Context Filters for Document-Based Information Filtering

K. R. K. Murthy
Dept. of Computer Science and Automation
Indian Institute of Science
Bangalore - 560 012
India
E-mail: murthy@csa.iisc.ernet.in

S. S. Keerthi
Dept. of Mechanical and Production Engg.
National University of Singapore
10, Kent Ridge Crescent
Singapore 119260
E-mail: ssk@nus.edu.sg

Abstract

In this paper we propose a keyPhrase-sense disambiguation methodology called "context filters" for use in keyPhrase based information filtering systems. A context filter finds whether an input keyPhrase has occurred in the required context. Context filters consider various factors of ambiguity. Some of these factors are special to information filtering and they are handled in a structured fashion. The proposed context filters are very comprehensibile. Context filters consider varieties of contexts which are not considered in existing word-sense disambiguation methods but these are all needed for information filtering. The ideas on context filters that we report in this paper form important elements of an Instructible Information Filtering Agent that we are developing.

1. Introduction

Information filtering is the process of separating out irrelevant documents from relevant ones. Its importance has motivated several researchers to develop software agents such as SIFT, InfoScan, iAgent, RAMA, MailFit, MailAgent, SearchPad, Topic Search'97, ZyFilter, GroupLense, NewT [1].

Two important characteristics of information filtering domain are (1) dynamic documents' collection and (2) Long term interest of the user. Characteristic 2 can give rise to the evolution of rich involvement of user. This can be achieved only when the decision making process is comprehensible to the user. Expertise of the user is important since rich and accurate feedback can be expected from the user which will hasten the learning process. Rule based systems are best suited for good comprehensibility. Therefore we will consider only rule based systems in this paper.

Many a times a keyPhrase used in documents can represent more than one sense. In keyPhrase based information filtering systems, if a keyPhrase is used without finding the proper sense of that keyPhrase, it will lead to some uncertainty in the filtering process. In NLP there is a significant interest in word-sense disambiguation. This interest has led to the development of several word-sense disambiguation systems. These are either knowledge-base based [2,9] or lexicon based [3,8] or corpus (statistics) based [4,7] methods. Section4 explains how the scheme proposed in this paper is better suited than the existing ones for information filtering.

In this paper we first describe a few factors of uncertainty in information filtering. Then we propose a methodology called context filters to reduce the uncertainties caused by these factors. Some of these uncertainty factors are not handled in traditional word-sense disambiguation systems. This is mainly because these noise factors are of concern in information filtering domain only. Also the methodology we provide is based on comprehensibility and easy handling of filter building process. This methodology allows the user to build context filters in various modes such as learning with minimal interaction (passive learning of filter), learning by interaction (active learning of filter) and hand crafting the entire filter. Since the proposed methodology is both comprehensible and flexible, sharing of partial filters between users is well supported. This relieves the user from building the context filters from scratch. Context filters correctly recognize proper supporting context for a keyPhrase. This will be helpful in extracting relevant paragraphs of a document for a given relevance rule on confirmed keyPhrases. Context filters are not limited to rule based systems; these can be equally applicable in the other systems also.

Elaborate discussion of the ideas presented in this paper appears in [5].

2. Factors of Uncertainty

Noise in information filtering is contributed partly by the uncertainty in the usage of keyPhrases and partly by the uncertain relation of presence/absence of keyPhrases to the presence/absence of a topic in the document. A few factors of uncertainty are as follows.

Implicit keyPhrases: In a document, if a topic of discourse is not accompanied by the corresponding keyPhrase then such a keyPhrase is called as an implicit keyPhrase. In this situation, either some of its synonyms can be present or its implied phrases are present. For example, in the absence of the phrase "information filtering" , either its synonym "personalized filtering" can be present or its implied phrases "conference" and "information retrieval" can be present.

Pseudo keyPhrases: If a keyPhrase is not accompanied by the required topic or if it has little relevance to the topic then the keyPhrase is called pseudo keyPhrase. This does not refer to the intended broad sense at all. For example, the keyPhrase "information filtering" may refer to either document filtering or communication data filtering. If the required topic is document filtering, occurance of "information filtering" in documents other than document filtering documents will not be relevant.

Broadly defined keyPhrases: If an occurance of a keyPhrases is only broadly related to the required topic, then it is called broad keyPhrase. For example, occurance of the keyPhrase "information filtering" in a document may be referring to document filtering but it can refer to either html document filtering or news filtering or e-mail filtering. Then the exact sense of reference has to be found out.

Nonlocal keyPhrases together defining (ir)relevance: Two keyPhrases which occur far off from each other may define (ir)relevance. This may not be correct in all cases. For example, for a relevance rule "information filtering & genetic algorithms", there can exist one or more paragraphs discussing each one of the above topics. But none of the paragraphs have a discussion of both. In that case a simple presence of the two keyPhrases somewhere in the document is not sufficient for deciding the presence of a topic that requires a interrelated presence of both keyPhrases.

The next section presents a scheme to reduce the noise caused by the above uncertainty factors.

3. Context Filters

Solving the problems described in the last section will eliminate a lot of noise and thereby improve filtering. We use phrases called contextPhrases along with their contexts of occurance as features of a (keyPhrase, position) pair to solve these problems. Therefore, the features (or, attributes) used in context filters are (contextPhrase, context) pairs. Depending on to which (keyPhrase, position) this feature is referring, it takes a value either TRUE or FALSE i.e. value set of these features is $\{$TRUE, FALSE$\}$. The value set of context is $\{$*in-local, local, near-local, global*$\}$ and the value is found out with respect to a given (keyPhrase, position) pair. All these contexts are defined using windows around the (keyPhrase, position) in question. If the position of a keyPhrase in a document is P, then this keyPhrase is said to have occurred in the context 'C' of a context phrase only if this context phrase occurs in the position P_c satisfying the "window" condition $(P - W_c) <= P_c <= (P + W_c)$, where W_c is the window size of context 'C'. Depending on the window size we define four types of contexts. These are: (1) In-Local context (2) Local context (3) Near-Local context and (4) Global context. These contexts use different window sizes satisfying the relationship: $W_{in-local} < W_{local} < W_{near-local} < W_{global} = DocumentLength$. Even though these definitions are valid for structured documents also, improved definitions are given in [5].

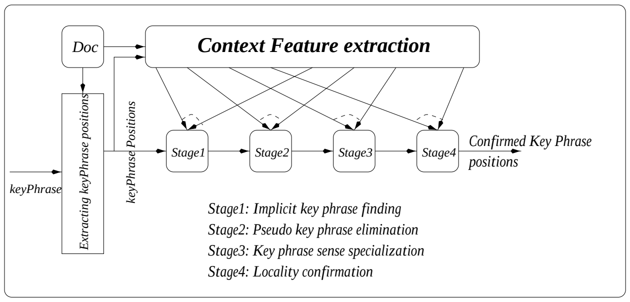
Every keyPhrase position is passed through a filter called context filter. This context filter filters out a keyPhrase position if it has not occurred in a specified positive context or if it has occurred in a specified negative context. Positive context is the requirement of occurance of a contextPhrase in a given context and negative context is the requirement of absence of a contextPhrase in a given context.

There are, mainly, three reasons for not including contextPhrases in relevance definition directly but for using in a separate filter. These are as follows.

Relatively stable context rule along users and time axes: Context definition of a keyPhrase for a particular user group remains stable over time. Also, this can be used by other users who are working on similar streams of documents. On the otherhand relevance definition keeps varying from user to user and time to time. Therefore it is a good idea to process the documents in two stages rather than in a single stage.

Prohibitive rule length if used in relevance definition directly: Another reason for not including contextPhrases directly in the relevance definition is that it can lead to longer rules. Rule length, sometimes, can be prohibitive. Higher rule lengths can make the rule incomprehensible and can also slow down the learning.

Uniform applicability of this filter in entire relevance definition: The context filter applied on a keyPhrase is applicable wherever this keyPhrase appears in the relevance definition. So it is more meaningful to have a short augmentation to the keyPhrase name by the corresponding context filter name rather than augmenting with all of its positive and negative contextPhrases. This will definitely reduce rule length thereby improving the comprehensibility and speeding up the learning.



Doc
Context Feature extraction
keyPhrase
Extracting keyPhrase positions
keyPhrase Positions
Stage1
Stage2
Stage3
Stage4
Confirmed Key Phrase positions
Stage1: Implicit key phrase finding
Stage2: Pseudo key phrase elimination
Stage3: Key phrase sense specialization
Stage4: Locality confirmation

Figure 1. Context processing of a keyPhrase positions

3.1. Structure of context filters

We suggest a context filter as a series of four context filter stages as shown in figure 1. Each stage removes uncertainty caused by one of the factors discussed in the previous section. Input of any stage is a list of all positions of the keyPhrase confirmed by its earlier stages. Each context filter works on only one keyPhrase. Functions of these stages are indicated in the figure. Working of the various stages is as follows.

Stage 1 outputs all input positions and positions which meet the conditions described in this stage. Stages 2 and 3 pass only those positions which are present in their input list and which meet conditions in these stages. Stage 4 eliminates all positions which are not occuring in the specified context of other keyPhrases. Stage 1 only adds positions whereas other stages only eliminate positions.

This order is important for a number of reasons and it has to be understood from either of a correctness view point or a comprehensibility view point.

Position of stage 1 can be supported from both of the above view points but mainly from correctness view point. This is because unlike other stages stage 1 adds new positions to the input list. If it is placed after any other stages then these added positions will not be processed by all of its preceding stages, which is not correct. Position of stage 1 can be understood from the comprehensibility view point also. Since all remaining stages process an occurring keyPhrase and they qualify it further, it is more meaningful to first explore all possible positions of the keyPhrase which are captured by the implicit keyPhrase stage.

The order of the remaining three stages in the context filter is supported from comprehensibility view point. This is because unlike the implicit keyPhrase stage, a position is filtered out if any of these three stages filter it out; therefore, correctness will not be affected by any change of the order of these stages. Locality finder is meaningful only when it is the last stage since this is supposed to work on valid keyPhrases (keyPhrases with right sense) only. Any checking for correct locality of an invalid keyPhrase with reference to another keyPhrase is not meaningful. KeyPhrase specializer specializes the broad sense of a given keyPhrase to the required sense. Obviously for this is to be done, the sense of input keyPhrase has to be right in the broadly accepted sense of that keyPhrase. Since pseudo keyPhrase eliminator outputs only those keyPhrases whose senses are broadly accepted, the position of keyPhrase specializer has to be after pseudo keyPhrase eliminator. This justifies the ordering of the various stages.

3.2. Decision Models and Operation

We propose the use of "decision lists" [6] to represent context filters. A "decision list" is a list L of pairs $\{(f_1, v_1), (f_2, v_2), ...(f_r, v_r)\}$. where each f_i is a test, each v_i is class label, and f_r is the constant function, TRUE. A decision list L defines a classification function as follows: for any input **x**, L(**x**) is defined to be equal to v_j where j is the least index such that $f_j(x)$ = TRUE. These tests can be anything such as perceptron test, cnf, dnf or conjunct of attributes.

Features or attributes used in context filter decision lists are (contextPhrase, context) pairs as described earlier. In decision list tests, these are used in either complemented or uncomplemented form. A decision list of conjuncts of features is chosen because any DNF can be expressed as a decision list of conjuncts and vice versa; and hierarchical structure of decision list is more natural for information filtering domain.

The following example describes preliminary context filters for the keyPhrases "information filtering". *Example1:* This is for the phrase, "information filtering" used in the document filtering sense. This is to be filtered for html documents filtering using machine learning techniques. Decision lists of various stages are given below.

Stage1: {((("call for papers", global) & ("information retrieval", in-local)), TRUE), (("document filtering", in-local), TRUE), (TRUE, FALSE)}

Stage2: {(((document, global) & (user, global)), TRUE), (TRUE, FALSE)}

Stage3: {(((html, global) & (url, global)), TRUE), (TRUE, FALSE)}

Stage4: {(("machine learning", local), TRUE), (TRUE, FALSE)}

Output of stage1 will be TRUE for every position of the phrase "information filtering". Output will be TRUE for positions at which this phrase does not occur, but "call for papers" appears in *global* context and "information retrieval" occurs in *in-local* context. In this case these two contextPhrases imply the presence of topic represented by the keyPhrase. Similarly output will be TRUE for positions at which the phrase “document filtering” occurs. This is a synonym of the keyPhrase. Otherwise output of all other positions will be FALSE as suggested by default test. Since we are looking for “information filtering” in document filtering sense, stage 2 looks for the contextPhrases “document” and “user” in *global* context. If this condition is not met for any position of the keyPhrase then that is a pseudo occurance of the keyPhrase and therefore it will be filtered out. Specific definition is for html document filtering, and hence stage 3 looks for the contextPhrases “html” and “user”. If this is not met, such positions will be filtered out. Since we are looking for paragraphs talking about machine learning applied to information filtering, it looks for the keyPhrase “machine learning” in *local* context of every keyPhrase position. All positions which are not meeting this requirement will be filtered out and all positions which are meeting this will be expanded just enough to cover nearby phrase “machine learning”.

4. Suitability of Context Filters

This section presents the features of the proposed approach which makes it better suited for information filtering than the existing ones. These are as follows.

Structured handling of noise: Noise is handled in a staged manner i.e. from more general to specific definition of keyPhrase, and each stage has well defined function. In addition, function of a contextPhrase is also well defined based on which stage is employing them. Result of these two factors is good comprehensibility of the operation of the stages which will reduce the *credit assignment problem* i.e it is easy to the find culprit stages that contribute to improper filtering.

Handling noise on implicit keyPhrases and non-local keyPhrases will reduce the noise further. These are absent in existing systems.

Use of rich contexts: Local, *near-local* and *global* contexts are also crucial in information filtering. But existing systems use, mainly, *in-local* context and very few systems use *local* context and none uses the remaining two contexts.

Flexible model usage: Decision lists facilitate both incremental and batch learning. Since these are the decision models, context filters can be developed on-line, off-line or combination of both. Decision lists developed by different users can be combined to give better performance.

No need for lexicons or corpus: For new sources of documents it is difficult to provide either good lexicon or good corpus. The proposed approach needs neither of them; instead it can be built either on the fly by one user or collaboratively by several users.

Looking only for the required sense: The proposed model looks only for the required sense of a keyPhrase instead of all senses. It reduces the search space, there by speeding up the learning process.

Flexibility of selection of phrases: A given phrase can be chosen either as a context phrase or as a keyPhrase depending on user’s requirements.

5. Conclusions

In this paper we first presented various noise factors in keyPhrase based information filtering systems that need to be resolved for improved performance. Then we proposed a new structure called context filters to resolve these noise factors, and defined its features and working. Finally we provided a brief comparative analysis between the proposed method and existing word-sense disambiguation systems.

References

[1] IFsite, Information Filtering Resources, *http://www.ee.umd.edu/medlab/filter/filter.html.*

[2] Hirst, G., *Semantic Interpretation and the Disambiguation of Ambiguity*, Cambridge University Press, England, 1987.

[3] Li, X., Szpakowicz, S., Matwin, S., A WordNet-based Algorithm for Word Sense Disambiguation, *In Proc IJCAI-95*, 1368-1374, 1995.

[4] Pedersen, T., Raw Corpus Word Sense Disambiguation, *In AAAI-98*, *Madison,WI.*, July 28-30, 1998.

[5] Murthy, K.R.K., Keerthi., S.S., Context filters for document-based information filtering, *TR-ISL-99-KRK01*, *http://144.16.67.112/˜krish*, 1999.

[6] Rivest, L.R., Learning Decision Lists, *Machine Learning Journal*, 2:229-246, 1987.

[7] Brown, P.F., Della Pietra, S.A., Della Pietra, V.J., Mercer, R.L., Word-Sense Disambiguation using Statistical Methods, *In Proc. ACL meeting, Berkeley*, 264-270, 1991.

[8] Sussna, M., Word Sense Disambiguation for Free-Text Indexing Using a Massive Semantic Network, *In CIKM’93*, 1993.

[9] Wilks, Y.A, A Preferential, Pattern-Seeking Semantics for Natural Langauge Inference, *Artificial Intelligence* Vol6, 1975.