoo™, Bing™) or a regional focus (e.g., Baidu™) are economically viable. Given the prevalence of searching in the course of daily activity around the world, it is not surprising that tremendous amounts of effort have been devoted, in both academia and industry, towards improving internet search.

[005] In much of this work, an overarching goal has been to improve the quality of search results. With respect to a search engine, responding to a typical search request involves a two-step procedure: (a) determining which documents are *relevant* in the context of a given query, and (b) ranking those documents in order of relevance. Innovative science has resulted in substantial advances in both of these steps, with the PageRank algorithm that spawned Google™ being an example of (b). The state of this work has reached a point where it is generally accepted that dramatic improvements in document retrieval and/or document ranking are now going to be difficult to achieve.

[006] Yet, from a consumer perspective, the search experience still has substantial room for improvement. Gaining a better understanding of users' needs has been cited as one of the top two outstanding problems in internet searching. Google™ for example appreciate that if a user clicks on the number-one result of a search, without searching further, then the user has received what they wanted. By contrast if the user is observed to scroll down a search result, page after page, and reformulating the query, then it is apparent that the results were not what the user wanted.

[007] Of course, by this point, it is too late – the user is already frustrated. A closer look at the search process can reveal the sources of this frustration. There are two key components to the search process from the consumer's perspective: (a) formulating the search query, and (b) receiving the responses for the query from the search engine. Both these components must be high-quality to result in a quality search – it requires both good problem formulation (query formulation) as well as good problem solving (retrieving and ranking relevant documents) to lead to good outcomes (satisfied users). If a user is not receiving the desired results, there is clearly a problem in at least one of these steps. The server side of this solution, i.e., the "responding to the query" component, has been investigated to a point where substantial improvement will be difficult to achieve. Intuitively therefore, it would appear that current user frustration with searching is likely to be rooted in the query formulation stage – if the user does not provide a good-quality query, it is unreasonable to expect a search engine to provide the expected results. Substantial research evidence has shown that a critical determinant of search result quality is the quality of the search term.

[008] Substantial research and industry investigations have investigated how effective humans are, in general, at formulating effective search queries. From this work, there exists conclusive evidence showing that humans tend to be quite ineffective at formulating good search queries, resulting in the repeated query reformulation and iterative search behavior discussed above. Many reasons are cited for this, including:

(i) humans often search in domains where they have little expertise and thus might not know which terms better capture the essence of the topic of interest, and

(ii) humans tend to be biased towards search terms that are either too general or too specific.

[009] All of this makes a compelling case for disburdening the user of the query formulation task. However, to date, there has been little work directly addressing automatic query formulation. A primary area of related work describes *keyword extraction techniques*, including both generic approaches, as well as those specifically aimed at improving query relevance on the internet.

Generic Keyword Extraction Techniques

[0010] Generic techniques for keyword generation typically apply statistical and machine-learning approaches. Three most widely-cited examples are: (a) a practical automatic key phrase extraction approach; (b) a statistical corpus-based term extractor; and (c) an approach based on genetic learning.

[0011] The Practical Automatic Key Phrase Extraction (KEA) algorithm is a method for automatically extracting key phrases from text. KEA identifies candidate key phrases using lexical methods, calculates feature values for each candidate, and uses a machine-learning algorithm to predict which candidates are good key phrases. The Statistical Corpus-Based Term Extractor method describes a language-independent statistical corpus-based term extraction algorithm. Initially, the algorithm collects bigram frequencies from a corpus, and extracts two-word candidates. It then uses mutual information and log-likelihood ratios to extend them to multi-word terms. Using perplexity, it quantifies the definition of a term to obtain a comparative evaluation of term extraction algorithms. Perhaps one of the best known methods for keyword extraction is the

GenEx system, which is a rule-based key phrase extraction system with twelve parameters tuned using a genetic algorithm.

Keyword Extraction for Query Improvement

[0012] Term frequency-inverse document frequency (commonly referred to as $tfidf$) is a measure of a term's importance in a corpus. "Term frequency" (tf) describes how often a term (a single word) occurs within a document, while "inverse document frequency" (idf) represents the proportion of documents in the corpus that contain the word. While tf gives a gross measure of term occurrence, idf provides a way to promote the importance of terms that occur relatively rarely across the documents in the corpus, based on the intuition that rarer words are more discriminating than more common words.

[0013] Recently, methods have been proposed for using $tfidf$ for document-driven search query generation. These methods, like many others in keyword extraction using $tfidf$ measures, estimate the $tfidf$ score of a phrase by summing the $tfidf$ scores for the all the individual words in the phrase. A phrase is a group of at least two words that function to as a single constituent in a sentence.

[0014] Others have explored the domain of keyword extraction from a news source, to automatically drive queries. In particular, query terms were extracted from the closed captioning of television news stories to drive a search system that retrieved related online news articles. Because of the nature of broadcast news programs, where boundaries between topics are not explicit, improvements were found by using a history feature that automatically detected topic boundaries.

[0015] Other researchers have proposed a method for transforming a query patent into search queries for related patents. This work combined three types of search features for automatic query generation for patent research.

[0016] A well-studied technique to improve query relevance involves multiple interactions between an individual and the key word search system to iteratively refine the search results. In this method, the user provides feedback at each step, marking returned documents as relevant or not relevant. The iterative query refinement process continues until no more feedback is provided.

[0017] The Yahoo™ term extraction method generates phrases that capture the general topical area of an input document, but that are not specific enough to retrieve documents that will be deemed relevant by the users. The extracted terms could be used as general-purpose tags for summarizing and classifying the document in broad classification taxonomy, but are typically inadequate when used as queries due to their generality.

SUMMARY

[0018] The present disclosure is directed to impacting the quality of search by automatically generating queries for users. Typically a user's search interest is prompted by text he or she has already encountered. Suppose, for example, that a user is reading about how U.S. campaign finance laws have changed over the past century, and wishes to find more information about the effects of a recent U.S. Supreme Court decision that removed limitations on corporate funding for candidates. Ideally, the user should simply point to a portion of text that represents the topic of interest, and a system should construct a high-quality query that captures the essence of the input text. Presently disclosed is a configuration and implementation of a system that addresses this objective, and experimentally demonstrate the quality of the described implementation.

[0019] In many applications, the generation of good queries, in and of itself, is not sufficient to substantially enhance the searching experience. However, if it is desired to impact the internet search space in general, then any solution must scale to provide online responsiveness under internet-sized loads. Thus, it is desirable to develop a method for generating search queries that both produces high-quality queries and scales well under substantial loads.

[0020] The present disclosure includes:

(i) an analysis of current methods for automatic query generation, and the identification of potential dimensions for improving the quality of search queries. Based on this analysis, a new method of query ranking based on phrase-level analysis, rather than the single-term estimation methods used in existing approaches, is proposed.

(ii) an architecture and method for automatically generating search queries, based on a set of input text.

(iii) experimental testing of the architecture and method across a number of quality dimensions of interest. A set of relevance and run-time performance experiments are performed which demonstrate that the disclosed approaches produces high-quality queries, and that the present implementation scales well under substantial loads. Specifically, the presently disclosed system produces relevant search terms with roughly two-thirds precision and recall compared to search terms selected by experts, and that typical users find significantly more relevant results (31% more relevant) more quickly (64% faster) using the present system than self-formulated search queries. Further, the proposed implementation can scale to request loads of up to ten requests per second within current online responsiveness expectations (sub-2-second response times, even at the highest loads tested).

[0021] In accordance with one aspect of the present invention, there is provided a search query construction system comprising:

an n-gram database of inverse document frequency data of phrases in network locations;

a phrase extractor configured to extract candidate phrases from an input document;

a phrase scorer configured to determine a phrase frequency of a candidate phrase in the input document and, with the inverse document frequency data of phrases from the n-gram database and a metric of quality of the candidate phrases, form a score for each of the candidate phrases;

a phrase ranker to rank the candidate phrases according to the scores; and

a query formulator configured to transform a predetermined number of top ranked candidate phrases into a search query associated with the input document.

[0022] According to another aspect of the present invention, there is provided an automated query constructor comprising:

a phrase extractor configured to extract candidate phrases from an input document;

a phrase scorer configured to determine a phrase frequency of a candidate phrase in the input document and to use at least inverse document frequency data of phrases obtained from an n-gram database and the determined phrase

frequency to form a score for each of the candidate phrases;

a phrase ranker to rank the candidate phrases according to the scores; and

a query formulator configured to transform a predetermined number of top ranked candidate phrases into a search query associated with the input document.

[0023] Other and alternative aspects of the present invention are also disclosed.

BRIEF DESCRIPTION OF THE DRAWINGS

[0024] At least one embodiment of the present invention will now be described with reference to the accompanying drawings and tables, in which:

[0025] Figure 1 is a schematic block diagram representation of an architecture for automated search query generation.

[0026] Figure 2 is a plot of Precision versus Affinity Tuner for *WQF*.

[0027] Figure 3 is a plot of Recall versus Affinity Tuner for *WQF*.

[0028] Figure 4 is a plot of Precision versus Document length for *WQF*.

[0029] Figure 5 is a plot of Recall versus Document Length for *WQF*.

[0030] Figure 6 is a histogram of Search Results Relevance.

[0031] Figure 7 is a histogram of Time to Search.

[0032] Figure 8 is a plot of Response Time versus Input Load.

[0033] Figure 9 is a plot of response Time versus Length of Document.

[0034] Figure 10 is a schematic block diagram representation of a computer system in which the architecture of Figure 1 may be implemented.

[0035] Tables 1 to 7 arranged at the end of this detailed description.

DETAILED DESCRIPTION INCLUDING BEST MODE

Proposed Solution Approach

[0036] It is desired, given a set of input text from a user, to determine a new search query that will identify highly-related documents. Intuitively, it is desired to identify the *most relevant* and *most specific* phrases from the input text to compose a new search query.

[0037] With respect to *relevance*, tf and idf, form the basis of the most prevalent relevance measure in current use. Originally, tf and idf measures were proposed to measure the relevance of a single word in the context of a document or set of documents. Information retrieval researchers realized that similar measures were required for phrases. The most commonly used phrase relevance measure based on tf and idf simply sums the product of tf and idf for each word in an input phrase P. Such a sum may be expressed as:

$$tfidf(P) = \Sigma_{\forall w_i \in P} tf(w_i) \times (idf(w_i)).$$

More formally (and following known notation and mirroring the notation used herein and described in Table 2):

$$tfidf(P) = \Sigma_{\forall w_i \in P} f_D(w_i) \times \left(log \left(\frac{|D|}{df(w_i)} \right) \right),$$

where the first term in the expression represents tf and the second term represents idf. The tf value will vary from document to document, e.g., the word "president" may occur ten times in one document, and just twice in another. Because of this variation, a $tfidf$ score is specific to a document, and not generically attributable to the phrase. The idf for a word w is based on the frequency of w within a corpus of documents, and does not vary from document to document.

[0038] This method of summing the $tfidf$ scores for individual words to generate a relevance score for a phrase effectively estimates the relevance of the phrase, since it does not look at a phrase as a whole, only the constituent parts of the phrase. Proposed herein is a more accurate phrase-level relevance ranking method that does not require estimation. This new scoring method is termed by the present inventors as "phrase frequency-inverse document frequency (*pfidf*)," denoted $pfidf$. In the new method, the traditional $tfidf$ measure is modified in two ways to thereby avoid the summation in traditional estimated $tfidf$ phrase measures.

[0039] In a first modification, the phrase frequency pf of an entire phrase in a document is computed. For example, if it is desired to find the phrase frequency of the phrase "president obama," a count of the occurrences of that specific phrase within the document is performed, rather than using the frequencies of "president" and "obama" separately. The pf of a phrase is denoted as $pf = f_D(P)$, where f_D refers to the frequency of occurrence of a phrase P in document D.

[0040] In a second modification, the phrase idf values are computed directly from a large corpus of n-gram data. This is a significant difference from existing estimation methods, because such permits removal of the estimation step, thereby generating a true, accurate inverse document frequency measure for a phrase.

[0041] Formally, the idf for a phrase P, is computed as follows:

$$idf(P) = log_2 \frac{|D|}{df(P)}$$

where $|D|$ is the total number of documents in the corpus and $df(P)$ is the document frequency of phrase P, namely the number of documents in the corpus that contain P.

[0042] Consider an example of idf for the phrase "president obama". This is derived not by approximation, using the individual idf values for the words "president" and "obama", but rather by using occurrence information of the actual phrase in the corpus described herein. (The n-gram data are drawn from the Wordster n-gram Corpus (Wordster, Inc., see http://www.crunchbase.com/company/wordster), which is described in detail in the *Solution Details* section below.) Currently, the inventors have crawled 1,466,692,280 documents, out of which the phrase "president obama" appears in 304,235 documents, making of that phrase idf equal to 12.235.

[0043] Once the pf and idf for a phrase are known, there is no need according to the present disclosure for the previously described summing step, because $pfidf$ is based on a phrase as a whole, not on the individual words in the phrase. Detail of the calculation of $pfidf$ is described later with reference to Algorithm 1.

[0044] A study was performed comparing estimated $tfidf$ and $pfidf$ scoring methods to assess the suitability of these methods for ranking phrases as search query terms, and to see how the scoring differs. A set of 300 documents drawn from the "Most Popular" news stories from CNN's web site were examined by selecting the top ten stories at the same time each day for 30 days. Noun phrases from each document were then extracted (to be described in the *Solution Details* section below), and each phrase was scored using the estimated $tfidf$ and $pfidf$ metrics. For the estimated $tfidf$ metric, the term tf for each word w_i is the frequency of the word within the phrase's parent document, while the idf for w_i is calculated based on the single-word frequencies for w_i in the Wordster n-gram corpus. Values for $pfidf$ were calculated as described above. After calculating the estimated $tfidf$ and $pfidf$ scores for the noun phrases, the data was analysed.

[0045] An examination of the study data revealed that the estimated $tfdf$ method showed systematic biases towards long phrases. Intuitively, this is easy to explain. The length bias derives from the way phrase $tfidf$ scores are computed: by summing the $tfidf$ values of the individual words comprising a phrase. For instance, the $tfidf$ score of the phrase "United States" in a document D is given by

$$tfidf_D(United\ States) = tfidf_D(United) + tfidf_D(States),$$

where the idf scores for the individual terms are calculated based on their single-word frequencies in the n-gram corpus. By this method, any k-word phrase will result in a score higher than that of any of its m-word sub-phrases ($m \in [1, k-1]$). This in turn leads to the propensity for longer phrases to score higher than shorter phrases.

[0046] A demonstration of this length bias may be made using an example. Consider the CNN story located at http://edition.cnn.com/2011/11/03/us/china-russia-industrial-espionage/index.html?hpt=ias_c2, which discusses alleged industrial espionage. A phrase extractor according to the present disclosure was run on this story, and scored the resultant noun phrases by their $tfidf$ values. The top phrases based on summation of individual term $tfidf$ values are listed in Table 1, along with the individual $tfidf$

values for each word in the phrase. The last column shows the phrase-level $pfidf$ values.

[0047] Examination of the data in Table 1 revealed that, firstly, the estimated $tfidf$, generated by summing the word level $tfidf$, is much higher than the $pfidf$ values generated from a phrase level corpus. This is an artifact of how the two frequency scoring methods compute scores, and is not significant. Second, and more interestingly, the order of phrases based on estimated $tfidf$ values is completely different from the order based on the $pfidf$ scores, which are based on phrase frequencies in the input text and the actual phrase idf values in a corpus. The ordering of the $tfidf$ scores demonstrates the bias in favor of long phrases (word phrases with length 4 or more, which are typically considered long phrases) over shorter ones – where the list is sorted almost entirely by length of phrase. Of the top five phrases in the 18-phrase list based on estimated $tfidf$ values, 60% of them may be classified as long, because there is only one 2-word phrase and one 3-word phrase, but there are three phrases of length 4. At the same time if the phrases are sorted based on $pfidf$ values, only one phrase with 4 words, i.e., only 20% of the top five phrases, will be categorized as a long phrase.

[0048] The estimated $tfidf$ methods tend to favor long phrases. This is useful in many typical phrase-extraction application scenarios. For instance, in document summarization, long phrases (including full sentences) are desirable. However, long search phrases are not good search query candidates, since the probability of finding matching results is reduced as length increases. Thus, a bias toward longer length actually introduces diminished quality in the context of search terms.

[0049] Considering the issue of *specificity,* intuitively it is reasonable to expect more specific words to occur less frequently in a corpus, and in fewer documents, than more general words. These less-frequently occurring words, when included in a search query, tend to describe more specific concepts, which would be expected to generate more selective results.

[0050] The inventors conducted an analysis of the 300-document set of news stories discussed above, specifically looking for rare words in the phrases ranked by $tfidf$ and $pfidf$. An analysis of the data revealed that both $tfidf$ and $pfidf$ could indeed identify phrases containing rare words. The rarity of a word, i.e., the uncommonness of the word in the background corpus, is measured by the idf of the word: the higher the idf, the more uncommon the word. Clearly, all other influences remaining constant, the rarer the word, the larger will be the $tfidf$ and $pfidf$ scores for the word. By corollary, phrases containing rare words would tend to score higher as well.

[0051] The inventors considered how well each ranking method performs in this regard. For the case of $tfidf$, it was observed that the length bias (described above) adds a tendency toward longer phrases, such that the rare words are obscured within longer phrases. Rare words are not as useful as search terms when contained in long phrases, because these phrases are less likely to recur verbatim in related documents. For instance, suppose a user is interested in searching for and identifying articles related to the story in the document found at http://edition.cnn.com/2010/WORLD/africa/07/13/uganda.explosives.found/index.html?hpt=T1&fbid=buenNlC0q_a, which reports on explosives in Uganda. It is likely the many documents regarding this event will reference "Yoweri Musevini," the Ugandan President. It would be useful, therefore, to get "Yoweri Musevini" as a high scoring candidate phrase. Yet, in the $tfidf$ scoring of the document, such offered these words contained only inside long phrases. For example, $tfidf(\text{“}ugandan\ president\ yoweri\ museveni) = 24.65 + 4.11 + 14.82 + 14.40 = 57.98$, while $tfidf(\text{“}yoweri\ museveni\text{"}) = 14.82 + 14.40 = 29.223.$

[0052] In contrast, the $pfidf$ ranking placed "yoweri musevini" in the top phrases for the input document, ahead of the longer phrase "ugandan president yoweri museveni." Specifically,

$pfidf(\text{“}ugandan\ president\ yoweri\ museveni\text{"}) = 16.36,$

while $pfidf(\text{“}yoweri\ museveni\text{"}) = 17.84.$

[0053] While $pfidf$ does provide improvements over $tfidf$ in identifying phrases consisting primarily of rare words, the analysis by the inventors across the 300 news documents showed that the $pfidf$ scores did not substantially

distinguish rare terms over containing rare words along with additional terms (17.84 vs. 16.36 in the example above). This led the inventors to further tune the scoring method to ensure generation of the most specific phrases possible for search queries by using the phrase-level frequency data. Intuitively, if certain words in a phrase occur often in the presence of another word, as opposed to the individual words' frequencies of occurrence in other phrases, they have high *affinity* for each other. For example, consider the phrase "nuclear power". Both of the constituent words are in wide usage in a variety of contexts: "nuclear" will occur widely across domains in physics, engineering, power generation, and many others. "Power" is a very generic term, which could describe electrical power, political power, or other semantic meanings. Consider now the phrase "nuclear non-proliferation." While "nuclear" still occurs across many domains, "non-proliferation" occurs in virtually no other context. Here, the phrase "nuclear non-proliferation" would have a higher *affinity* than "nuclear power". The present inventors therefore propose a metric called $Affinity$, which measures how common the specific n-gram occurrence is in relation to other n-grams with overlapping constituent words to help further refine the quality of candidate phrases for search queries. As such, *Affinity* is a metric of the quality of the candidate noun phrase based upon a frequency of occurrence of words in the phrase with other words in the phrase. This is described in detail in the *Solution Details* section below.

[0054] A summary of the notation use in this patent specification is provided in Table 2 located at the end of the detailed description.

Architecture and Method Overview

[0055] Fundamentally, the arrangements presently disclosed involve two types of processing: (a) crawling the web to gather and update n-gram frequency data; and (b) responding to input text documents with a relevant search query derived from the n-gram frequency data. The processing for (a) can take place offline through continuous background processing. More significantly, the processing for (b) is time-sensitive, in that users expect a fast response to an input request. Based on this, a system architecture 100 is depicted in Figure 1 and formed using two main parts: (1) a set of modules 130 dedicated to offline

processing for gathering and updating n-gram frequency data; and (2) an automated query constructor 110 for generating search query requests. As seen in Figure 1, the automatic query constructor 110 receives a selected document segment 112 from a user or associated application and, with *idf* values 142 from the n-gram module 130, formulates a search query 122 which can be input to a search engine for the generation of search results.

[0056] Figure 10 depicts a general-purpose computer system 1000, upon which the presently disclosed arrangements can be practiced to implement the architecture of Figure 1.

[0057] As seen in Figure 10, the computer system 1000 includes: a computer module 1001; exemplary input devices such as a keyboard 1002, a mouse pointer device 1003, a scanner 1026, a camera 1027, and a microphone 1080; and exemplary output devices including a printer 1015, a display device 1014 and loudspeakers 1017. An external Modulator-Demodulator (Modem) transceiver device 1016 may be used by the computer module 1001 for communicating to and from a communications network 1020 via a connection 1021. The communications network 1020 may be a wide-area network (WAN), such as the Internet, a cellular telecommunications network, or a private WAN. Where the connection 1021 is a telephone line, the modem 1016 may be a traditional "dial-up" modem. Alternatively, where the connection 1021 is a high capacity (e.g., cable) connection, the modem 1016 may be a broadband modem. A wireless modem may also be used for wireless connection to the communications network 1020.

[0058] The computer module 1001 typically includes at least one processor unit 1005, and a memory unit 1006. For example, the memory unit 1006 may have semiconductor random access memory (RAM) and semiconductor read only memory (ROM). The computer module 1001 also includes an number of input/output (I/O) interfaces including: an audio-video interface 1007 that couples to the video display 1014, loudspeakers 1017 and microphone 1080; an I/O interface 1013 that couples to the keyboard 1002, mouse 1003, scanner 1026, camera 1027 and optionally a joystick or other human interface device (not illustrated); and an interface 1008 for the external modem 1016 and printer 1015.

In some implementations, the modem 1016 may be incorporated within the computer module 1001, for example within the interface 1008. The computer module 1001 also has a local network interface 1011, which permits coupling of the computer system 1000 via a connection 1023 to a local-area communications network 1022, known as a Local Area Network (LAN). As illustrated in Figure 10, the local communications network 1022 may also couple to the wide network 1020 via a connection 1024, which would typically include a so-called "firewall" device or device of similar functionality. The local network interface 1011 may comprise an Ethernet circuit card, a Bluetooth™ wireless arrangement or an IEEE 802.11 wireless arrangement; however, numerous other types of interfaces may be practiced for the interface 1011.

[0059] The I/O interfaces 1008 and 1013 may afford either or both of serial and parallel connectivity, the former typically being implemented according to the Universal Serial Bus (USB) standards and having corresponding USB connectors (not illustrated). Storage devices 1009 are provided and typically include a hard disk drive (HDD) 1010. Other storage devices such as a floppy disk drive and a magnetic tape drive (not illustrated) may also be used. An optical disk drive 1012 is typically provided to act as a non-volatile source of data. Portable memory devices, such optical disks (e.g., CD-ROM, DVD, Blu-ray Disc™), USB-RAM, portable, external hard drives, and floppy disks, for example, may be used as appropriate sources of data to the system 1000.

[0060] The components 1005 to 1013 of the computer module 1001 typically communicate via an interconnected bus 1004 and in a manner that results in a conventional mode of operation of the computer system 1000 known to those in the relevant art. For example, the processor 1005 is coupled to the system bus 1004 using a connection 1018. Likewise, the memory 1006 and optical disk drive 1012 are coupled to the system bus 1004 by connections 1019. Examples of computers on which the described arrangements can be practised include IBM-PC's and compatibles, Sun Sparcstations, Apple Mac™ or a like computer systems.

[0061] The architecture of Figure 1 and the methods to be described may be implemented using the computer system 1000 wherein the processes to be

described, may be implemented as one or more software application programs 1033 executable within the computer system 1000. In particular, the process and method steps are effected by instructions 1031 in the software 1033 that are carried out within the computer system 1000. The software instructions 1031 may be formed as one or more code modules, each for performing one or more particular tasks. The software may also be divided into three separate parts, in which a first part and the corresponding code modules performs the *n*-gram data generation methods 130, a second part and the corresponding code modules performs the automated query construction 110, and a third part and corresponding code modules manage a user interface between the second part and the user, recalling that the first part is configured to operate autonomously of any particular user.

[0062] The software may be stored in a computer readable medium, including the storage devices described below, for example. The software is loaded into the computer system 1000 from the computer readable medium, and then executed by the computer system 1000. A computer readable medium having such software or computer program recorded on the computer readable medium is a computer program product. The use of the computer program product in the computer system 1000 preferably effects an advantageous apparatus for search query formulation.

[0063] The software 1033 is typically stored in the HDD 1010 or the memory 1006. The software is loaded into the computer system 1000 from a computer readable medium, and executed by the computer system 1000. Thus, for example, the software 1033 may be stored on an optically readable disk storage medium (e.g., CD-ROM) 1025 that is read by the optical disk drive 1012. A computer readable medium having such software or computer program recorded on it is a computer program product. The use of the computer program product in the computer system 1000 preferably effects an apparatus for search query formulation.

[0064] In some instances, the application programs 1033 may be supplied to the user encoded on one or more CD-ROMs 1025 and read via the corresponding

drive 1012, or alternatively may be read by the user from the networks 1020 or 1022. Still further, the software can also be loaded into the computer system 1000 from other computer readable media. Computer readable storage media refers to any non-transitory tangible storage medium that provides recorded instructions and/or data to the computer system 1000 for execution and/or processing. Examples of such storage media include floppy disks, magnetic tape, CD-ROM, DVD, Blu-ray™ Disc, a hard disk drive, a ROM or integrated circuit, USB memory, a magneto-optical disk, or a computer readable card such as a PCMCIA card and the like, whether or not such devices are internal or external of the computer module 1001. Examples of transitory or non-tangible computer readable transmission media that may also participate in the provision of software, application programs, instructions and/or data to the computer module 1001 include radio or infra-red transmission channels as well as a network connection to another computer or networked device, and the Internet or Intranets including e-mail transmissions and information recorded on Websites and the like.

[0065] The third part of the application programs 1033 and the corresponding code modules mentioned above may be executed to implement one or more graphical user interfaces (GUIs) to be rendered or otherwise represented upon the display 1014. Through manipulation of typically the keyboard 1002 and the mouse 1003, a user of the computer system 1000 and the application may manipulate the interface in a functionally adaptable manner to provide controlling commands and/or input to the applications associated with the GUI(s). Other forms of functionally adaptable user interfaces may also be implemented, such as an audio interface utilizing speech prompts output via the loudspeakers 1017 and user voice commands input via the microphone 1080.

[0066] Typically, the modules 130 for the formation of the *n*-gram data are implemented in a server computer 1050 accessible to the automatic query constructor 110, which may be implemented in a server computer, such as the same server 1050, or in the computer module 1001 operating as a server. In such an instance, a user operating a client device 1060, such a personal computer, tablet device or smartphone can, via an add-on associated with a browser application, invoke the operation of the automated query constructor 110 to generate the search

query 122 that is provided to the browser application as an input to a search engine. The browser application can then receive and display the search results, for example via the display 1014.

[0067] The main parts of the architecture 100 will now be described.

Offline Processing: Gathering and Updating n-gram Frequency Data

[0068] The offline processing portion 130 of the architecture 100 is shown formed of three modules: (a) a crawler 132; (b) a data store 140 for n-gram data; and (c) an idf-updater module 136. The crawler 132 forms and provides phrase occurrences 134 to the store 140 and the *idf*-updater 136 provides new *idf* values to the store 140, which in turn outputs, on request *idf* values 142 of phrases, $idf(P)$, to the constructor 110. In specific implementations, the offline processing portion 130 may be afforded by a third-party provider distinct from a provider of the automated query constructor 110.

[0069] The crawler 132 is configured to periodically or continuously crawl a specified set of network locations such as websites to gather at least phrase frequency data of text in the websites, updating the frequency data in the n-gram data store 140 for phrases, and most preferably for phrases up to $n = 4$ (i.e. phrases having at least 2 and no more than 4 words). Where desired the crawler 132 may further provide to the store 140 frequency data for individual words. Since idf scores are based solely on the frequency of a phrase in the corpus, the architecture 100 is configured to pre-compute and store idf values with the phrases. The idf-updater module 136 may be configure to operate periodically, such a once per day, to interpret phrase occurrence data in the store 140 and to update the idf values for the phrases in the n-gram data store 140.

Online Processing: Responding to Automated Search Query Generation Requests

[0070] The Automatic Query Constructor (AQC) module 110 represents an on-line portion of the architecture 100 and is responsible for responding to search query generation requests of users. The AQC module 110, as shown in Figure 1, contains four sub-modules: (a) a phrase extractor (PE) 114, (b) a phrase scorer (PS) 116, (c) a phrase ranker (PR) 118, and (d) a query formulator (QF) 120.

Each of these modules is described below, in the order in which they are invoked in the process of responding to a user request.

[0071] The AQC 110 is configured to receive a document, or more preferably a selected document segment 112 from a user. The selected document segment 112 is segment of contiguous text for example selected from a source document by a copy operation. A typical segment may be a paragraph of a news item about which the user wishes to conduct further searching. With the segment 112, the PE 114 extracts a set of candidate noun phrases 124 based on a known set of noun phrase patterns, as discussed below. For example, Table 1 discussed above lists several noun phrases, which were all drawn from the same news story, including "government intelligence services," "China and Russia," and "U.S. government."

[0072] The PS 116 then scores candidate phrases 124 using $pfidf$ and $Affinity$, the above described measures of relevance and specificity, respectively and determined from the *idf* values 142, using the scoring methods to be described below in the *Solution Details* section. At the end of this step, each of the candidate phrases is associated with a score, as represented by 126 in Figure 1. For example, based on the input text and the noun phrases identified in the previous step, the following scores might be assigned to the candidate noun phrases, as follows: "government intelligence services" (20.64), "China and Russia" (81.53), and "U.S. government" (16.29).

[0073] The PR 118 then selects the top ranked phrases 128. Here, the number of phrases selected depends on the total number of candidate phrases – a larger set of candidate phrases will return a larger set of output queries, the two being generally proportional. However, given a large set of candidate phrases, it is preferred to return a smaller set of the most relevant and specific phrases in the output search query. Following the above example, "China and Russia," with the highest score across the three phrases above, 81.53, would be returned in the output query, while "government intelligence services" and "U.S. government," which have much lower scores, might not.

[0074] Finally, the QF 120 transforms the set of candidate phrases 128 into a resulting set of search queries 122, performing any necessary transformations, such as removing overlapping terms or redundant terms.

Preferred Solution Details

[0075] In this section, details of the operation of the components shown in Figure 1 is described. First considered is the offline processing required to maintain n-gram frequency data, and then the method for responding to search query generation requests.

Building and Maintaining the n-Gram Frequency Corpus

[0076] The phrase-frequency approach described herein is for example implemented using the Wordster n-gram Corpus, which is a lexical corpus created by continuous crawling of a set of 42 well-defined websites. These websites are carefully selected to represent a variety of document categories (news, sports, lifestyle, etc.) and geographies (documents from all English-speaking countries). Moreover, the websites are so chosen as to represent internet content visited by about 95% of internet users (the sites crawled represent the top sites in their respective Alexa (www.alexa.com) categories). The current corpus represents the effort of over two years of continuous crawling – over 300 million documents and roughly 2 TB of raw data.

[0077] The n-gram database 140 stores n-gram data for phrases, including the phrase itself, the document frequency of the phrase (count of documents where the phrase appears), idf, and integer word ids (identities) representing each constituent word in the phrase. This data is collected in the store 140 in a database format for $n = 1$, $n = 2$, $n = 3$, and $n = 4$. Table 3 shows the schema of the 2-gram corpus (similar schemas store data for the other phrase lengths of interest). Also stored in the store 140 is a count of the total number of documents crawled, D, in a separate table.

[0078] The tables and indexes in the n-gram data store 140 are desirably configured with scalability in mind. Specifically, the tables are designed such that queries to support retrieval for idf and data required for $Affinity$ do not require joins – this results in very fast retrieval of the needed data.

[0079] According to the preferred n-gram storage schema, firstly, rather than computing idf values for phrases in real time, the modules 130 pre-compute the $idf(P)$ for all phrases in the n-gram corpus which is then stored in the database 140. This is possible, since all the data is readily available in the n-gram data store, and aids scalability in that the computation does not need to be done at runtime. Secondly, the constituent words in an n-gram phrase are preferably represented with integer word ids, where each unique word is associated with a unique integer word id. This representation allows for the matching of words across phrases based on numeric equivalence testing, rather than string comparison. This significantly reduces the processing required for string operations.

[0080] The crawler 132 is desirably a multi-threaded crawler module that retrieves documents from the selected web sites on a continuous basis, and updates the phrase frequency data 134 in the n-gram database 140. When a crawler thread finds a new document, the crawler thread extracts the noun phrases (for example using the same method used for extracting candidate phrases in request input text, which is described in the next section, together with how to respond to search query generation requests), maintaining a count of the occurrences of each noun phrase in the document. The crawler thread then updates the df for each phrase in the appropriate n-gram table in the database 140, and increments the count of documents crawled D by 1.

[0081] Since the crawled data is updated periodically and thus effectively constantly in the database 140, the *idf* module 136 is configured to update the relevant idf values 138 in the n-gram corpus 140. The module 136 considers each n-gram phrase individually, calculating the idf value of the n-gram as follows:

$$idf = \left(log \frac{|D|}{df(P)} \right)$$

and then updates the phrase's idf value in the n-gram corpus database 140.

Responding to Search Query Generation Requests

[0082] Responding to a search query generation request is a four-step process: (a) extract candidate phrases 114; (b) score candidate phrases 116; (c)

determine the top candidate phrases 118; and (d) formulate the output search query 120.

Extracting Candidate Phrases

[0083] The candidate phrase extraction process performed by the phrase extractor 114 includes two steps: (a) part-of-speech tagging; and (b) noun phrase identification through part-of-speech patterns.

[0084] The phrase extractor 114 uses a part-of-speech tagger to attach a part-of-speech tag to each token (i.e., word) in the document of interest sample 112. A sampling of part-of-speech tags and their associated meanings can be found in Table 4. More precisely, the document 112 is parsed into sentences, which are then processed by the part-of-speech tagger. Supplied with a sentence, the tagger can produce an ordered list of part-of-speeches as output for each word in the sentence (such as noun, verb, adjective, etc.). For example, the sentence "The Netherlands beats Uruguay in today's match" would have the word 'The' tagged as a pre-determiner, 'Netherlands' tagged as a noun, 'beats' tagged as a verb, and so on. The tagging result may then be represented as "The/DT Netherlands/NN beats/VB Uruguay/NN in/CC today/NN 's/POS match/NN", where DT, NN, VB, CC and POS stands for pre-determiner, noun, verb, conjunction and possessive ending, respectively.

[0085] Nouns are considered especially important in present context as they refer to people, things, and the concepts about which the authors of source documents wish to communicate. Typically, a noun is flanked by determiners, modifiers and adjectives to form a noun phrase. The importance of noun phrases as descriptors is well-known and most phrase extraction methods concentrate on extracting noun phrases. A pattern-match approach is used to extract possible noun phrases.

[0086] From the ordered list of part-of-speech tags, all possible noun phrases are then extracted as candidate phrases. A base set of noun phrase patterns is used to choose sequences of words from the sentence whose part-of-speech pattern matches any of the base set of patterns. The Table 5

shows a small sampling of patterns and observed noun phrases found during noun phrase extraction in the examples discussed herein.

Scoring the Phrases

[0087] As noted above, it is desired to rank candidate phrases by attractiveness as search query terms. Desired are phrases that are relevant, such that the results are of interest to the user, and selective, so that the user is not overloaded with results. The preferred phrase-scoring strategy is based on the $pfidf$ and $Affinity$ measures outlined above. Algorithm 1, listed below describes a preferred scoring method in pseudo code. The pseudo-code is preferably implemented as a software application forming the phrase scorer 116 and able to be stored in the HDD 1010 and executed by the processor 1005 to implement the functionality of the phrase scorer 116.

[0088] **Algorithm 1: Scoring Process**

Input: Phrase Set = all Noun Phrases extracted from document D

Title Phrase Set = all Noun Phrases extracted from the title of document D

Top N Words= top N ranked words extracted from document D based on *pfidf* score

Output: Scored phrases

foreach *phrase P in Phrase Set* **do**
  if $f(P) + fD(P) < 4$ **then**
    $idf(P) = 0$
  end
  $Affinityc(P) = \frac{f(P)}{min_{\forall w_i \in P(f(w_i))}} \times (1 - \max(P1, P2))$;
  $S(P) = log2(1 + fD(P)) \times idf(P) + \delta \times Affinityc(P)$
  foreach *TP in Title Phrase Set* **do**
    if P = *TP and TP is a Proper Noun Phrase* **then**
      $S(P) = 3\beta \times S(P)$
    end
    if P = TP then

11 $S(P) = 2\beta \times S(P)$

12 **end**

13 **end**

14 **if** *P contains at least 2 Proper Nouns* **then**

15 $S(P) = \beta \times S(P)$

16 **end**

17 **end**

[0089] The preferred scoring function implemented by the phrase scorer 116 is formally represented with the expression:

$$S(P) = log_2 (1 + f_D(P)) \times idf(P) + \delta \times Affinity_c(P).$$

The first term of the score $S(P)$, being $log_2 (1 + f_D(P)) \times idf(P)$, represents the $pfidf$ score of a phrase P. Here, $f_D(P)$ is the frequency of the phrase P in the input document D 112. The term $log_2 (1 + f_D(P))$ is used to reduce the weight of repeated occurrence of an n-gram in the document D.

[0090] The idf value 142 for a phrase P is computed using :

$$idf(P) = \left(log_2 \frac{|D|}{df(P)} \right),$$

where $|D|$ is the total number of documents in the corpus and $df(P)$ is the document frequency of phrase P , namely the number of documents which contain phrase P in the corpus.

[0091] One special case that arises is the rare occasion when the input document 112 yields a phrase that cannot be found in the corpus 140, or has an extremely low frequency of occurrence in the corpus 140. The temptation in such cases is to discard the phrase as irrelevant. However, one of the key motivations of the overarching Wordster project is to capture the evolution of linguistic patterns. As a result, a new phrase that is observed has some likelihood of being relevant. To account for this, a heuristic has been developed by which the pf of the phrase (the phrase's frequency in the input document) is added to the frequency of occurrence of the phrase in the corpus 140. If this sum is greater than or equal to 4, then the phrase is added to the corpus and retained as a candidate phrase. This is seen in Algorithm 1, at lines 2 to 4.

[0092] The second term in $S(P)$, being $\delta \times Affinity_c(P)$, is used to modify the score represented by the first term to help to boost the selectivity of the generated search query by boosting scores for rare, but meaningful, phrases.

[0093] $Affinity$ measures the likelihood of co-occurrence of the constituent words in a phrase. Intuitively, if certain words in a phrase occur often in the presence of one another, they have high $Affinity$. For example, consider the following sentence: "Attention deficit hyperactivity disorder is more common in boys than girls, and it affects 3-5 percent of children in the United States." The phrase extractor 114 will extract "attention deficit hyperactivity," "attention deficit hyperactivity disorder," and "deficit hyperactivity disorder" as some of the noun phrases in this sentence. Intuitively, based on the set of noun phrases extracted, the most meaningful phrase in this phrase set is the 4-gram phrase "Attention Deficit Hyperactivity Disorder," as compared to the other 3-gram phrases, e.g., "attention deficit hyperactivity" and "deficit hyperactivity disorder".

[0094] Formally, $Affinity$ is defined as:

$$Affinity_c(P) = \frac{f(P)}{min_{\forall w_i \in P(f(w_i))}} \times (1 - \max(P1, P2)) ,$$

where $f(P)$ is the frequency of phrase P in the corpus and $min(f(w_i))$ is the minimum frequency across the words in phrase P. The term $(1 - max(P\,1, P\,2))$ computes the relevance of the phrase with respect to its neighborhood. If there is a pre-word for phrase 'P' and the pre-word is not a stop word, then the pre-word is combined with phrase 'P' to generate the new phrase called P_{pre}. If there is a post word for phrase 'P' and the post word is not a stop word then the post-word is combined with phrase 'P' to generate the new phrase called P_{post}. If a pre-word or post-word is $null$ then $P1,\ P2 = 0$. Values of $P1,\ P2$ are measured as follows, $P1 = \frac{f(P_{pre})}{1+f(P)}$ and $P2 = \frac{f(P_{post})}{1+f(P)}$. The higher the value of $\left(1 - max(P1, P2)\right)$ is, the less relevant the word phrase is with respect to its neighborhood.

[0095] Returning to the example, in the corpus 140, the words "attention", "deficit", "hyperactivity" and "disorder" occurs 774,404, 255,626, 218,686 and 87,131 times respectively. The phrases "attention deficit hyperactivity," "deficit hyperactivity disorder," and "attention deficit hyperactivity disorder" occur 9,478,

9,389, and 9,389 times, respectively. When the phrase scorer 116 calculates $Affinity$ for the phrase "attention deficit hyperactivity", the phrase scorer 116 checks the pre-word and post-word of the phrase in the document, which is labeled during the noun phrase extraction step. For this phrase, the pre-word is "null" and the post-word is "disorder." Since the pre-word is $null$, $P1 = 0$ and $P2$ will be $\frac{9389}{9479} = 0.99$. Therefore $Affinity(\textit{"attention deficit hyperactivity"}) = \frac{9389}{9479} \times (1 - max(0, 0.99)) = 0.0108$. Likewise $Affinity(\textit{"deficit hyperactivity disorder"}) = 0$ and $Affinity(\textit{"attention deficit hyperactivity disorder"}) = 0.107$. Therefore, among these phrases, "attention deficit hyperactivity disorder" will be chosen as phrase with highest $Affinity$ and is more meaningful than other similar phrases.

[0096] The concept of $Affinity$ requires phrases of at least two words in length, since *Affinity* is a measure of the relationship between words. *Affinity* is undefined for single-word phrases. For single-word phrases, the second term of $S(P)$ is set to 0.

[0097] Now consider the term δ. Ideally, the first and second terms of $S(P)$ should contribute to the score on an equal footing. However, the values generated by each term lie within limited non-overlapping ranges. A leveling tuner δ is used to bring $Affinity$ values into the same range as $pfidf$.

[0098] Finally, some noun phrases, by virtue of their position in a title or type (proper nouns), deserve additional weighting. These phrases may be overweighted using an weighting factor β, where $\beta > 1$. This is seen in Algorithm 1, at lines 7 to 16. Values of β are applied as follows:

(i) If a noun phrase contains two or more proper nouns, $S(P)$ is multiplied by a factor of β.

(ii) If a noun phrase is appeared in both the content and title, $S(P)$ is multiplied by 2β .

(iii) If a noun phrase appears in both the content and title of a document, and it is a proper noun phrase, $S(P)$ is multiplied by 3β.

[0099] For example, the score for a noun phrase "Barak Obama" would be weighted as $\beta \times S$ ("Barak Obama") because the phrase contains two proper

nouns. In the same vein, if the noun phrase "government spending" appears in both the title and content (i.e. twice) in an article in the New York Times, then the score would desirably be weighted by $2\beta \times S("government\ spending")$.

Determining the Top Phrases

[00100] Having extracted and scored phrases 126, processing moves on to the Phrase Ranker 118 of the AQC 110, where the most relevant phrases in the input document 112 are determined. This is described in Algorithm 2 below. The pseudo-code of Algorithm 2 is preferably implemented as software stored in the HDD 1010 and executable by the processor 1005 to perform the functionality of the phrase ranker 118

[00101] **Algorithm 2: Extracting Top Phrases**

Input: Phrase Set =scored phrases extracted from document D
Output: $Top\ N$ Phrase Set

```
1  foreach phrase P in Phrase Set do
2      if Score(P) >= μ(S(Pi)) + β × γ(S(Pi))] then
3          add P into Top N Phrase Set
4      end
5      if size(s) of Top N Phrase Set < 10 then
6          add other (10 − s) highest P into Top N Phrase Set
7      end
8      if size(s) of Top N Phrase Set > 50 then
9          only keep top 50 P in Top N Phrase Set
10     end
11 end
```

[00102] In the phrase ranker 118, the number of phrases extracted depends on the length of the input document. It will be appreciated that, even for a relatively small input document 112 (about 1KB), a fairly large number of noun phrases are extracted and scored (roughly 100 on average). Since the purpose is to construct a single search query, it is appropriate to find a way to select a small number of these noun phrases (the "top phrases") (e.g. 128) for input to the query

formulator 120. To do this, the phrase ranker firstly calculates the mean (μ) and deviation (σ) of the scores of all phrases. In a preferred implementation, only those phrases whose scores are larger than $\mu(S(P_i)) + \rho \times \sigma(S(P_i))$, for all P_i in the candidate output set, are be selected into top phrase set. Here ρ is a tunable parameter, where a higher ρ value will result in greater selectivity and fewer phrases in the output query. Experiments conducted by the present inventors have found that $\rho = 2$ is a reasonable value for generating output search queries that are both selective and relevant. Assuming a normal distribution, this means that the top 2.5% of the phrases will be selected for further processing. To avoid very large and very small output sets, the number of phrases returned is restricted to no less than 10 phrases and no more than 50 phrases.

Query Formulation

[00103] The input 128 for the query formulation step is the top phrases identified by the phrase ranker 118, which form the base to construct a query 122. The main purpose of query formulation 120 is to remove redundancies across top phrases. For example, the phrases "government" and "government spending" might both be identified as top phrases, and the two phrases overlap, where the phrase "government" conveys more general information than "government spending." In general, it is preferred to choose one for inclusion, with a bias toward shorter phrases. A preferred query formulation method is described in Algorithm 3. The pseudo-code of Algorithm 3 is also preferably implemented as software stored on the HDD 1010 and executed by the processor 1005 to provide the functionality of the query formulator 120.

[00104] **Algorithm 3: Query Formulation**

Input: *Top N Phrase Set*
Output: *Top N Query Term Candidate Set*

```
1  /*get all collocates*/
2  foreach phrase P in Phrase Set do
3        look up all collocates Ci of phrase P within window size w
4        foreach Collocate Ci do
```

remove stop words; score C_i with $S(C_i)$
if C_i in *Collocate Set* **then**
Add C_i into *Collocate Set*
end
end
end
/*remove redundancy of collocates*/
Sort *Collocate Set* based on frequency and score of C_i Descending
Add C_1 ranked first in *Collocate Set* into *Top N Query Term Candidate Set*;
foreach *Collocate* C_i (i > 1) in *Collocate Set* **do**
flag = false
foreach *Collocate* C_i' **do**
if C_i *contains* C_i' and $f_D(C_i) > f_D(C_i')$ **then**
remove C_i and add C_i into *Top N Query Term Candidate Set*
flag = true and break
end
if C_i *contains* C_i' **then**
flag = true and break
end
end
if flag == false **then**
add C_i into *Top N Query Term Candidate Set*
end
end
Keep top (one third of size of *Top N Phrase Set*) *collocate* C_i' in *Top N Query Term Candidate Set*

[00105] To implement the query formulator 120, a configurable window size z is set to be the maximum length of a phrase contained in the final query. The present inventors consider a recommend setting to be $(z \leq 3)$. Each top phrase is then decomposed into all possible n-gram collocates, where $N \in [1, z]$. For example, for $z = 3$, the four-word phrase "all U.S. government spending" will be decomposed into the following n-gram collocates: "all U.S. government", "U.S.

government spending", "all U.S.", "U.S. government", "government spending", "all", "U.S.", "government" and "spending". The formulator 120 then removes any stop words, which are considered noise. After this step, the present example reveals "U.S. government spending", "U.S. government", "government spending", "U.S.", "government" and "spending" as query term candidates. Next, all query term candidates, i.e., the n-gram collocates identified thus far, are scored by formula $S(P)$, and then sorted in descending order based on score, as seen in Algorithm 3, at lines 2 to 13. Next, redundancies between query term candidates are removed.

[00106] Specifically, if collocate C_i (e.g., "government spending") contains collocate C_i' (e.g. "government") and the frequency of C_i in document is larger than γ (set to 0.5 here) times the frequency of C_i' in document, then C_i will be selected as the query term. Here, γ is a frequency threshold for showing preference to longer (and therefore rarer) phrases over their shorter sub-phrases when the longer phrase occurs frequently relative to the shorter phrase. For example, if $\gamma = 0.5$, the longer phrase would need to occur at a frequency of at least 50% of the sub phrase, so γ should be in the range $[0,1]$.

[00107] In addition, if C_i is already selected as the query term candidate, and C_i' is contained in C_i, C_i' will not be considered again, as seen from Algorithm 3, at lines 14 to 28. Finally, the query formulator 120 selects a predetermined number, such as the top third, of the n-gram collocates and concatenates them to form the output query, as seen in Algorithm 3, at line 29.

Analysis

[00108] In this section, the time complexity and storage complexity of the above described approach to search query formulation is discussed.

Time Complexity

[00109] Time complexity analysis begins with Algorithms 1 and 2, which describe (a) scoring each extracted phrase, and (b) selecting the top phrases. Let v be the number of phrases in the phrase set, u be the number of phrases in title phrase set and l be the number of phrases in top phrase set. In the first step, it takes $O(u \cdot v)$ to score each phrase. In the second step, it takes $O(n)$ to get

the top phrases. Thus, the time required for Algorithms 1 and 2 can be expressed as $O(u \cdot v)$.

[00110] For Algorithm 3, there are two steps: (a) getting all collocates for phrases in top phrase set, and (b) removing redundancy of collocates. Let k be the average length of phrases in the top phrase set and z be the window size (i.e., the maximum length of collocate). The time required to look up collocates in first step will be $O(z.(k-1))$. Let c be the number of collocates found in step (a). In step (b), sorting the collocate set takes $O(c\,.\,log(c))$, and removing the redundancies takes $O(c^2)$ in the worst case. Thus, the time required for Algorithm 1 can be expressed as $O(x \cdot (k-1)) + O(c \cdot log(c))$.

Storage Complexity

[00111] The storage 140 maintains a cache of all 1-gram statistics. For 1-gram data, the database system 140 will be used as a backup only in case there is a memory dump or new 1-gram word is added into the system. For n-gram $(n > 1)$ data, caching does not make sense due to the huge data sizes (3.5 GB for 2-gram data, 17.5 GB for 3-gram data and 38.3 GB for 4-gram data), in which case the cache hit ratio would be very low. In operation, the computer 1001 can cache most frequently accessed n-gram $(n > 1)$ data from the store 140 into the memory 1006 to speed up the processing for the most frequently used phrases, and retrieve less-frequently accessed data from database 140.

[00112] Thus, in-memory storage complexity amounts to ensuring that 1-gram statistics can be cached. The corpus of the database 140 contains 1,001,926 1-gram words. For each 1-gram word in a MySQL database management system, there is one INT type word id [, one double type idf value, one double type $Affinity$ value, and one big INT type $f(P)$ value. So the total size for 1-gram statistics data is (4 + 8 + 8 + 8) * 1001926 bytes, i.e. 28 MB.

Experimental Study

[00113] The present inventors have implemented the architecture 100 described above in a preferred form by what they term as the Wordster Query Formulator (WQF). A set of experiments have been devised to measure the

quality of results the WQF produces along two specific dimensions: *relevance* of output, and *scalability* under load.

[00114] In terms of *relevance* of output, the present inventors are interested in the following:

(i) The *search terms* that the WQF produces should be *relevant* in the context of the request input text. It is expected that relevant generated search terms should have high rates of *precision* and *recall* when compared to a standard known to be of high quality.

(ii) The *search results* that are returned when a WQF-produced search query 122 is submitted to a general-purpose search engine (e.g. Google™, Yahoo!™) should be *relevant* in the context of the request input text. Expected search results based on a WQF-produced search query should be judged to be significantly more relevant to typical real-world users than manually crafted queries.

[00115] In terms of *scalability* under load, it is desired that:

(i) The average *response time* for queries submitted to the WQF should be within the window of *responsiveness expectations* for typical internet users. Surveying indicates online users' current expectation is that content should be served in two seconds or less.

[00116] The architecture 100 described in the *Architecture and Method Overview* section was implemented with all modules (PE, PS, PR, QF, Crawler, and idf-Updater) in Java 1.7. These modules, along with a MySql 5.1 database for the *n*-gram data store 140, were installed on a 64-bit Windows Server 2008 R2 Enterprise Edition Server with 24 GB RAM and a quad-core 2.26 GHz CPU implementing the computer 1001.

[00117] Results of experiments on (a) the relevance of generated search query terms, (b) the relevance of search results based on generated search queries, and (c) the scalability of the WQF implementation under load can now be presented.

Relevance of Generated Search Query Terms

[00118] In the experiments, a focus was made on the relevance of generated search query terms, since high-quality search terms are the starting point for retrieving relevant results. Specifically, the inventors compared the relevance of the search terms generated by the WQF with those selected by a panel experts. The panel of experts consisted of a group of 10 professional lexicographers, each of whom had a graduate degree in a language-related field and more than 10 years of language-related work experience.

[00119] A set of 100 recent news articles of varying lengths were then selected from the CNN, WebMD and New York Times web sites. Each member of the expert panel was asked to manually identify a set of relevant phrases which could be a query candidate for each document based on document content, and to rank the set of phrases in order of relevance. From the ranked lists, a single top list with a set of query candidates was constructed on a per-document basis by ordering all the phrases by popularity across the expert panel, identified as Man (for manual). The same 100 documents were then submitted to the WQF implementation, where the expert panel did not have access to the results from the WQF.

[00120] The present inventors then computed the precision and recall for the WQF case compared to the Man result, where precision and recall are defined as follows: $Precision\ (WQF\ ) = \frac{r(WQF \cap Man)}{\mathrm{r(WQF)}}$, while $Recall(WQF\ ) = \frac{r(WQF \cap Man)}{r(Man)}$, where $r(WQF \cap Man)$ is the number of identified phrases common to the WQF and Man results,
$r(WQF)$ is total number of phrases extracted for each document using WQF, and $r(Man)$ is total number of top phrases extracted by the Man process.

[00121] The experiment first studied how precision and recall for the WQF approach vary as the $Affinity$ tuner δ is varied. As described in the *Scoring the Phrases* section, the purpose of δ is to bring $Affinity$ into the same range as $pfidf$ values. Analysis of the $pfidf$ and $Affinity$ values, revealed that $pfidf$ values were generally in a range from single-digit values to high double-digit values, while the maximum $Affinity$ values were less than 1. In order to bring $Affinity$ values into the range of $pfidf$ values, a δ value of at least 200 was

required. In this experiment, the value of δ was varied from 200 to 500, with the resulting precision and recall results being shown in Figures 2 and 3, respectively. Based on these results, WQF is seen to perform best when δ = 300, being a maximum for both precision and recall. Thus, this value of δ was used for the remainder of the experiments.

[00122] As seen in Figure 2, precision measures how many of the results generated were relevant (i.e., matched the *Man* results). Here, the WQF approach achieved a precision of 0.62 for δ = 300. Recall measures the completeness of returned results. Here, as seen in Figure 3, WQF attained a recall of 0.67 for δ = 300. To summarize, WQF was able to provide results that measure at or near two-thirds for both precision and recall.

[00123] Values of precision and recall were then computed for the WQF case for varying document lengths (which varied from less than 100 words up to 900 words). Figures 4 and 5 show how these metrics vary as document length varies for both approaches.

[00124] As seen in Figure 4, Precision increases with increasing document length, from roughly half when document length is between 0 and 300 words, to 0.83 when document length is between 701 and 900 words. Intuitively, this makes sense – with more context, the WQF can identify more of the correct results.

[00125] As seen in Figure 5, the recall results show a more interesting pattern. Recall initially increases as document length increases from 0.6 at 0-100 words to 0.73 for both 101-300 and 301-500 words. Recall then drops off for documents with length greater than 500 words to levels below the 0-100 word recall result. These results show that, unlike precision, additional context does not necessarily improve the completeness of WQF results. Rather, additional information is useful up to a point, after which it becomes more difficult to identify the most relevant results. This makes sense when it is understood how the WQF approach works. Longer documents are expected to contain more detail than shorter documents, with more rarer (higher idf) phrases, and more rare combinations of words (with higher $Affinity$). When a larger number of these phrases are present in a document, the WQF has a more difficult time discriminating among them. Clearly, the size of input text matters. In these

experiments, the inventors found that the input sizes that result in the best recall are between roughly 100 and 500 words.

[00126] Another factor in the recall results of Figure 5 is the threshold setting for the number of phrases the WQF approach will return. This threshold is selected to prevent the overall algorithm from generating very large numbers of phrases, which would tend to be problematic for the purposes of building search queries. In these experiments, the WQF approach was limited to 25 returned phrases, but no limit was placed on the number of phrases the expert panel of lexicographers could select as relevant. For larger document sizes in these experiments, the number of phrases returned by the lexicographers was larger than the threshold configured for WQF. Since the denominator of the recall expression is the number of phrases generated by the *Man* method, this had a negative impact on the recall results.

Relevance of Search Results Based on Generated Search Queries

[00127] In the experiment, the relevance of the actual search results obtained when queries produced by the WQF are submitted to a general-purpose search engine were considered. A laboratory experiment was conducted to compare the relevance of search results and the time consumed to find the desired results using either (i) the WQF plus (+) Google™ search, or a manual approach, where participants formulated their own search queries and submitted them to a Google™ search.

[00128] Eight popular news documents were selected from different fields (science, lifestyle, business, sports, news, medical, social networking, and travel) for this experiment. The popularity of the topics ensured the participants would be able to understand the text. Document lengths varied between 200 and 550 words.

[00129] Eighty participants signed up for the experiment and thirty-two participants took part. Among the subjects 51.5% were female and 48.5% were male graduate students. The average age of the participants was 25.6 years. This shows the random sampling attribute in the participant selection and would ensure the results can be generalizable. All participants have a university degree

in computing (or a related field) with at least one to two years of work experience in the field of computing or information systems. By the virtue of their backgrounds, all participants are internet savvy and familiar with the use of search engines to find documents related to their interest. In general, the participants felt very comfortable with internet usage (mean: 6.5/7.0). Thus, the participants were internet savvy and conversant with searching documents in the internet. Hence, the background of the participants is relevant to the experiment. Since these participants are advanced users of internet search, they can be considered experts in the domain. Therefore, if these participants report that the WQF method provides better results than they can find on their own, it is reasonable to expect that the WQF method would provide even greater benefit to naïve search users.

[00130] Each participant was asked to read each document. For half of the documents, the participant was instructed to use WQF to generate the search query, and then submit the generated query to Google. For the other half of the documents, the participant was instructed to manually craft a search query to find related documents, and then submit the query to Google. Participants were asked to iteratively search until satisfied with the results, and then to rate the relevance of the first 10 documents returned. For each document-task, the participant was asked to record the following information: (1) the final search query that returned the results that satisfied the participant; (2) the time required to find the related satisfied results, including the time to read the document (i.e., less than 30 seconds, less than 1 minute, less than 5 minutes, other); and (3) the relevance of the first ten documents returned, rated on a scale from 1 to 5 (where 5 denotes a highly relevant document, and 1 denotes a less relevant document).

[00131] To avoid bias based on prior results or due to the order of tool usage, the 8 documents were randomly divided into two sets S_1 and S_2, where each set contained four documents. The participants were randomly divided into two groups, G_1 and G_2. The participants in G_1 were all asked to perform the same task; the participants in G_2 were asked to perform a similar task. Each group was then subdivided into two subgroups G_{1a} and G_{1b}, and G_{2a} and G_{2b}, where only the ordering of tasks differed across subgroups (to demonstrate that task sequencing does not confound the experimental results). The participants in G_{1a} were asked to search the documents in S_1 using WQF and Google™, and then the documents

in S_2 by manually crafting a query and submitting it to Google™. The participants in G_{1b} were asked to search the documents in S_2 using the manual method, and then the documents in S_1 using WQF. The participants in G_{2a} were asked to search the documents in S_1 using the manual method, and then the documents in S_2 using WQF. The participants in G_{2b} were asked to search the documents in S_2 using WQF, and then the documents in S_1 using the manual method.

[00132] As is evident from the above, the experiment was designed the test to make sure that each combination of the treatment (document set, query generation method) and sequence was verified, to dispel any concerns of idiosyncratic effects biasing the results of the experiment. Further, when the 32 individuals were put in different groups using random allocation and the experiment conducted again, similar scores were observed, thereby confirming the experiment results. Previous studies related to semantic structuring in requirement analysis and end user query development were conducted with similar numbers of participants to confirm their findings. Thus, the results of the experiment were considered reliable.

[00133] Each participant rated the relevance for each of the first ten search results. Then, the average relevance for each document was computed across all participants. Figure 6 shows the average relevance across the first ten results and all participants for each of the eight documents for both the WQF and manual cases. Figure 7 plots the average WQF search time for each document across all participants.

[00134] When averaged across all documents, the WQF case shows a full point of greater relevance (on a 5-point scale) when compared to the manual case (4.42 to 3.37), and more than 3 minutes of savings in search times. These are significant results, showing that WQF-formulated queries can retrieve high-relevance documents within a short period of time.

[00135] More detailed data of this experiment is presented in Table 6, which shows the average, maximum and minimum relevance per document across all participants, to compare the manual and WQF cases. Table 6 shows that WQF outperformed the manual method consistently in all 3 measures - maximum,

minimum and average. Table 7 shows similar results for the search time for each document.

Scalability

[00136] In this section, a set of experiments are described to characterize the scalability of the WQF approach. Scalability is considered along two dimensions: (a) as request arrival rate is varied; and (b) as input document size is varied. In both cases, the average response time of the WQF system is relevant.

[00137] Arrival rate experiments were conducted. The arrival rate is the rate of requests received by the server executing the WQF, typically measured as the number of simultaneous requests per second. Here, the arrival rate was varied, while holding document length steady at around 700 words. To generate the workload, an experimental set of web pages with 650-750 words per document, drawn from the test bed set of CNN most popular stories described above in the *Solution Approach* section.

[00138] A multi-threaded client program was used to generate the query workload. Each thread, at a regular interval (where shorter intervals generate higher workloads), randomly selected a web page and submitted the selected web page for WQF processing. The rate of request arrival at the WQF was varied by controlling the sleep time between two submissions in a thread. The client program measured the elapsed time between the submission of an input document to the WQF system and the time at which the WQF system returned with the top ten search terms for that document. The response time is averaged across all requests for each tested workload.

[00139] The results are reported in Figure 8, which is a plot the average response times (measured in milliseconds) against request arrival rate on the WQF (measured in requests/second). As can be seen from Figure 8, WQF response time shows classic exponential growth behavior with increasing load. For the highest workload tested, 10 requests per second (which works out to a daily load of more than 690,000 requests), the average response time is 1.79 seconds.

[00140] The next investigation was how the WQF approach scaled with document length. An experiment was conducted measuring the average WQF

response time as the document length was varied from less than 100 words up to 1,000 words (using the 300-document news set described in the *Solution Approach* section), holding the request arrival rate constant at 8 requests per second. The results are plotted in Figure 9. The result shows that the response time for the WQF increases non-linearly with the document length. For a thousand word document, the average response time for WQF approach is 1.78 seconds at the highest document-length range tested, which is within the 2-second user responsiveness expectation.

Conclusion

[00141] The described WQF approach is a method which can be used in an advanced step of an information retrieval process, when the user has already identified a set of relevant text(s) which can be used as a set of input text. Experimental results demonstrate that the WQF system produces relevant search terms with roughly two-thirds precision and recall compared to search terms selected by experts, and that typical users find significantly more relevant results (31% more relevant) more quickly (64 % faster) using the WQF system than self-formulated search queries. Further, the WQF implementation can scale to request loads of up to ten requests per second within current online responsiveness expectations (sub-2-second response times, even at the highest loads tested).

Industrial Applicability

[00142] The arrangements described are applicable to the computer and data processing industries and particularly for the generation of search queries to afford enhanced search results, for example for the searching of the Internet, World Wide Web and, as appropriate, private computer networks or resources. The arrangements are also relevant to data mining of databases.

[00143] The foregoing describes only some embodiments of the present invention, and modifications and/or changes can be made thereto without departing from the scope and spirit of the invention, the embodiments being illustrative and not restrictive. Further, the computer-implemented algorithms described above using pseudo-code may be equivalently represented by

flowcharts to specifically illustrate the computerised processing of the query formulation methods described herein.

[00144] On the following pages, and before the Claims of this patent specification, Tables 1 to 7 referred to above are provided.

Table 1: Example Comparison of $\sum_{i=1}^{n} tfidf(wi)$ to $pfidf$

	Phrase	*tfidf* (w_1)	*tfidf* (w_2)	*tfidf* (w_3)	*tfidf* (w_4)	Estimated *tfidf* $\sum_{i=1}^{n} tfidf(wi)$	*pfidf*
1	Russia 's Intelligence Services	49.92	9.03	69.14	15.64	143.73	23.31
2	Senior U.S. Intelligence Official	5.33	58.23	69.14	10.93	143.63	18.66
3	Intelligence Report	69.14	54.91	0	0	124.05	13.78
4	Government Intelligence Services	34.24	69.14	15.64	0	119.02	20.63
5	U.S. Private Sector Firms	58.23	37.06	13.86	7.81	116.96	23.54
6	Economic And Technology Information	54.99	6.36	17.97	30.25	109.57	22.96
7	Economic Information And Technology	54.99	30.25	6.36	17.97	109.57	21.43
8	Foreign Economic Espionage	17.68	54.99	33.39	0	106.06	27.13
9	Battle Against Foreign Intelligence	5.7	12.93	17.68	69.14	105.45	26.13
10	China And Russia	49.14	6.36	49.92	0	105.42	50.46
11	National Intelligence Officer	24.41	69.14	6.47	0	100.02	18.58
12	National Intelligence Council	24.41	69.14	6.33	0	99.88	15.72
13	Economic Information And Technolo-gies	54.99	30.25	6.36	8.26	99.86	23.81
14	U.S. Government Produce	58.23	34.24	6.11	0	98.58	25.54
15	Foreign Intelligence	17.68	69.14	7.67	0	94.49	21.58

	Threats						
16	National Intelligence	24.41	69.14	0	0	93.55	34.51
17	U.S. Government	58.23	34.24	0	0	92.47	9.33
18	Theft Of U.S.	25.93	8.21	58.23	0	92.37	24.88

Table 2: Notation

Notation	Meaning
W	word
P	phrase
C	collocate
$\|D\|$ $tf(w_i)$	total document count in the corpus term count of word w_i in document D
$idf(w_i)$	inverse document frequency of word w_i in corpus
$tfidf(w_i)$	product of term frequency in to inverse document frequency of word w_i
$df(P)$	number of documents which contains phrase P in corpus
$df(w_i)$	number of documents which contains word w_i in corpus
n	number of words in phrase P
$P1$	ratio between frequency of pre word with phrase P to the frequency of phrase P
$P2$	ratio between frequency of post word with phrase P to the frequency of phrase P
$f_D(P)$	frequency of the phrase P in document D
$f_D(C)$	frequency of the collocate C in document D
$f(P)$	frequency of the phrase P in corpus
$idf(P)$	inverse document frequency of phrase P in corpus
$f_{DP}(w_i)$	summation of frequency in document D of all words in phrase P; $w_i \square$ P and i = 1 to $\|P\|$
$f_P(w_i)$	summation of frequency in corpus of all words in phrase P; $w_i \square$ P and $i = 1$ to $\|P\|$
$f_{posP}(w_i)$	summation of frequency in corpus of all words in phrase P whose POS are not started with NP, VB, JJ or RB
$Affinity(P)$	measure of frequency of an n-gram in relation to other n-grams with overlapping constituent words
$S(P)$	score of the phrase P
δ	*Affinity* leveling tuner
β	weight parameter for phrases with special characteristics
ρ	selectivity parameter for keeping top phrases
γ	Frequency threshold for retaining to longer phrases when removing redundant terms

Table 3: *2-gram* Corpus Schema

gram	value	df	*idf*(P)	wordid1	wordid2

Table 4: Meaning

NN	Noun
JJ	Adjective
RB	Adverb
DT	Pre-determiner
CC	Coordinating conjunction
PRP$	Possessive pronoun
POS	Possessive ending

Table 5: Pattern

Pattern	Instance
NN	match
PRP$ NN	his daughter
NN NN	world cup
JJ NN	beautiful mind
NN NN NN	National University Hospital
NN CC NN	School of Computing
NN DT NN	Alexander the conqueror
NN DT JJ	Alexander the great
NN POS NN	today's match
NN NN NN NN	United States President Obama
NN NN CC NN	United States of America
JJ CC JJ NN	nice and smooth skin
JJ JJ NN NN	great big sea tour
NN NN NN NN NN	FIFA World Cup South Africa

Table 6: Average-Maximum-Minimum Relevance using manual approach and Wordster system

	Google Search			Wordster (WQF) Search		
DocID	Average relevance	Max Relevance	Min Relevance	Average relevance	Max Relevance	Min Relevance
1	3.4	4.2	2.7	4.4	5	3.9
2	3.87	4.9	2.6	4.21	5	3.2
3	3.15	4.5	1.9	4.6	5	4
4	3.49	4.6	2.4	4.31	5	3.6
5	3.49	4	2.9	4.59	4.9	4.2
6	3.47	4.6	2.7	4.49	4.9	4.1
7	3.47	4.6	3	4.38	4.9	3.8
8	2.65	4.5	1.3	4.38	4.6	4.1

Table 7: Average-Maximum-Minimum Time (minutes) using manual approach and Wordster system

	Google Search			Wordster (WQF) Search		
DocID	Average Time	Max Time	Min Time	Average Time	Max Time	Min Time
1	5.5	10	2	2.58	5	0.5
2	4.3	5	1	1.92	5	0.5
3	6.1	10	2	2.25	5	0.5
4	6	10	2	2.92	5	0.5
5	6.4	10	1	1.75	6	0.5
6	5.8	10	2	1.83	5	1
7	5.2	10	1	1.17	5	0.5
8	5.9	10	3	1.92	6	0.5

CLAIMS:

1. A search query construction system comprising:

an n-gram database of inverse document frequency data of phrases in network locations;

a phrase extractor configured to extract candidate phrases from an input document;

a phrase scorer configured to determine a phrase frequency of a candidate phrase in the input document and, with the inverse document frequency data of phrases from the n-gram database and a metric of quality of the candidate phrases, form a score for each of the candidate phrases;

a phrase ranker to rank the candidate phrases according to the scores; and

a query formulator configured to transform a predetermined number of top ranked candidate phrases into a search query associated with the input document.

2. A system according to claim 1, wherein the score comprises a phrase frequency inverse document frequency term determined from a product of the inverse document frequency of a phrase and a term derived from phrase frequency of the document.

3. A system according to claim 1, wherein the metric of quality of the candidate phrases is based upon a frequency of occurrence of words in the phrase with other words in the phrase, the metric of quality being used to modify the score for the corresponding candidate phrase.

4. A system according to claim 1, wherein the score is determined according to the expression:

$$S(P) = log_2\,(1 + f_D(P)) \times idf(P) + \delta \times Affinity_c(P)\ ,$$

where

$f_D(P)$ is the frequency of phrase P in the input document D ,

$log_2\,(1 + f_D(P)) \times idf(P)$, represents the $pfidf$ score of phrase P,

$idf(P) = \left(log_2 \frac{|D|}{df(P)} \right)$, where $|D|$ is the total number of documents in a corpus of the database and $df(P)$ is the document frequency of phrase P ,

$Affinity_c(P) = \frac{f(P)}{min_{\forall w_i \in P(f(w_i))}} \times (1 - \max(P1, P2))$, where $f(P)$ is the frequency of phrase P in the corpus and $min(f(w_i))$ is the minimum frequency across the words in phrase P, and

δ is a leveling tuner used to bring $Affinity$ values into the same range as $pfidf$.

5. A system according to claim 1, wherein the phrase ranker ranks the candidate phrases according to the scores and selects a plurality of top ranked ones of the candidate phrases according to the number of candidate phrases.

6. A system according to claim 5 wherein the number of selected candidate phrases is generally proportional to the total number of candidate phrases.

7. A system according to claim 1, wherein the query formulator removed redundancies from top ranked candidate phrases and collocates and concatenates the predetermined number of top ranked candidate phrases into the search query.

8. A system according to claim 7 wherein the predetermined number is the top one-third of the ranked candidate phrases.

9. A system according to claim 1, wherein the phrases comprise noun phrases and the inverse document frequency data in the database comprises inverse document frequency values of noun phrases in the network locations.

10. A system according to claim 1, further comprising:

a crawler associated with the database and configured to extract at least phrase frequency data of phrases at the network locations; and

an updater associated with the database and configured to interpret the phrase frequency data in the database and to update inverse document frequency values for the phrases in the database.

11. A system according to claim 1, wherein the phrase extractor extracts candidate phrases each having at least two and no more than four words.

12. An automated query constructor comprising:

a phrase extractor configured to extract candidate phrases from an input document;

a phrase scorer configured to determine a phrase frequency of a candidate phrase in the input document and to use at least inverse document frequency data of phrases obtained from an n-gram database and the determined phrase frequency to form a score for each of the candidate phrases;

a phrase ranker to rank the candidate phrases according to the scores; and

a query formulator configured to transform a predetermined number of top ranked candidate phrases into a search query associated with the input document.

13. An automated query constructor according to claim 12 wherein the score comprises a phrase frequency inverse document frequency term determined from a product of the inverse document frequency of a phrase, a term derived from phrase frequency of the document, and a metric of quality of the candidate phrases, determined by the phrase scorer, based upon a frequency of occurrence of words in the phrase with other words in the phrase, the metric of quality being used to modify the score for the corresponding candidate phrase.

14. An automated query constructor according to claim 12, wherein the score is formed according to a first term expression:

$$S(P) = log_2\,(1 + f_D(P)) \times idf(P),$$

where $idf(P) = log_2 \frac{|D|}{df(P)}$

where $|D|$ is the total number of documents in a corpus of the database and $df(P)$ is the number of documents in the corpus that contain a phrase P, and $f_D(P)$ is the frequency of the phrase P in the input document D.

15. An automated query constructor according to claim 14 wherein the score is modified by addition of a second term expression $\delta \times Affinity_c(P)$ to the first term expression, where

$$Affinity_c(P) = \frac{f(P)}{min_{\forall w_i \in P(f(w_i))}} \times (1 - \max(P1, P2)),$$

where $f(P)$ is the frequency of phrase P in the corpus and $min(f(w_i))$ is the minimum frequency across the words in phrase P, and the term

$(1 - max(P\,1, P\,2))$ determines the relevance of the phrase with respect to its neighborhood, and δ is leveling tuner is used to bring $Affinity$ values into the same range as the first term expression.

16. A non-transitory computer readable storage medium having a program recorded thereon, the program being executable by a computer to form a search query from an input document, said program comprising:

code for extracting candidate phrases from the input document;

code for determining a phrase frequency of an extracted candidate phrase in the input document and to use inverse document frequency data of phrases ($idf(P)$) obtained from an n-gram database and the determined phrase frequency to form a score for each of the candidate phrases;

code for ranking the candidate phrases according to the scores; and

code for formulating a search query from a predetermined number of top ranked candidate phrases.

17. A non-transitory computer readable storage medium according to claim 16 wherein the score comprises a phrase frequency inverse document frequency term determined from a product of the inverse document frequency of a phrase and a term derived from phrase frequency of the document.

18. A non-transitory computer readable storage medium according to claim 17, wherein the score is determined according to a first term expression:

$$S(P) = log_2\,(1 + f_D(P)) \times idf(P),$$

where $$idf(P) = log_2 \frac{|D|}{df(P)}$$

where $|D|$ is the total number of documents in a corpus of the database and $df(P)$ is the number of documents in the corpus that contain a phrase P, and $f_D(P)$ is the frequency of the phrase P in the input document D.

19. A non-transitory computer readable storage medium according to claim 18 wherein the score is modified by addition of a second term expression $\delta \times Affinity_c(P)$ to the first term expression, where

$$Affinity_c(P) = \frac{f(P)}{min_{\forall w_i \in P(f(w_i))}} \times (1 - \max(P1, P2)),$$

where $f(P)$ is the frequency of phrase P in the corpus and $min(f(w_i))$ is the minimum frequency across the words in phrase P, and the term $(1 - max(P\,1, P\,2))$ determines the relevance of the phrase with respect to its neighborhood, and δ is leveling tuner is used to bring $Affinity$ values into the same range as the first term expression.

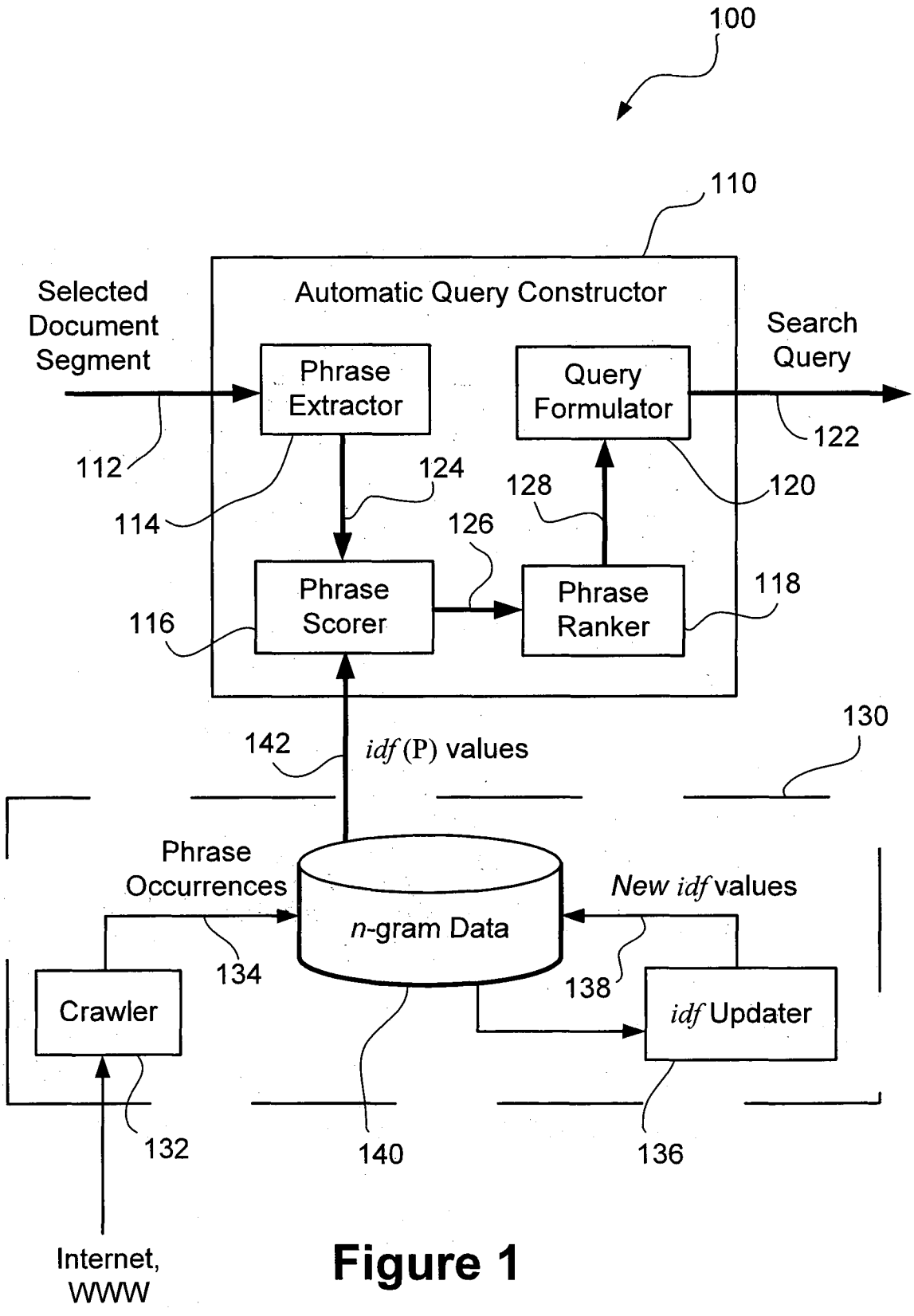

100
110
Automatic Query Constructor
Selected Document Segment
Search Query
Phrase Extractor
Query Formulator
Phrase Scorer
Phrase Ranker
112
114
124
126
128
120
122
116
118
130
142
idf (P) values
Phrase Occurrences
n-gram Data
New idf values
Crawler
idf Updater
134
138
132
140
136
Internet, WWW

Figure 1

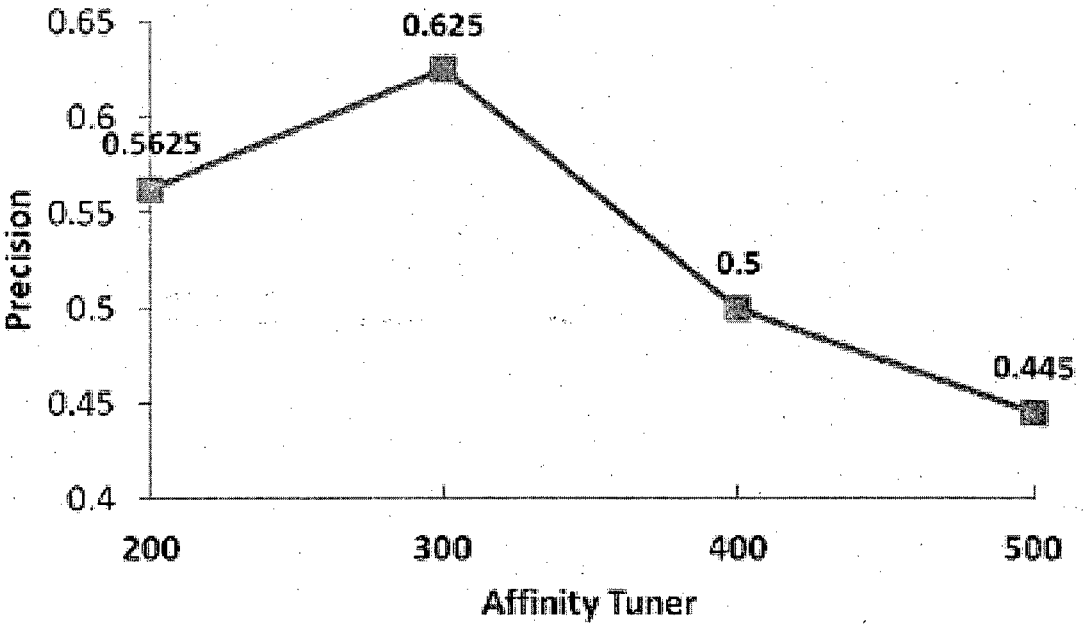


Figure 2: Precision vs Affinity Tuner of *WQF*

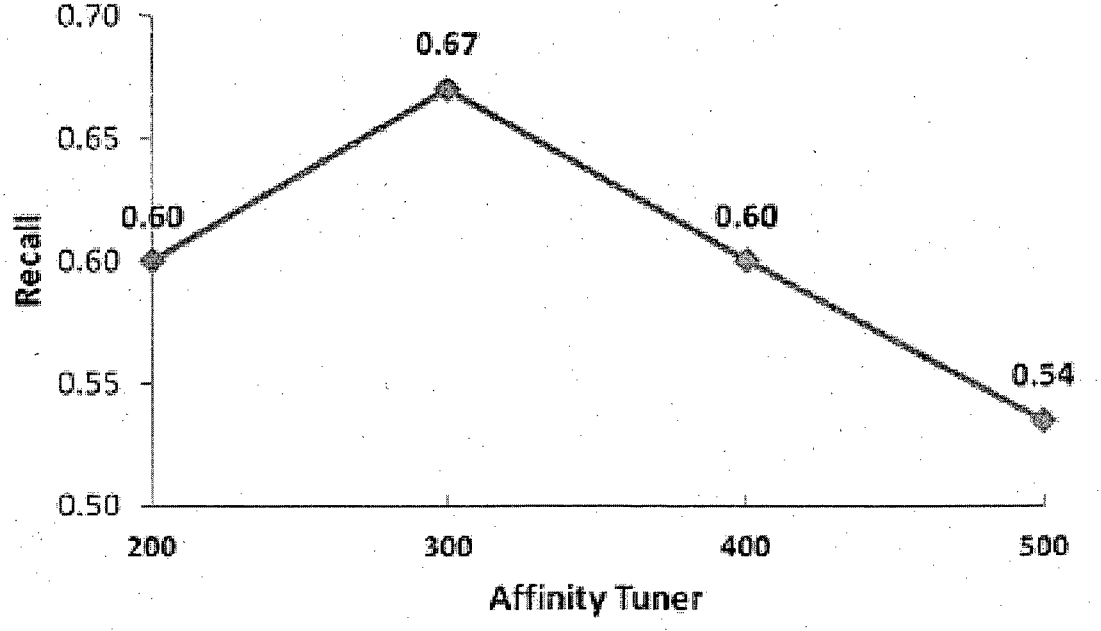


Figure 3: Recall vs Affinity Tuner for *WQF*

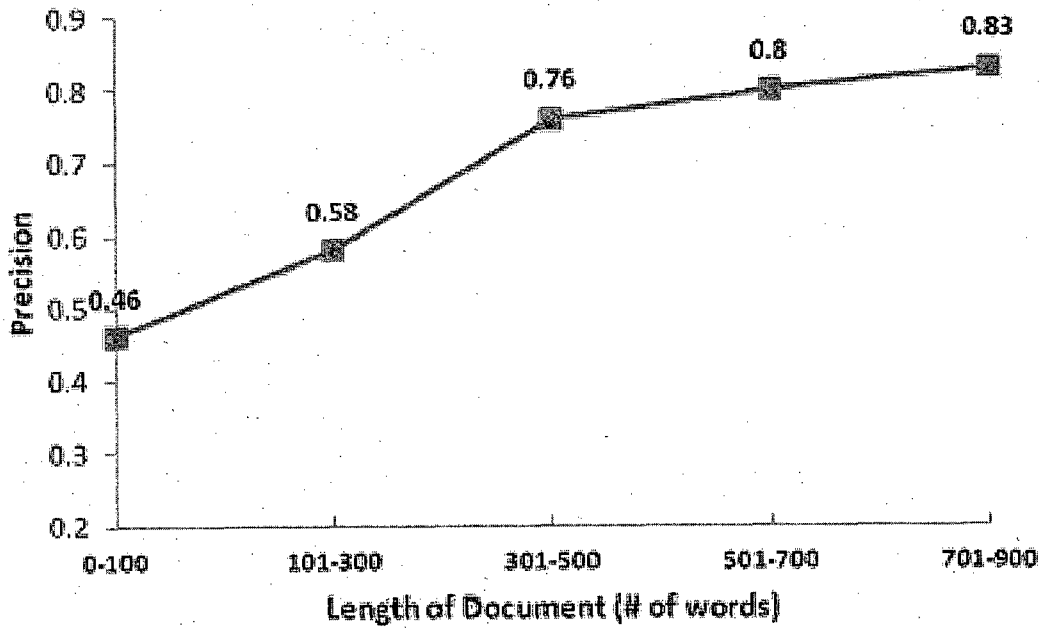


Figure 4: Precision vs Document Length for *WQF*

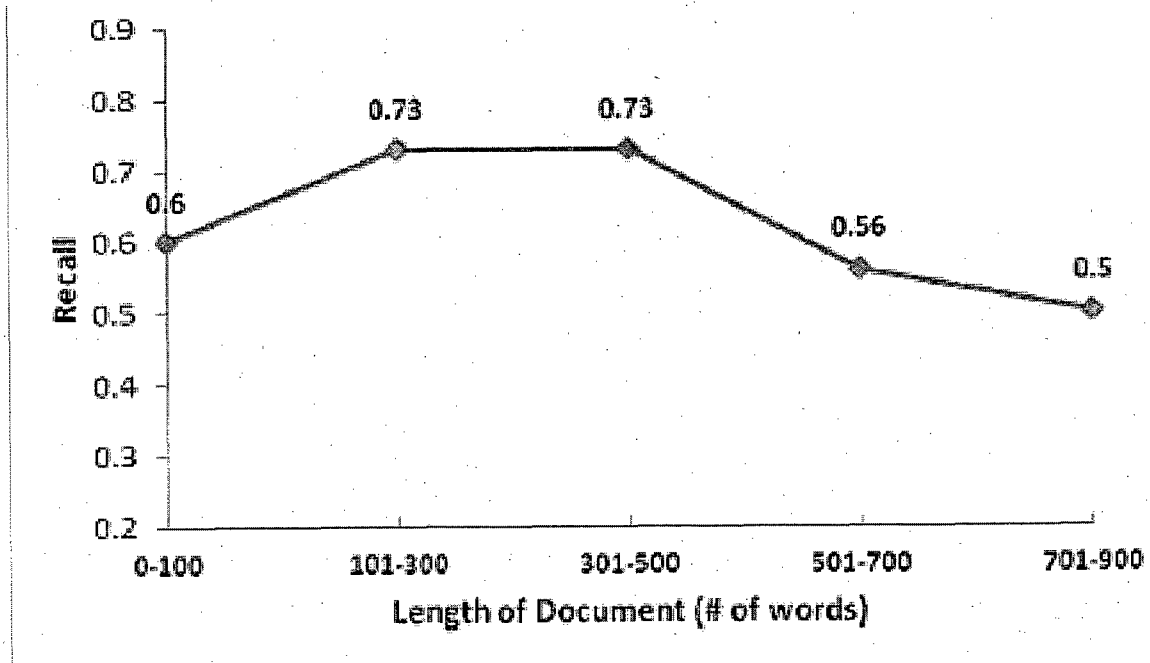


Figure 5: Recall vs Document Length for *WQF*

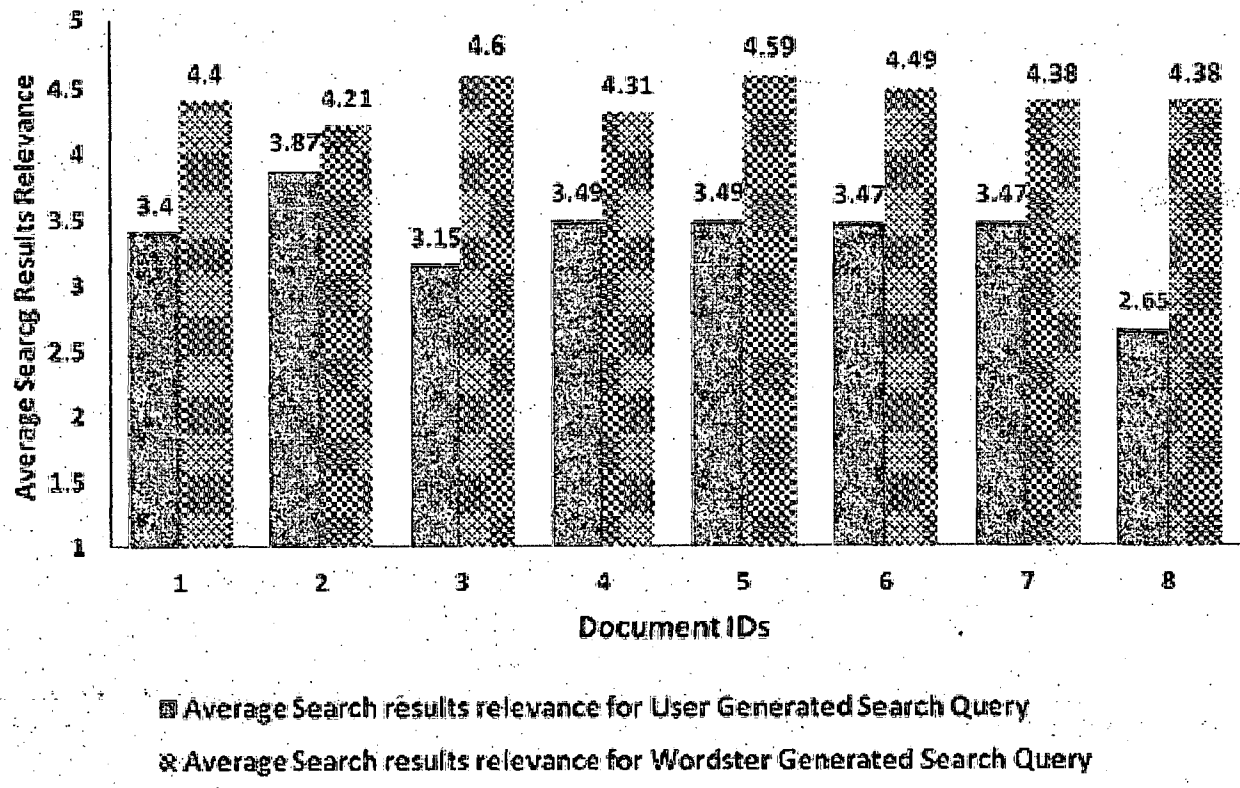


Figure 6: Search Results Relevance

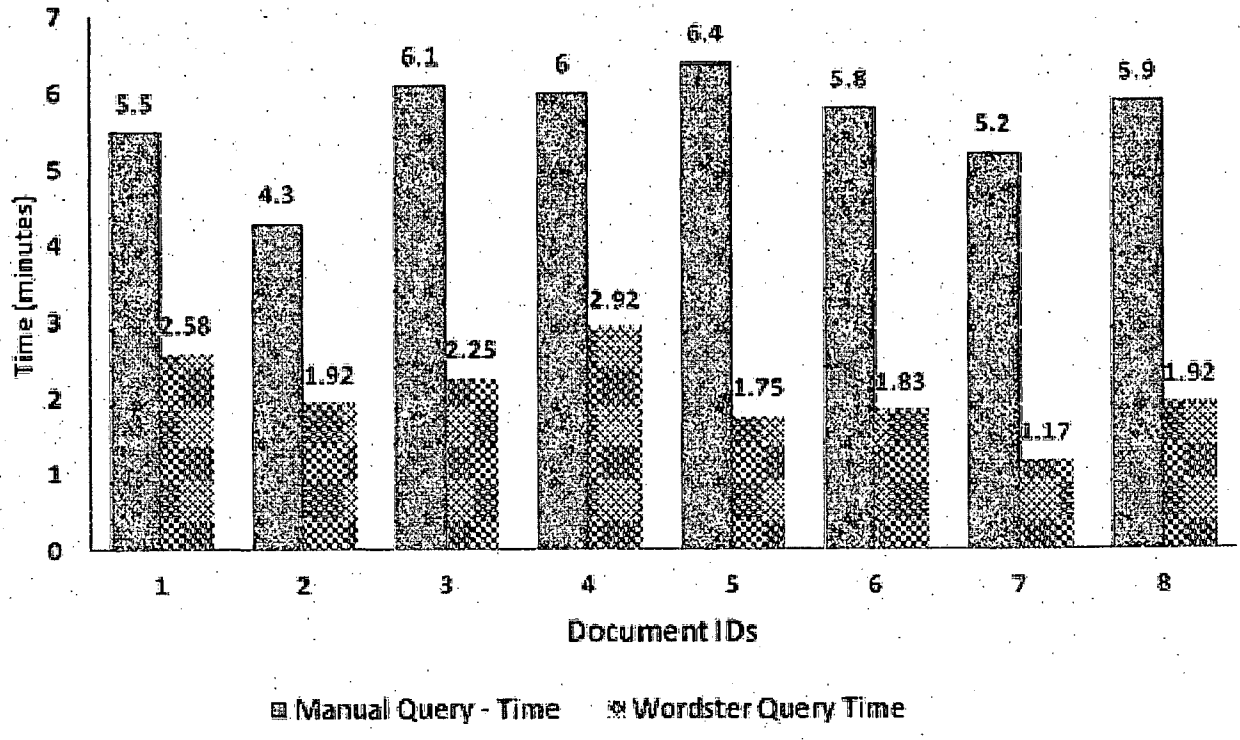


Figure 7: Time to Search

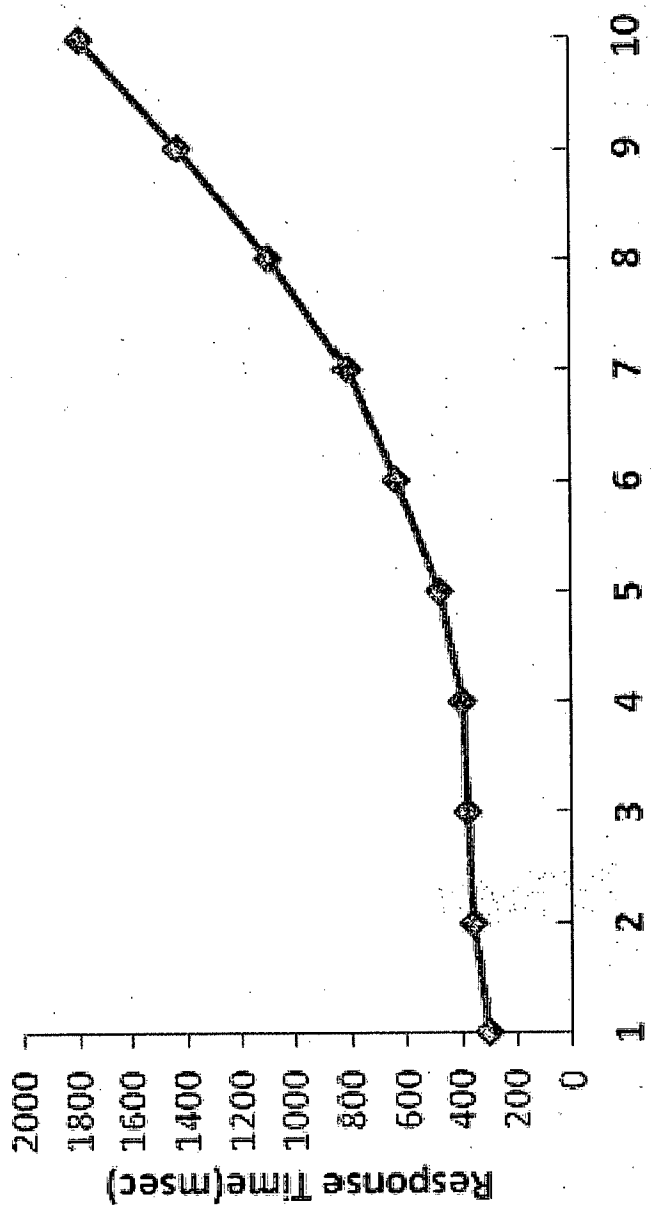


Figure 8: Response Time(msec) vs. Input Load (Request/Seconds)

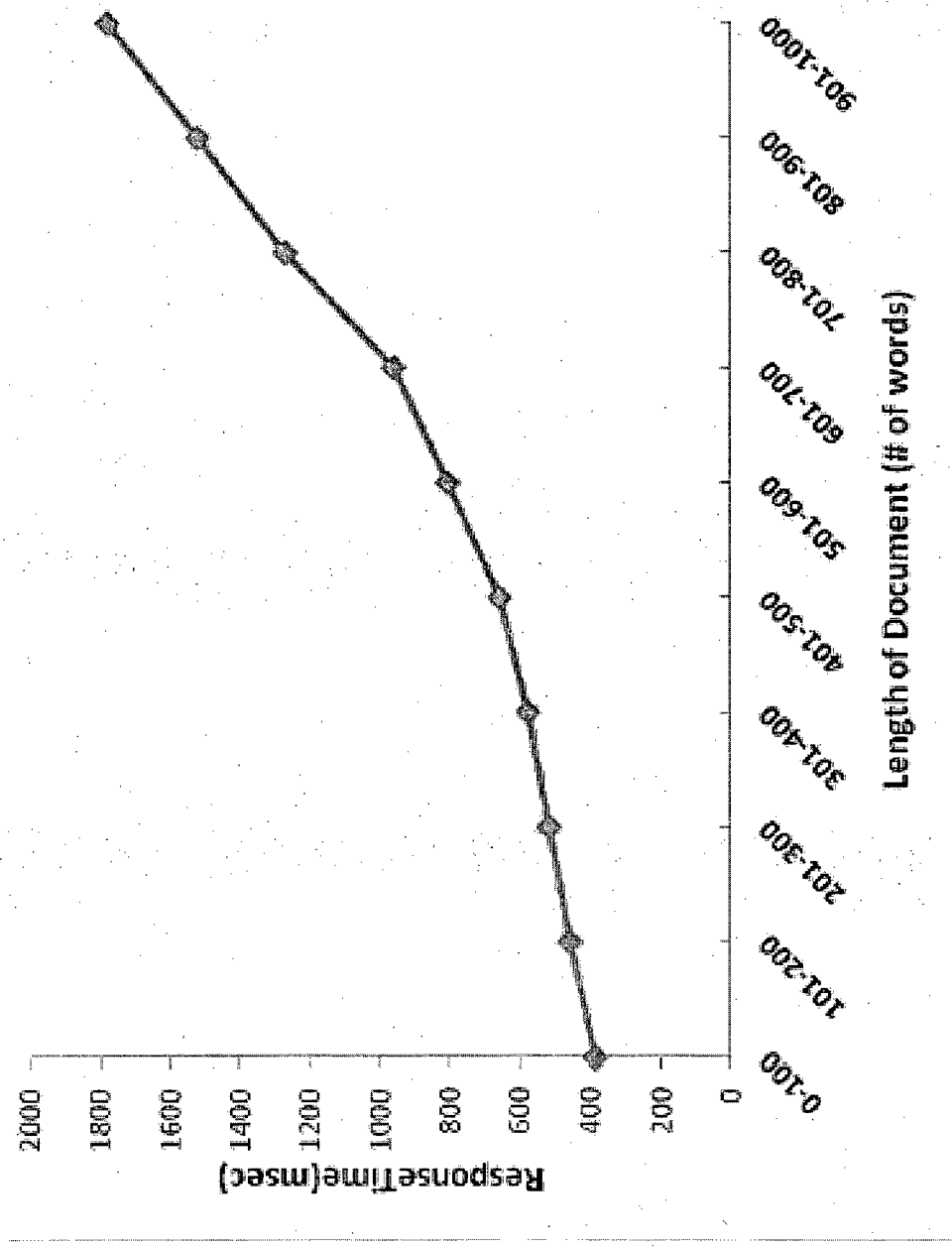


Figure 9: Response Time(msec) vs. Length of Document (Number of words)

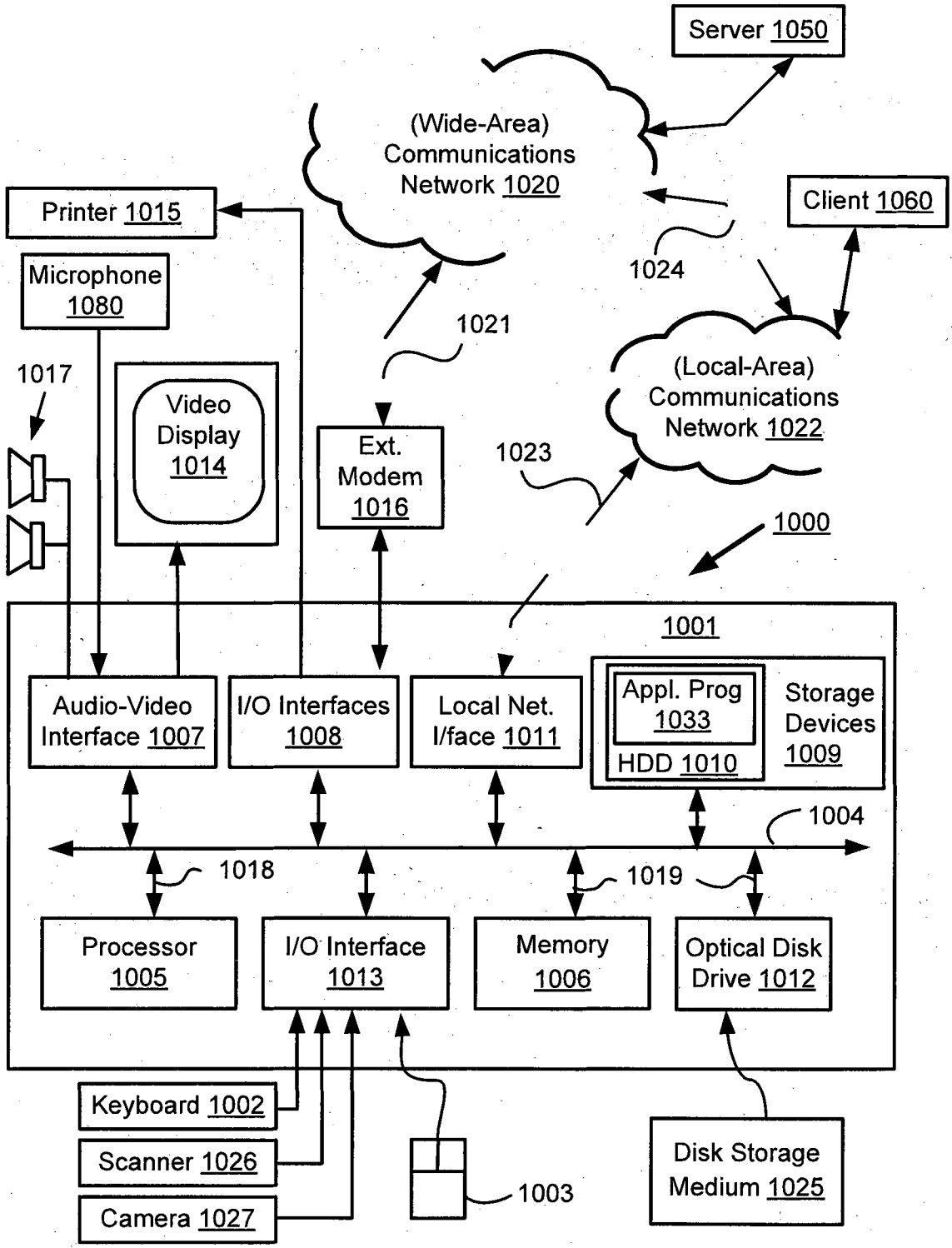

Server 1050
(Wide-Area) Communications Network 1020
Client 1060
Printer 1015
Microphone 1080
1024
1021
1017
Video Display 1014
Ext. Modem 1016
(Local-Area) Communications Network 1022
1023
1000
1001
Audio-Video Interface 1007
I/O Interfaces 1008
Local Net. I/face 1011
Appl. Prog 1033
HDD 1010
Storage Devices 1009
1004
1018
1019
Processor 1005
I/O Interface 1013
Memory 1006
Optical Disk Drive 1012
Keyboard 1002
Scanner 1026
Camera 1027
1003
Disk Storage Medium 1025

Figure 10

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SG2013/000411

A. CLASSIFICATION OF SUBJECT MATTER

G06F 17/30 (2006.01)

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WPI : Keywords (usage, frequency, phrase, inverse, document, idf) & like terms
TXTUS5, TXTUS4, TXTUS3, TXTUS2, TXTUS1, TXTUS0, TXTEP1, TXTGB1, TXTWO1, TXTAU1, TXTCA1, TXTSG1 : Keywords (usage, frequency, phrase, inverse, document, idf, score, rank, weight, n-gram) & like terms
EspaceNet : Keywords (datta, anindya, dutta, kaushik, kajanan, sangaralingam) & like terms

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	Documents are listed in the continuation of Box C	

[X] Further documents are listed in the continuation of Box C

[X] See patent family annex

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
26 November 2013	26 November 2013

Name and mailing address of the ISA/AU

AUSTRALIAN PATENT OFFICE
PO BOX 200, WODEN ACT 2606, AUSTRALIA
Email address: pct@ipaustralia.gov.au
Facsimile No.: +61 2 6283 7999

Authorised officer

Ross Stopford
AUSTRALIAN PATENT OFFICE
(ISO 9001 Quality Certified Service)
Telephone No. 0262832177

INTERNATIONAL SEARCH REPORT C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		International application No. **PCT/SG2013/000411**
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 7617176 B2 (ZENG et al) 10 November 2009 column 1 lines 39/46, column 17 lines 42/58, column 5 lines 46/63, column 7 lines 6/23, prior art document claim 1, column 8 lines 40/47, column 11 lines 23/30, column 17 lines 42/58, column 10 lines 23/25, column 8 lines 34/51, column 11 lines 38/47	1 - 19

INTERNATIONAL SEARCH REPORT Information on patent family members	International application No. **PCT/SG2013/000411**

This Annex lists known patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent Document/s Cited in Search Report		Patent Family Member/s	
Publication Number	**Publication Date**	**Publication Number**	**Publication Date**
US 7617176 B2	10 Nov 2009	US 7617176 B2	10 Nov 2009

End of Annex

Due to data integration issues this family listing may not include 10 digit Australian applications filed since May 2001.