Hybrid Ontology and Keyword Matching Indexing System

Géry Ducatel
British Telecommunications plc
Adastral Park
Orion Building pp1/12
+44(0)1473 605 472
gery.ducatel@bt.com

Zhan Cui
British Telecommunications plc
Adastral Park
Orion Building pp1/12
+44(0)1473 605 472
zhan.cui@bt.com

Ben Azvine
British Telecommunications plc
Adastral Park
Orion Building pp1/12
+44(0)1473 605 472
ben.azvine@bt.com

ABSTRACT
This paper describes a search tool that currently is being used by British Telecommunications engineers to access a technical repository. Our search tool is composed of two major components: an ontology assisted query understanding/rephrasing and a keyword based search that has an indexer enhanced with automatic text classification. Although the ontology was built for engineers it is a good basis to help represent semantics within the telecom industry. Our ambition is to apply our technology to an online self-service application.

Categories and Subject Descriptors
H.3.5 [**Information Storage and Retrieval**]: Online Information Services – *web based services.*

General Terms
Algorithms, Measurement, Performance, Reliability, Experimentation, Human Factors.

Keywords
Information Management, Semantic Analysis, Ontology, Information Retrieval, Services, Self-Service.

1. INTRODUCTION
This paper describes a hybrid system that combines an ontology and a keyword matching approach. It has been notoriously difficult to standardise ways of classifying information available on computer networks. The semantic web is a framework that intends to describe different strategies and technologies that can be used in order to start building broad and generally acceptable standards. This work has many research issues to be addressed; So far it is clear that content providers will be able to bring their contribution as well as benefit from a consensual classification scheme. Unifying globally obtained semantics into one theoretical model will result in a self-building categorization scheme. Ultimately, information management with the semantic web would not suffer restrictions inherent to keyword matching; information would be classified and searched on more abstract criteria such as concepts, validity, or quality. However, the semantic web is maturing [15, 17] but slowly and some concepts had to be tested behind closed doors first (i.e. on Intranets). In our case, we are using an ontology in order to understand the meaning of domain dependent queries. We then use the context clarified queries in a domain specific search engine.

Our technology had been applied to other problem spaces: accessing technical information for BT engineers, or for operators in contact centres [5]. An ontology (manually built) is used to capture domain knowledge; this is used to put coming queries into their rightful context and access information in the appropriate space. Self-service (or self-help) is another possible application that can follow the same format. Traditionally self-service application are managed by FAQ management software. These solutions are popular at the entry level of self-help because the majority of contact centre queries revolve around the same three to five problems [3]. However, it proves difficult to maintain the appropriate semantics, and answers are often a one size fits all that falls short of being a reliable source. The next generation of self-service applications sees a growing number of software products [1, 11], both from support vendors and marketing automation firms. They are aiming at providing fast, automated answers to common questions, using natural language technology, which many believe to be key to the future of customer support.

This paper describes an ontology driven query processing and its dialogue/conflict resolution process. The second section describes the search engine part of the system. The emphasis is put on the automatically classified information that allows the index to be in tune with the structure of an ontology. The conclusion discusses strengths and weaknesses of such a system as well as asking more general questions about introducing the semantic web on a larger scale.

2. Combining Ontology Driven and Keyword Matching

2.1 Ontology Driven Solution
An ontology is a description of the things which we are interested in (referred to as concepts) along with information about the relationships between the concepts and properties which the concepts have. A formal definition of an ontology can be found in [6]. It can be seen that a product set can be readily represented in an ontology with concepts for products and for the aspects of the products such as price, fault information etc. Each product can have properties specifying which aspects are appropriate for the concept. The concepts are classified into two types: topic (typically products and services, or special offers) and action (including such requests as tariff information, or fault inquiry). Instead of a simple taxonomy, the ontology allows to add constraints to taxonomy and to perform inferences with concepts.

The ontology provides a semantic index to information sources, processes and expertise of human experts. The basic mode of operation is to analyse natural language queries, chunk them into phrases and then map to concepts in the ontology. If it can refer to several concepts then the system can ask further questions to disambiguate the phrase. Finally the system should apply some action attached to the concept e.g. forward to a web-page or display compliance advice.

2.1.1 Phrase to Concept Mapping

Phrase to concept mapping (seen on Figure 1 in the query processing stack) is a tool able to chunk phrases into separate input (which may be sentences, whole emails etc.) into smaller chunks which can be matched to concepts. Key phrases are identified on three possible basis: they are occurring within the text corpus, the exist in a dictionary (our system uses WordNet [16]), or they have been manually entered into a local dictionary. For each concept in the ontology there is an associated list of key-phrases which are related to the concept. Each key-phrase has an associated support of between 0 and 1 which corresponds to the relevance of the phrase to the concept. For example for the concept :action:fault: the relevant key-phrases might be: broken = 0.9, not working = 0.9, loose = 0.3, squeaky = 0.1.

In order to obtain a weighting different methods were tried. No satisfactory solution was identified. Therefore, most of the weighting is done manually. Amongst the solutions tried the most promising one consisted of adapting a *tf.idf* type of weighting scheme to keyword co-occurrence were the *idf* part is replaced by the sum of all co-occurrences throughout the text corpus. This weight can be normalised to a number between 0 and 1. However, tests showed that only one word in three was associated with some relevant context. Therefore, no real automatic solution has been found using word frequency based techniques.

The fuzzy-concept mapping connects phrase chunks to concepts in the ontology. At any given time the concept-mapping may consider a number of concepts to be activated, each of which will have an associated support. The concept-mapping uses the context-phrase database to match input chunks to phrases or patterns in the context-phrase database. The concept-mapping aggregates the supports for each activated concept during the customer communication.

2.1.2 Concept and Resource Editor

The concept editor allows an administrator to add new concepts to the system. This involves selecting a place in the hierarchy to add the concept adding a name and corresponding context phrases with supports. This can be done manually or semi-automatically. In the latter case, the administrator specifies a set of documents which describe the concept. Using query expansion techniques using synonyms and manually added domain knowledge, the system then suggests where the concept should be placed in the ontology and which key-phrases should be associated with it. The user can accept or amend these as desired.

2.1.3 Query Engine and Query Processing

Upon receiving customer queries such as natural language queries and emails, phrase chunks are broken into real keywords and key-phrases, typically n-grams taken from the input where $n < 5$ [13]. The n-grams are compared to phrases in the fuzzy concept-matcher and a list of matching concepts with corresponding supports is found and, in the case of returning queries (a returning query is one that has been refined by the users during a query dialog process), used to update the list of current concepts, typically by summing the supports of the current concepts with those of the new concepts. Therefore the weight of returning concepts is incremented and it is possible to rank the concepts found iteratively.

The action selector looks for concepts of two types: task and non task. Tasks are abstract concepts e.g. fault, sales, pricing, overview etc. Each task can be associated with some non-tasks. For example a task called “broadband” will be associated with action concepts including “buying/tariffs”, “reporting fault”, “self-service”, and “billing”. The current Task item is considered to be that with the maximum support. Each topic is explicitly linked to a number of actions within the ontology. Based on which action concepts are found in the query the system selects appropriate topic/action pairs. In the case where the action concept cannot be isolated a default show_general_information is assumed.

The ontology also contains information about whether a topic/action pair concept is appropriate, or whether further dialogue with the user is necessary to narrow down to the precise concept. For example, it is appropriate to apply the action show_general_information to internet_access, but not the action sell since the customer must first choose between dial_up, midband and broadband. In this case the concept resolution dialogue presents the user with a list of possible child nodes, which the user can select. It repeats this until an appropriate node is found (typically this will be a leaf-node). Therefore it is important to link the right actions to the right topics in the ontology. This problem is inherent to ontology design and can make maintenance difficult.

2.2 Handing Queries Over to a Keyword Matching Strategy

Once a query has been disambiguated and even augmented with appropriate keywords it can be handed over to a search engine that is designed to pinpoint information. Classifying documents has been covered in numerous publications from automatic systems [9] using support vector machines, Bayesian classifiers [14] or at the other end of the spectrum, there are sophisticated systems using ontology based classification [8]. The solution presented here is a compromise between an automated system and a fully configurable one. The main advantage is that it allows for simple set up and low maintenance. Furthermore, classification is also exploited to help improve the searching process. The spirit of this approach is to be powerful and user friendly to both users and administrators.

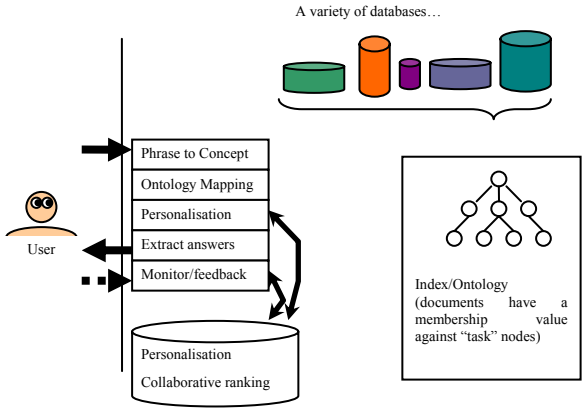

Figure 1: Overview of application.

Figure 1 shows an overview of the system. There are three major sections: at the top the databases, to the right the indexing system (shown as a list of different indexes) and to the left the query processing stack that receives a user query and returns answers whilst interacting with the bottom database containing all the personalisation and collaborative information (these two aspects

are not discussed in this paper, this work is discussed in [7]). Broadly speaking personalisation consists of associating keywords (from viewed documents) and repeatedly used query terms to individual users, and then using these keywords to enrich subsequent queries. Collaborative information is a way of exploiting this information for the benefit of other users, where frequent queries (from different users) are associated with good results (obtained from explicit feedback or moderated as in an FAQ system). The following sections describe the three main components of this system.

2.2.1 Database wrappers

This application has methods of extracting simple text from individual PDF files, databases or HTML. The indexer also extracts meta data (only found in HTML documents so far). This helps re-enforce the categorisation algorithm because manually provided information is considered more relevant than other keywords.

2.2.2 Indexing system

The indexing system is what gives this solution its specificity. First, documents are cut into series of small texts. At the moment they are chunked using known structure elements, for PDF files, it indexes pages separately, and for HTML it uses anchor tags as separators. This technique has two advantages, first, the index only refers to small pieces of text therefore queries can bring not only documents but precise location information.

Because the index is classified precision is necessarily increased, this requires the interface to support classified search. It is worth mentioning that in its current state, FocusSearch supports overlapping categories, and also, the ranking solution can include result hits from different categories. This happens because ranking is an aggregated value. Therefore, hits from outside of the queried categories can be listed albeit they will typically show a low relevance value.

2.2.3 Categorisation algorithm

The current data set is composed of technical files and health and safety information representing about 50MB (other techniques to obtain data automatically have been discussed in [4]). There are short and long documents (especially the health and safety section and the drivers handbook). Content providers identified the following categories: “safety”, “repair”, “provision”, and “quality”. These categories match the “non-task” concepts described in the section about the ontology. They describe generic concepts that users are likely query. Two other categories, overview, and general are provided for convenience. The former (overview) helps find menu pages, and the latter (general) is a fall-back solution for when categorisation is not helpful. Content providers also provided initial keywords to start the categories. In order to improve the classification, the algorithm has to help maintain these keyword lists following these three objectives.

In this implementation there are few leaf nodes, using a larger ontology would require a more complex classification able to manage class inheritance. This classifier has to be able to achieve an overlapping and not necessarily complete categorisation. Documents can hold multilateral information, therefore, themes (categories) can be densely or sparsely distributed within a document; for example, one of the categories created for this implementation is “safety”. This type of information is often found as a recurrent theme interweaved within technical documents throughout the text corpus. The categorisation help group such dispersed themes together allowing homogenous compounded views.

A list of keywords is associated with every category. The algorithm uses these terms to find out if a document belongs to a category by measuring their frequency against a confidence threshold (which may vary depending of the nature and size of the data set). The solution returns, for every keyword in the category, the *tf.idf* value of a match in the document. However, this value is also modified by an importance factor attached to the keyword. This factor is either set up manually and can also vary depending on whether a keyword is found in text or in meta data (so long as the document supports meta data). This weighting is described in a section below. These values are added up and measured against a confidence threshold set manually. Therefore, the classifier can only perform so long as it has good keywords.

This section describes how to maintain these keyword lists, i.e. remove, add and weigh keywords. There can also be some special categories that require ad hoc techniques. The application is able to single out menu pages such as table of content or HTML hypertext menus (linking to different sub sections in a text). Being able to index these pages separately is valuable because unsuccessful queries can be given a second chance, by finding the appropriate menu page. The current solution consists of identifying known pattern structures both in PDF and HTML. For example, PDF document table of contents are normally found within the first few pages and consist of bullet points. This is analogous to work described in [12], making use of hyperlinks to extend the reach of document indexing.

In order to improve the classification, the algorithm has to help maintain these keyword lists following these three objectives:

- Uncover keywords that can or should be removed.

- Discover new keywords

- Assign a weighting system to reflect keywords’ relative importance.

There are strong benefits in involving content providers (in our case, people from whom data was actually generated from) in the deployment phase. They are knowledgeable about key issues such as domain dependent vocabulary, rightful classification, and data maintenance processes. Once the application is set up, some manual tuning is possible (e.g. more knowledge such as acronyms and jargon can be added, or more key terms can describe existing categories) therefore maintenance is minimal.

2.2.4 Uncovering keywords that contributes poorly to the classification

For the algorithm to be able to run the categorisation has to be performed once with the manually chosen keywords. Once the categories exist the algorithm can measure in which category between poor, medium, and good a keyword is classified. The algorithm uses keyword co-occurrence frequency. The probability of a keyword k to belong to a category c (poor, medium, and good):

$$P(c)=n/d$$

Where d is the number of documents in a given category and where c is one of three possible values: poor, medium, and good. If c is poor, n is the frequency of k not co-existing (found in the same document) with any other keywords of its category. If c is medium, n is the frequency of all combinations of k and any other keyword from the same list. If c is good, n is the frequency of all

combination of this keyword and any other two keywords. The category with the highest experimental probability is the winning one. In the light of this keywords' quality measure, an administrator can be assisted in removing irrelevant keywords. However this task cannot be automated. There is no evidence that low co-occurrence is detrimental to the quality of the classifier, it is only an indication.

2.2.5 Discovering new keywords

Again based on the assumption that co-existing keywords help indicate a connection to a category. A pre-processing step consists of extracting a complete picture of key terms (keywords, and phrases) co-occurrence. In order to avoid long computations a term co-occurs with another one so long as it is found in the same sentence. Already running this type of computation on the 50 MB of data can take several days with a recent PC. Initially two complementary strategies are used subsequently. First, WordNet is used to help highlight any instance. The second method consists of systematically computing the frequency of keywords following one another. This can highlight compounds of any size in the text corpus. A threshold as to how many times a key phrase should be found before it can be considered relevant is defined empirically, as a rule of thumb, it usually ends up being a number between five and ten. Key phrases discovered using the latter method can show noisy characteristics, nevertheless, they are very valuable because they are meaningful and often specific to the domain. For example, the following key phrases: "asbestos fibre", "asbestos containing material", "breathing apparatus", "safety helpdesk", and "safety legislation" were discovered as potential candidates to the "safety" category. This algorithm is also extremely time consuming because of the rapid explosion of combinations.

Looking up co-occurrences to the word "repair" shows the following: "fault", "closure", "redcare" (a BT internal network alarm system), "adsl", and "provision". Amongst these keywords, two are general terms: fault and provision. Three are domain specific: "closure", "adsl", and "redcare". The word "fault" is obviously relevant to this category. Other words are only relevant when found co-occurring with keywords from the "repair" list: "closure", "adsl", and "redcare". Finally, "provision" belongs to another category which illustrates the overlapping nature of the data. The challenge of the algorithm is to be able to select the right keywords in order to improve the classification. The algorithm follows these steps:

- Work out frequency of co-existing key terms for the entire data set

- Work out co-existing relevance threshold (and maximum frequency)

- Extract relevant keywords

The first step has already been described, the second step will return the minimum frequency of co-occurrence between two individual terms before they can be considered complementary to one another. A co-existing relevance threshold is calculated for every keyword in relation to two factors: the size of the data set, and the frequency of individual keywords. The role of function T is to set the frequency threshold value of keyword k:

$$T(k)=(f \,.\, p)+((m \,.\, (1/(1+e\text{-}x)\,))/2)$$

Where x represents the frequency f of k normalised between minus *5* and *-5*, p is a threshold percentage value (here set to 5%), m is the maximum percentage value of representation of a keyword. Therefore this function boosts thresholds for low frequency keywords by adding up to half of m (maximum value of $f.p$ for the same list), using a sigmoid function to obtain a factor: $1/(1+e\text{-}x)$. Keywords co-existing above this threshold are potential candidates for the category lists. However, the algorithm applies constraints. It is desirable only to keep nouns because it has been observed that verbs, adverbs, and adjectives do not carry enough meaning. This can be achieved using tools such as the Brill tagger [2]. Also, tests have shown that some noise can be introduced by high frequency keywords. Therefore, these have to be removed from an automated system. This threshold is empirically set as 0.1% of the number of keywords (excluding stop words).

2.2.6 Weighting keywords found

The most popular keyword weighting scheme for a keyword is the *tf.idf* algorithm. In this case however, weighting refers to entire categories and not discrete documents. The amount of co-existence with keywords of the same category is a closer measure to what the algorithm is trying to achieve. This however has to be corrected in relation to the frequency of the keyword in question. Therefore, the weight p can be calculated as the observed probability of a keyword k to co-occur with at least one other listed keyword. Therefore p is the measure of how representative a keyword is of its category based on how often it co-occurs with other keywords of the same category.

$$p(k)=n/d$$

Where d is the number of documents in a given category and n is the frequency of all combinations of k and at least one other keyword from the same list. However, at run time the goal is to find out whether a document belongs to a category. All keywords belonging to the said category are searched in documents; if they are found, the extracted value is their *tf.idf* weight multiplied by their observed probability p. This alteration values the individual value of keywords as well as the importance they play within each document. Therefore the formula to work out the weight of a document belonging to a category is:

$$\sum_i tf.idf_i \,.\, p_i$$

Using this weighting scheme, it is possible to measure how much a document belongs to a category by working out the sum of all the keywords it may contain. In order to decide whether a document belongs to a category, a threshold value is set up manually by trial and error.

2.2.7 Results:

Beyond technical implementation details it is also its configuration flexibility that makes the success of this system. The solution is intended to be used for Information Retrieval problems with domain knowledge constraints and low involvement for system administrators. Bootstrapping the classifier with few keywords is easy to achieve and can be sufficient to run the application in real conditions. This flexibility in deploying the application with customised feature has allowed it to be under trial within BT; the trial is under way at the time of writing this paper.

Figure 2 shows the classifier's performance using the manually selected keywords against the results obtained with the algorithm. There is an improvement in both the recall and precision. However, it is easy to raise the recall measure automatically but not the precision one. Scrutinising the results per category showed

that classes that had already many keywords showed little improvement, and were more prone to noise, whereas categories with few keywords tended to easily pick relevant keywords. Therefore the automation works fine until the recall level reaches about 90%, beyond this stage, recall will still improve but only to the detriment of precision. Other improvements can be achieved by lowering the relevance threshold and selecting keywords manually.

3. Conclusion

Customer self-helps will continue to be the focus of any customer contact centres in order to reduce their costs. This paper outlines a vision and presented a hybrid ontology/keyword matching approach. An internal trial is under way within BT at the time of writing this paper. It indexes a large database of technical content together with quality, provision, and health and safety information. This technical data is only accessible to BT engineers.

We are also developing other software which provides smooth flow from customer self-help to operator's assisted self-help based on the semantic web technology. We believe call centre information search is a different kind of search. The current technology used by major global search engine companies such as Google [10] would not be suited and an ontology based semantic indexing is a promising alternative.

There is scope to improve both parts of the system, and there is even a promising synergy in this hybrid system. The ontology is extremely useful to provide a view of the final classification. The keyword matching indexing has a keyword discovery algorithm that can benefit the ontology. Therefore the future research is looking at ways of simplifying the introduction of meta data automatically. Disambiguated semantic information is the only way to push the quality of Information Management applications. The representation of meta data has to be kept in an ontology and populated automatically or semi-automatically. The algorithm described here can help suggest keywords that can enhance an ontology that has been partially populated already.

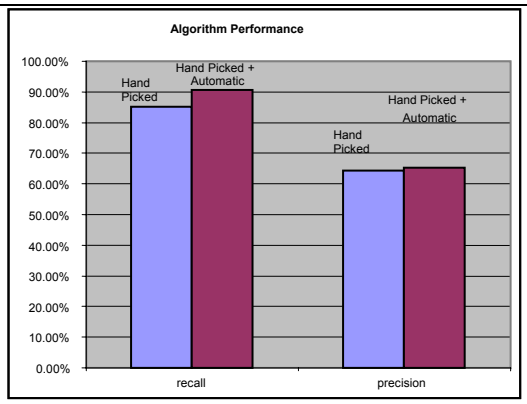

Figure 2: Precision and recall measures for the algorithm selecting keywords automatically. The automatic selection improves both recall and precision.

Also using reasoning languages such as OWL combined with powerful Ontology browsing techniques it will be even possible to make sensible connections that had to be made manually before. Inferring information will further automate maintenance of future Information Search technology.

The main attraction of the semantic web will be to help uncover new sources of information that are relevant to end users. However, using this information is challenging because there is no data quality checks. By providing information through the semantic web, corporate institutions must accept responsibility for content even when it is provided by third party entities. In order to adopt the semantic web, reliable data quality checks will have to be available. This includes challenges such as traceability, or the ability to update superseded information.

The benefit between a self-service application and the semantic web can be reciprocal. On one hand information endorsed by corporate institutions and on the other an information structure built and understood by users. Ultimately the semantic web is the missing link between users and the information they need. However it is not clear how end users will use this tool to help sort their problems themselves. Should users take a pro-active role in filtering out dated information, should they contribute to the semantic web for their benefit and for their peers? Or should self-service applications use the semantic web to automatically provide support to customers without seeking contribution?

The philosophy of the semantic web is to rely on communal work. The contribution of each and everyone helps towards the overall structure of information. New possibilities are added when new contributions are validated (e.g. through statistical relevance), also, some connections can be discarded when support is discredited or no longer compelling. The semantic web has to prove that it can organise contribution to the benefits of all. Issues such as quality and validity of information also have to be addressed to help provide a network can classify and search data using models that are closer to our own way of mentally representing information.

4. REFERENCES

[1] Bradshaw D. Web self-service - empowering customers while saving money, Ovum Report, December, 2004

[2] Brill E. http://research.microoft.com/users/brill

[3] Case S., Assadian B., Ducatel G., and Thint M. IIM Report on Contact Centres. BT Group Internal Report.

[4] Case S., and Thint M. Information management assistants for enterprise workers, BTTJ Vol 21, No 4, 2003

[5] Cui Z., Ducatel G., Thint M., Assadian B., and Azvine B. Towards Automated Customer Self-Help, BT Technological Journal Vol 24, No1, 2006

[6] Cui Z., Tamma V., and Bellifemine L. Ontology Management in Enterprises, BT Technological Journal Vol 17, No4, 1999

[7] Ducatel G., and Nürnberger A. iArchive: an Assistant to Help Users Personalise Search Engines. In Enhancing the Power of the Internet, Masoud Nikravesh, Ben Azvine, Ronald Yager, Lotfi Zadeh (Eds), pp 351-362, Springer-Verlag, 2004

[8] Fagin R., Kumar R., McCurley K. S., Novak J., Sivakumar D., Tomlin J. A., and Williamson D. P.

Searching the Workplace Web. In Proceedings of WWW2003, Budapest, pp 366-375, ACM Press, 2003

[9] Glover E. J., Tsioutsiouliklis K., Lawrence S., Pennock D. M., and Flake G. W. Using We Structure for Classifying and Describing Web Pages. In Proceedings of the WWW2002, Honolulu, 2002

[10] Google search site: http://www.google.com

[11] Kolsky E. MarketScope for Web Self-Service, 2H04; Gartner Report, January, 2005

[12] Kraaij W., Westerveld T., and Hiemstra D. The Importance of Prior Probabilities for Entry Page Search. In Proceedings of SIGIR'02, Tampere, pp 27-34, 2002.

[13] Manning C. D., and Schutze H. Foundations of Statistical Natural Language Processing, The MIT Press, 1999

[14] McCallum A. and Nigam K. A. Comparison of Event Models for Naïve Bayes Text Classification. In AAAI-98 Workshop on Learning for Text Categorization. 1998

[15] McIlraith S., and Martin D. Bringing Semantics to Web Services, IEEE Intelligent Systems, Vol 18 nb 1, pp 90-93, 2003.

[16] Miller G. A. http://wordnet.princeton.edu.

[17] W3C Semantic Web Initiative: http://www.w3.org/2001/sw