

Towards Topic-Based Summarization for Interactive Document Viewing

Achim Hoffmann

School of Computer Science and Engineering
University of New South Wales
Sydney, 2052 NSW, Australia
achim@cse.unsw.edu.au

Son Bao Pham

School of Computer Science and Engineering
University of New South Wales
Sydney, 2052 NSW, Australia
sonp@cse.unsw.edu.au

ABSTRACT

Our research aims at interactive document viewers that can select and highlight relevant text passages on demand. Another related objective is the generation of topic-specific summaries of texts as opposed to general purpose summaries. This paper introduces our notions of *discourse structure tree* and *level-of-detail tree*. Both structures are used to represent relevant aspects of a text segment for the above mentioned purposes. Furthermore, we introduce a Knowledge Acquisition Framework for DIScourse processing (KAFDIS) that allows the incremental and efficient acquisition of knowledge for the reliable construction of the discourse structure graph and the level-of-detail tree based on cue phrases. An effective knowledge acquisition process is crucial to allow the economical development of systems that can handle a large variety of topics. Our knowledge acquisition approach ensures always a consistent knowledge base whose semantics are well controlled by the expert. It is an incremental approach that allows patching of the knowledge base as soon as the need arises without causing any inconsistencies. We also present promising experimental results with our approach.

Categories and Subject Descriptors

I.2.7 [Artificial Intelligence]: Natural Language Processings; I.2.4 [Artificial Intelligence]: Knowledge Representation Formalisms and Methods; I.7 [Document and Text Processing]: Miscellaneous

General Terms

Algorithms, Human Factors

1. INTRODUCTION

The advent of the Web has made billions of documents available in electronic format. This gives us more access to a wide pool of knowledge but also causes the problem of in-

formation overload. We simply do not have enough time to digest even a small portion of the available information. Automatic support for digesting the available documents on the World-Wide Web has been developed chiefly in the form of information retrieval systems and web search engines. Another important line of research deals with automatic text summarization.

This paper proposes to go beyond traditional text summarization by providing help in two forms: a) Sophisticated assistance when reading documents on the screen by automatically highlighting portions of the text that are deemed relevant to the readers' interest, possibly by using multiple windows to allow the simultaneous display of text from multiple pages. b) The automatic generation of topic-based summaries of a document.

The idea of interactive document viewing involves an intelligent document viewer that can colour-code various portions of the text as well as allowing users to interactively request additional information such as the motivation for some techniques, justification for a statement, the definition or explanation of technical terms, the limitations of a technique or approach, etc. The user would indicate some sentences or phrases and then select a particular type of request by pressing a key. The requested information can be displayed in a separate window by marking the relevant parts of the original text. Such an intelligent document viewer has the potential to substantially reduce the time required to capture the parts of the document a reader might be interested in. This allows users to drill deeper into a topic where required and otherwise just to skim through a document or to skip large parts altogether.

Automatic text summarization is an established field of research. The selection of the most telling chunks, usually sentences, of the original text proved to be the most feasible approach. Generally, the detection of cue phrases along with statistical measures to select useful sentences proved reasonably successful, see e.g. [8]. The existing systems, some of them being commercially exploited, provide some kind of quick impression of what the document is about. However, for many tasks the produced summaries are still far from satisfactory.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

K-CAP'03, October 23–25, 2003, Sanibel Island, Florida, USA.
Copyright 2003 ACM 1-58113-583-1/03/0010 ...\$5.00

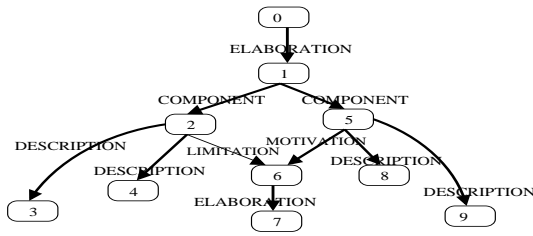


Figure 1: A discourse structure graph of the first paragraph of section 2. Node i represents the i^{th} sentence in the text.

Research on text summarization so far focused on the creation of a general purpose summary of a text. This is in contrast to the fact that for complex documents, different users will have different information needs which makes different parts of the original text relevant. For example a scientist might be interested to know the experimental approach being used in the study that a paper describes. To assist in that, a specialized summary emphasizing the experimental method used needs to be produced. At other occasions, a scientist might be interested to learn about the basic idea of the new technique being presented in a paper, without wanting to know all the details of the technique or the merits and limitations of the approach.

A major problem that needs to be overcome is given by the fact that for the automatic construction of topic-based summaries we will need to tailor our tools towards the topic as well as towards the domain of the documents to be processed. For instance, means to recognize useful sentences about the experimental method used are likely to vary substantially for different disciplines, e.g. in computational linguistics, physics or sociology. This implies that we need effective methods that allow us to economically tailor our tools towards a new topic and/or a new domain.

We propose to build a discourse structure graph and a level-of-detail tree. Both structures are constructed on the basis of recognizing significant cue phrases in the text. We demonstrate how those structures can be utilized to extract a short summary of different aspects of a document.

To address the mentioned problem of economically tailoring our tools, we introduce a new incremental knowledge acquisition approach, implemented in our Knowledge Acquisition Framework for DIScourse processing (KAFDIS). KAFDIS allows us to effectively build a knowledge base that controls the construction of the discourse structure graph of a text segment as well as the level-of-detail tree. Our knowledge acquisition approach borrows ideas from Ripple Down Rules [3] but substantially extends this known technique by introducing a new rule language making it applicable to acquiring knowledge for natural language processing.

The paper is organized as follows: In section 2 we introduce our level-of-detail tree and the discourse structure graph. Section 3 discusses related work. In the following section 4 we describe how to build knowledge bases that control the

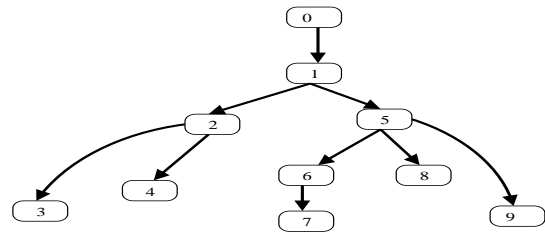


Figure 2: A level-of-detail tree of the first paragraph of section 2. Node i represents the i^{th} sentence in the text.

generation of the graph and tree structures. Section 5 presents our initial experimental results. Finally, the conclusions are presented in section 6.

2. OUR APPROACH

⁰The crucial step in our approach towards forming topic-based summaries of a document is to annotate the existing document by relating the individual sentences in a text segment, such as a section, to each other. ¹To do that we create two structures: ²The first one is the *level-of-detail tree*. ³The tree assigns one node to each individual sentence and edges to relate sentences to each other. ⁴The level-of-detail tree is thought to reflect the hierarchical structure that is typically found in technically written papers. ⁵The second structure is the *discourse structure graph*. ⁶The *discourse structure graph* is meant to reflect additional information that goes beyond the scope of the level-of-detail tree. ⁷Such information will reflect relationships among sentences that are other than different levels of detail. ⁸The discourse structure graph assigns a node to each sentence and directed labelled edges between sentences to indicate the respective relationship. ⁹The discourse structure tree can be traversed in order to identify the role of a sentence within the discourse in a text segment.

2.1 How the graph structures are being used

The discourse structure graph and level-of-detail tree of the previous paragraph are shown in figure 1 and 2 (where the superscripts show the numbering of the following sentences).

The level-of-detail tree can easily be used to generate a summary of adjustable length by just picking the desired number of sentences from the top of the tree. E.g. a summary of length 1 would result in only the first sentence 0, while a summary of length 4 would result in the sentences 0, 1, 2, and 5.

Unlike the discourse structure tree, the level-of-detail tree is not suited to find, e.g. the limitations of the tool *level-of-detail tree* as explained in sentence 6. If, e.g. in an interactive viewing scenario, the user wants to know the limitations of the level-of-detail tree, an answer can be generated by traversing the discourse structure graph in Figure 1 scanning the sentences for the key phrase *level-of-detail tree*, which occurs in sentence 2. From sentence 2 one can easily arrive at sentence 6 containing limitations of the level-of-detail tree with an elaboration in sentence 7. Similarly, for summarizing certain aspects of the text segment we can strip the text

segment of the motivational part of the discourse structure graph (again in sentence 6 and 7) if that is desired. It is also easy to extract the description of only a particular item, such as the level-of-detail tree by identifying sentence 2 by matching it against the phrase and selecting sentence 3 and 4 as further elaboration.

2.2 Discourse Relations

A text is more than just a random collection of sentences but there are discourse relations linking portions of text together making the text coherent. However, there is currently no overall agreement about how to define a standard set of discourse relations. The number of relations proposed varies in terms of semantics [6], intentions [5] or a combination of both [10]. As noted by [12], although current discourse theories are built on fundamentally different principles, they all share some common intuitions. Using our approach, whenever an expert specifies that a relation holds between two sentences, (s)he needs to justify the decision so that KAFDIS can form a rule from that. We do not define the relations used in this work formally but allow the experts to use their intuition. The experts could add more relations as the need arises. In the end, there were 22 different relations of which ELABORATION, COMPONENT, SEQUENCE, DESCRIPTION, DEFINITION, COMPARISON, CONTRIBUTION, ASSUMPTION, EXPERIMENT, CONSEQUENCE were most frequently encountered.

2.3 Building the discourse structure graph and level-of-detail tree

Let S_0, S_1, \dots, S_n denote $n + 1$ sentences of the text segment in order. Figure 3 shows the pseudo code on how to build the discourse structure graph and the level-of-detail tree.

We will now describe how to construct both the discourse structure graph and the level-of-detail tree.

The graph and tree are built together incrementally by processing one sentence at a time in the order of their appearance in the text segment. The level-of-detail tree is a sub-graph of the discourse structure graph and is used to guide the expansion of the discourse structure graph. At step i , we have a discourse structure graph G_{i-1} and a level-of-detail tree T_{i-1} . We consider how to expand G_{i-1} and T_{i-1} to G_i and T_i respectively by adding S_i . Initially we start at step 1 with G_0 and T_0 containing only S_0 . G_i is constructed first based on G_{i-1} and T_{i-1} . Next, T_i is constructed using T_{i-1} and G_i .

In building the discourse structure graph and level-of-detail tree, we use a graph rule-base and a tree rule-base respectively (See section 4 for details).

2.3.1 How to build the discourse structure graph

To construct G_i by adding S_i to G_{i-1} , we have to decide which node in G_{i-1} to link S_i to. Given a node S in G_{i-1} and the current sentence S_i , there is a graph rule-base that decides whether there is any relation between the two, and which relation it is. However, there are situations where G_{i-1} has the same sentence at multiple locations. One sen-

```

 $G_0$  and  $T_0$  contain only  $S_0$ 
for  $i = 1$  to  $n$  do
   $G_i = G_{i-1}, T_i = T_{i-1}, LinkSet = \{ \}$ 
   $CandidateSet = \{ P_1, \dots, P_k \text{ \& their siblings} \}$ 
  for all nodes  $target$  in  $CandidateSet$  do
     $relation = \text{graph rule-base}(target, S_i, T_{i-1})$ 
    if  $relation \neq null$  then
       $G_i += \text{edge}(target, S_i, relation)$ 
       $LinkSet = LinkSet + \{target\}$ 
    end if
  end for
  if  $LinkSet \cap \{ P_1, \dots, P_k \} == \{ \}$  then
     $G_i += \text{edge}(S_{i-1}, S_i, UNKNOWN)$ 
     $LinkSet = LinkSet + \{ S_{i-1} \}$ 
  end if
   $CandidateSet = \{ \}$ 
  for all nodes  $target$  in  $\{ P_1, \dots, P_k \} \cap LinkSet$  do
     $val = \text{tree rule-base}(target, S_i)$ 
    if  $(val == true)$  then
       $CandidateSet += \{ target \}$ 
    end if
  end for
  if  $CandidateSet == \{ \}$  then
     $S = \text{highest index in } \{ P_1, \dots, P_k \} \cap LinkSet$ 
     $T_i = T_{i-1} + \text{edge}(S, S_i, G_i(S, S_i))$ 
  else
     $S = \text{sentence with highest index in } CandidateSet$ 
     $T_i += \text{edge}(S, S_i, G_i(S, S_i))$ 
  end if
end for
return  $G_n$  &  $T_n$ 

```

Figure 3: The graph and tree construction algorithm in pseudo code.

tence can be directly related to S_i while the other is not. This is because the two appear in different contexts. It is therefore desirable that not all nodes in G_{i-1} are candidates for being linked to S_i . To model this *context* notion, we use a heuristic based on the level-of-detail tree T_{i-1} .

For every node in the tree, its children represent some elaboration, sub-components, supporting arguments or similar. The path from the root node to the most recently added node (i.e. S_{i-1}) is considered to be the current thread of argument in the text. S_i is expected to be related to the current thread or subcomponents of the arguments in the thread. Let P_1, \dots, P_k denote the current thread of argument where P_1 is S_0 and P_k is S_{i-1} . Then, the candidate nodes that are considered for linking S_i to are now restricted to P_1, \dots, P_k and all sibling nodes of P_1, \dots, P_k .

All the nodes with corresponding relations that the graph rule-base suggests to link S_i to would be added to G_{i-1} resulting in G_i . In case the graph rule-base does not suggest anything, the default rule is to link S_i to its previous sentence S_{i-1} with the *UNKNOWN* relation.

Consider the graph and tree in figure 1 and 2. The construction of both when adding sentence 6 (node S_6) to G_5 and T_5 respectively proceeds as follows: The current thread of argument is the path S_0, S_1, S_5 . As all siblings of the thread nodes are also candidates for linking S_6 to, the candidates are S_0, S_1, S_2 , and S_5 . The graph rule-base suggests to link S_6 to S_5 and S_2 with the relations *MOTIVATION* and *LIMITATION* respectively.

2.3.2 How to build the level-of-detail tree

Among those nodes that are connected to S_i in G_i , only those in the last path P_1, \dots, P_k are considered to be selected for constructing T_i . This is based on the assumption that the author would elaborate on one idea completely before going to the next idea. Though this is a simplifying assumption, a more general structure is represented in the discourse structure graph. This assumption is a heuristic working hypothesis that is not critical to our overall approach as it can be altered without the need to abandon our overall approach. Given a node in the path P_1, \dots, P_k connected to S_i in G_i , the tree rule-base is used to decide whether S_i is to be linked to that node in T_i .

Because T_i is a tree, S_i can only be connected to one node in T_{i-1} . When the tree rule-base does not suggest any nodes to link S_i to, the default rule would link S_i to the candidate sentence with the highest index in that last path. Otherwise, S_i would be connected to the sentence with highest index in the list of nodes suggested by the tree rule-base.

Revisiting the example in Figure 2, we want to construct T_6 from G_6 in Figure 1 and T_5 by selecting the node to which S_6 should be linked to. It is trivial if there is only one edge connecting S_6 in G_6 . However in this case we have two edges to consider, i.e. (S_6, S_5) and (S_6, S_2) . The tree rule-base is used here to decide which edge should be used, and it suggests the edge (S_6, S_5) with the MOTIVATION relation.

3. RELATED WORK

3.1 Knowledge Acquisition with Ripple Down Rules

Ripple Down Rules (RDR) is an unorthodox approach to knowledge acquisition upon which we built KAFDIS. RDR does not follow the traditional approach to knowledge based systems (KBS) where a knowledge engineer together with a domain expert performs a thorough domain analysis in order to come up with a knowledge base. Instead a KBS is built with RDR incrementally, while the system is already in use. No knowledge engineer is required as the domain expert repairs the KBS as soon as an unsatisfactory system response is encountered. The expert is merely required to provide an explanation for why in the given case, e.g. the classification should be different from the system's classification. Say, the system's classification was produced by some rule R_A . The explanation would refer to attributes of the case, such as patient data in the medical domain. The new rule R_B will only be applied to cases for which the provided conditions in R_B are true and for which rule R_A would produce the classification, if rule R_B had not been entered. I.e. in order for R_B to be applied to a case as an exception rule to R_A , rule R_A has to be satisfied as well. A sequence of nested exception rules of any depth may occur. Whenever a new exception rule is added, a difference to the previous rule has to be identified by the expert. This is a natural activity for the expert when justifying his/her decision to colleagues or apprentices. A number of RDR-based systems store the case which triggered the addition of an exception rule along with the new rule. This case, being called the *cornerstone case* of the new rule R , is retrieved when an exception to R needs to be en-

tered. The *cornerstone case* is intended to assist the expert in coming up with a justification, since a valid justification must point at differences between the *cornerstone case* and the case at hand for which R does not perform satisfactorily.

This approach resulted in the expert system PEIRS used for interpreting chemical pathology results [3]. PEIRS appears to have been the most comprehensive medical expert system yet in routine use, but all the rules were added by a pathology expert without programming or knowledge engineering support or skill whilst the system was in routine use. The basic idea has been extended into a number of directions - none of it addressing the specific problems present in NLP applications. Ripple-Down Rules and some further developments are now successfully exploited commercially by a number of companies.

Single Classification Ripple Down Rules (SCRDR)

A single classification ripple down rule (SCRDR) tree is a finite binary tree with two distinct types of edges. These edges are typically called *except* and *if not* edges. See Figure 4. They are used for evaluating cases and will be discussed later. Associated with each node in a tree is a *rule*. A rule has the form: *if α then β* where α is called the *condition* and β the *conclusion*.

Cases in SCRDR are evaluated by passing a case to the root of the tree. At any node in the tree, if the example entails the condition of the node, the node is said to *fire*. If a node fires, the example is passed to the next node following the *except* branch. Otherwise, the case is passed down the *if not* branch. This determines a path through a SCRDR tree for an example. The conclusion given by this process is the conclusion from the last node which fired on this path. To ensure that a conclusion is always given, the root node typically contains a trivial condition which is always satisfied. This node is called the *default* node.

A new node is added to an SCRDR tree when the evaluation process returns the wrong conclusion. The new node is attached to the last node evaluated in the tree. If the node has no exception link, the new node is attached using an exception link, otherwise an *if not* link is used. The case causing the new node being added (call this example e) is associated with the new node and is called the *cornerstone case* for that node. To determine the rule for the new node, the expert formulates a rule which is satisfied by e but not by the *cornerstone case* for the last node which fired in the evaluation path.

While the process of incrementally developing knowledge bases will eventually lead to a reasonably accurate knowledge base, provided the domain does not drift and the experts are making the correct judgements, the time it takes to develop a good knowledge base depends heavily on the appropriateness of the used language in which conditions can be expressed by the expert. For example, if the target function to be represented is a linear threshold function in the numerical input space and the conditions an expert can use allow only axis-parallel cuts, it will take very long until most

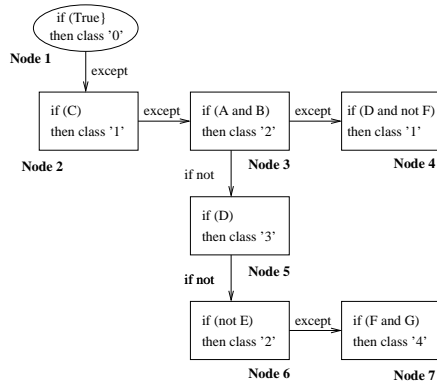


Figure 4: An example SCRDR tree. Node 1 is the default node. A case for which only A and C are true, is classified as '1'. If only 'A' is true, it is classified as '0'. If only A, B, and C are true, it is classified as '2'. If only 'A, C, and D are true, it is classified as '3', etc.

of the relevant objects will be classified correctly.

3.2 Automatically recognizing discourse relations

[12] proposes an approach to automatically recognize relations between two consecutive portions of text using word pairs appearing in those portions. In our approach, not only do we look at words in those texts but also at the structure of the whole document and other pieces of texts related to the ones at hand. Furthermore, we recognize relations between non-adjacent sentences which arguably has the potential to give a substantially more comprehensive discourse structure of the document.

3.3 Summarization approaches

There have been several summarization approaches including [7, 4] (see [8] for a recent survey). The works below are particularly relevant to our approach in the sense that they all seek to represent certain aspects of the text in a graph or tree structure in order to generate a summary.

[9] presented an approach to topic-focused multi-document summarization, which is, however, based on individual terms to encode the topic. This contrasts our objective, where we seek to identify various aspects of, e.g., a technical term, such as justification, motivation, etc. In [9] each document is represented as a graph of relationships among terms in different positions. The relationships include proximity, co-reference, synonymy and hypernymy. A spreading activation algorithm is then used starting with the query terms to find salient terms in documents.

[13] presented a graph-based cohesion work where nodes were paragraphs and edges were labelled with the similarity scores between paragraphs. Paragraphs that were connected to many other paragraphs with a similarity score above a particular threshold were considered salient as they would likely to contain topics discussed in many other paragraphs.

[11] presented work on rhetorical structure theory (RST) based summarization. He proposed an approach using cue phrases to build a RST tree. The RST tree can then be used to generate general purpose sentence-extraction summary. The latter two approaches seek to generate general purpose summaries.

4. BUILDING RULE-BASES

The graph rule-base and the tree rule-base are used in constructing the discourse structure graph and the level-of-detail tree respectively.

Both rule-bases are Single Classification Ripple Down Rules (SCRDR). Rules are composed of a condition part and a conclusion part. Both, graph rules and tree rules take one sentence S , called the *target*, in the graph G_{i-1} and the current sentence S_i and returns the rule's conclusion if the rule's condition is satisfied.

The graph rule's conclusion is a relation between the target sentence and the current sentence and a correspondingly labelled edge is inserted into the graph. The relation is *null* if no edge is to be inserted. The tree rule's conclusion is a boolean value. It is *true* if the target sentence and the current sentence are connected in both the graph and the level-of-detail tree. Below is an example of a graph rule:

Condition:

targetChild.contains *Firstly, algorithm*
sentence.contains *Secondly, algorithm*

Conclusion:

DESCRIPTION

This rule says that the target sentence and the current sentence should be linked with *DESCRIPTION* relation if a child sentence of the target sentence contains *Firstly, algorithm* and the current sentence contains *Secondly, algorithm*. The child-parent relationship comes from the level-of-detail tree.

4.1 Condition

A condition in our rule-bases is a conjunction of boolean attributes. An attribute has an identifier part and a content part. The identifier part determines what sentence the attribute is applied to. Possible identifiers are the sentence S_i , the *target* sentence and sentences surrounding the target sentence in the level-of-detail tree such as *targetChild*, *targetParent* and *targetSibling*.

The content of an attribute can either be a pattern or a relation to the sentence S_i . A pattern consists of a sequence of an arbitrary number of words. Between two words there may be a gap. A gap represents effectively a wild-card which is matched by any sequence of words.

When the content is a pattern, the attribute is true if the pattern matches the identifier sentence. When the content is a relation, the attribute is true if the identifier sentence is linked to the sentence S_i with that relation.

Part of speech phrase and parametrized patterns

To increase the expressiveness of the rule patterns, we introduce part of speech (POS) phrases and parameters.

Every sentence is tagged automatically using the Brill tagger [1]. The output tags are from the Penn Treebank project, but we just use a subset of it, namely: NN (noun), VB (verb), RB (adverb), JJ (adjective), CD (cardinal number) and DT (determiner). Noun, verb and adjective are broad categories including all sub-categories. For example, in the Penn Treebank project, there are four different groups of nouns including singular, plural and proper nouns. We group them under one NN tag.

A word in the pattern can be tagged with a part of speech by users. Initially, the POS tagged by Brill's POS tagger is suggested to the users. Possible POS for a word in a pattern are NN, VB, JJ, RB and CD. POS in a pattern has a different meaning. If a word in the pattern is not tagged with a POS, it will only match that word. With a POS tagged, the word matches a phrase of that POS according to the following rules:

- Adv phrase \rightarrow (RB)+
- Adj phrase \rightarrow [Adv phrase | ε] (JJ)+
- Verb phrase \rightarrow [Adv phrase | ε] VB
- Noun phrase \rightarrow [CD | DT | ε] [Adj phrase | ε] (Noun)+
- Cardinal phrase \rightarrow CD

Except for Cardinal phrase, a word tagged with POS P matches all P phrases ending with that word. A word tagged with CD matches everything in the sentence tagged CD by Brill's part of speech tagger. For example, in the pattern

We introduce algorithm/NN

algorithm is tagged as a Noun phrase. The pattern therefore would match these sentences:

We introduce/VB a/DT new/JJ algorithm/NN.....

We introduce/VB an/DT algorithm/NN....

We introduce/VB the/DT Apriori/NN algorithm/NN....

Because one condition can have patterns applied to multiple sentences, it is sometimes desirable to be able to put constraints on words appearing in different patterns of the condition. Parametrized patterns are allowed in our conditions to enable this. If we want multiple patterns to share a sequence of words, we can define that sequence of words as a parameter. For example, if we have a condition

target.contains *We introduce X1 in our paper.*

sentence.contains *This X1 must*

then *X1* can be matched by any non-empty phrase.

Parameters can further be constrained by tagging it with a POS. This will in turn make sure that the parameter only matches phrases of that POS. For instance, if the condition has parameter *X1* tagged with NN, this parameter will only match noun phrases.

4.2 Using KAFDIS for building the graph and tree rule-bases

When using KAFDIS the user is presented with the text. At each step i , whether it is to expand the discourse structure graph G_{i-1} or the level-of-detail tree T_{i-1} , KAFDIS uses the corresponding rule-base to automatically suggest which nodes should be connected to S_i and in case of building the discourse structure graph how the edge should be labelled. The user then verifies the displayed result. If a target node or the relation between the current sentence and the target was incorrectly suggested, the user needs to provide a condition that discriminates between the current case and the corner stone case of the fired rule, i.e. the case for which the fired rule was originally entered. A case consists of the sentence S_i together with the graph and tree at the time of processing the sentence, the target sentence in the graph or tree, and the conclusion. To support the user, both cases as well as the fired rule are displayed and the phrases that match the rule are highlighted. With this support it is quite easy to come up with a new exception rule that differentiates the two.

KAFDIS then checks the new rule for consistency against the existing set of previously classified cases. There is a conflict if the new rule misclassifies a case that has been correctly classified before. In the conflict case, the new rule is rejected and the user has to construct a different condition possibly by modifying the previous one. The conflicting case would be presented to the user to help revising the condition. The user has to go through this process until the new rule is accepted into the rule-base. Finally, the new case is added into the list of previously seen cases for future consistency checks.

This process results in a possibly complex tree of rules and their exception rules modelling the domain. The acquisition of new rules does not become measurably more difficult or time consuming as the tree grows.

4.3 Example

Let's look at the italic example text in section 5 (Figure 6 shows the level-of-detail tree and the discourse structure tree) to see how the expert constructs the two rule-bases to build the graph and tree representations at various steps.

Initially, the graph rule-base (GRB) has only one default rule that assigns the *null* relation to the pair of *target* and *sentence*. The tree rule-base (TRB) has only one default rule with the *false* conclusion. At step 1, our discourse structure graph G_0 and level-of-details tree T_0 contain only S_0 . GRB recommends the *null* relation between *sentence* S_1 and *target* S_0 . The expert corrects GRB by adding to it a rule:

Condition

target.contains *X1/NN*

sentence.contains *X1/NN (gap) is given in*

Conclusion

DESCRIPTION

At step 4, both G_3, T_3 contains S_0, S_1, S_2, S_3 with relations shown in figure 6. We now need to add *sentence* S_4 to

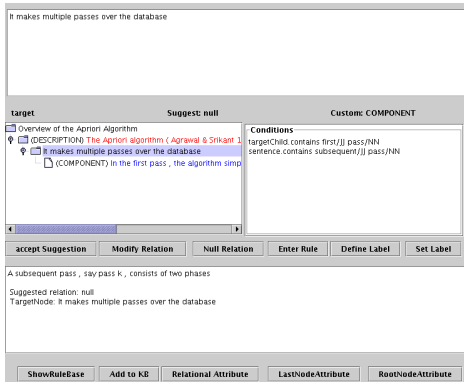


Figure 5: A snapshot of the rule formation process showing the graph structure.

G_3 . According to T_3 , possible *target* node to add S_4 are S_0, S_1, S_2, S_3 . With the current GRB, it would suggest the *null* relation for all four pairs. Suppose we want to say that S_4 has the COMPONENT relation to S_2 and the SEQUENCE relation to S_3 , we would add the following two rules:

Rule to recognize relation between S_4 and S_2
Condition
 targetChild.contains first/JJ pass/NN
 sentence.contains subsequent/JJ pass/NN
Conclusion
 COMPONENT

Rule to recognize relation between S_4 and S_3
Condition
 target.contains first/JJ pass/NN
 sentence.contains subsequent/JJ pass/NN
Conclusion
 SEQUENCE

Now to get T_4 , we need to decide which edge, among (S_4, S_2) and (S_4, S_3) , to add to T_3 . Suppose we want to choose (S_4, S_2) , the following rule could be added to TRB:

Condition
 target.hasRelation COMPONENT
Conclusion
 true

Note that in step 13 when looking at *sentence* S_{13} , sentences $S_6 \dots S_{11}$ are not considered as possible *targets* to link S_{13} to as they are not in the current thread of argument or the siblings of the nodes in the current thread of argument.

5. EXPERIMENTAL RESULTS

We built a graph rule-base and a tree rule-base by looking at papers of Agrawal in the data mining area. In this experiment, we only looked at papers of one author as we hypothesized that a single author will only have a limited set of phrases to express themselves. Hence we would expect a faster convergence of the knowledge acquisition process

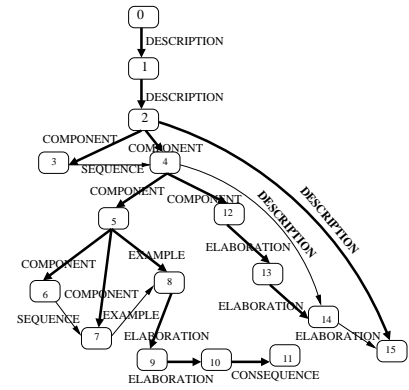


Figure 6: A discourse structure graph and a level-of-detail tree. The level-of-detail tree is shown in boldface. Node i represents i^{th} sentence in the text.

than otherwise could be expected. In total we acquired 116 rules which were capable of correctly annotating the papers.

The discourse structure graph and level-of-detail tree shown in Figure 6 are automatically generated by the rule-bases we developed. They represent a structure of the paragraph shown below. This paragraph is taken from a paper¹ of the same author Agrawal:

⁰Overview of the Apriori Algorithm. ¹The Apriori algorithm (Agrawal & Srikant 1994) used in Quest for finding all frequent itemsets is given in Figure 1. ²It makes multiple passes over the database. ³In the first pass, the algorithm simply counts item occurrences to determine the frequent 1-itemsets (itemsets with 1 item). ⁴A subsequent pass, say pass k , consists of two phases. ⁵First, the frequent itemsets L_{k-1} (the set of all frequent $(k-1)$ -itemsets) found in the $(k-1)$ th pass are used to generate the candidate itemsets C_k , using the apriori-gen() function. ⁶This function first joins L_{k-1} with L_{k-1} , the joining condition being that the lexicographically ordered first $k-2$ items are the same. ⁷Next, it deletes all those itemsets from the join result that have some $(k-1)$ -subset that is not in L_{k-1} , yielding C_k . ⁸For example, let L_3 be $\{\{1\ 2\ 3\}, \{1\ 2\ 4\}, \{1\ 3\ 4\}, \{1\ 3\ 5\}, \{2\ 3\ 4\}\}$. ⁹After the join step, C_4 will be $\{\{1\ 2\ 3\ 4\}, \{1\ 3\ 4\ 5\}\}$. ¹⁰The prune step will delete the itemset $\{1\ 3\ 4\ 5\}$ because the itemset $\{1\ 4\ 5\}$ is not in L_3 . ¹¹We will then be left with only $\{1\ 2\ 3\ 4\}$ in C_4 . ¹²The algorithm now scans the database. ¹³For each transaction, it determines which of the candidates in C_k are contained in the transaction using a hash-tree data structure and increments the count of those candidates. ¹⁴At the end of the pass, C_k is examined to determine which of the candidates are frequent, yielding L_k . ¹⁵The algorithm terminates when L_k becomes empty.

The level-of-detail tree in Figure 6 can be used to assist users in understanding the algorithm more quickly. For example, sentence 4 says pass k has two phases and they are represented as two children nodes of node 4. If the users are only interested in the second phase, they can jump straight to sentence 12 without reading a long description of the first phase. Another application that appears useful is to colour-code the different phases of the algorithm so that the reader can quickly jump back and forth between the different phases

¹Developing Tightly-Coupled Data Mining Applications on a Relational Database System. In Proc. of the 2nd Int'l Conference on Knowledge Discovery in Databases and Data Mining, 1996

until they understood the algorithm or are otherwise satisfied.

Of course, our experiments focused on a single author and do not immediately imply that our approach would also work for other papers in the domain. However, while different authors have different ways of expressing themselves, we believe that there are only a certain number of typical wordings being used within a community, so that a knowledge base built for a larger cross section of authors will reasonably generalize to other authors in the area.

6. CONCLUSION

We presented a new approach towards automatic text summarization. This included the generation of a level-of-detail tree and a discourse structure graph.

Furthermore, we developed an incremental knowledge acquisition framework, KAFDIS, allowing us to economically tailor the knowledge bases which govern the construction of the level-of-detail tree and the discourse structure graph of a document to suit new topics and/or new domains.

Finally, we presented promising initial experimental results with KAFDIS. Our experiments were limited to the analysis of papers of the same author resulting in rule-bases totalling in 116 rules. The acquisition of the rules was not difficult and reasonably quick (less than 5 minutes per rule). Most of the expert time was spent on deciding how to link a new sentence with the existing nodes in the graph. Once that decision had been taken, the formulation of a suitable rule was not difficult.

In fact, building our rule-bases does not take much more time compared to other supervised machine learning approaches which need training data that would require manual classification. We arguably require less examples by asking experts to formulate rules. In building training data, the expert still has to go through the same process except for the rule formulation part. Building a training corpus with a complex structure like our graph and tree is a time consuming process. Annotating an RST corpus [2] is arguably simpler than ours as they only consider relations between 2 adjacent chunks of text. The resulting RST corpus contains 385 WSJ articles (averaged 458 words per document) and took over one year of more than a dozen experts on a full or part time basis [2]. Of course, we can not compare ours with their quality (e.g. experts' annotation agreement) but we do have the flexibility of moving to a new domain with a new set of relations quickly.

In future research we will implement an interactive document viewer as sketched. In order to make it really useful we will develop more extensive knowledge bases for the automatic processing of scientific papers. Those knowledge bases will also be useful as a stand-alone application to generate topic-based summaries of scientific and technical documents.

The interactive document viewing seems to have enormous potential to substantially accelerate the digestion of large

documents by limiting the portion of text actually being read to a small fraction of the entire document and assisting the reader in selecting those passages.

7. REFERENCES

- [1] E. Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565, 1995.
- [2] L. Carlson, D. Marcu, and M. E. Okurowski. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of the 2nd SIGDIAL Workshop on Discourse and Dialogue, Eurospeech*, Denmark, 2001.
- [3] G. Edwards, P. Compton, R. Malor, A. Srinivasan, and L. Lazarus. PEIRS: a pathologist maintained expert system for the interpretation of chemical pathology reports. *Pathology*, 25:27–34, 1993.
- [4] N. Elhadad and K. McKeown. Towards generating patient specific summaries of medical articles. In *NAACL'01 Automatic Summarization Workshop*, 2001.
- [5] B. J. Grosz and C. L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–203, 1986.
- [6] J. R. Hobbs, editor. *Literature and Cognition*. CSLI Lecture Notes Number 21, Mento Park, CA, 1990.
- [7] E. Hovy and C. Lin. Automated text summarization and the summarist system. In *Proceedings of the TIPSTER Text Program, Phase III.*, pages 197–214, 2000.
- [8] I. Mani. *Automatic Summarization*. John Benjamins, 2001.
- [9] I. Mani and E. Bloedorn. Summarizing similarities and differences among related documents. *Information Retrieval*, 1(1):35–67, 1999.
- [10] W. C. Mann and S. A. Thompson. Rhetorical structure theory: Towards a functional theory of text organization. *Text*, 8(3):243–281, 1988.
- [11] D. Marcu. *The Rhetorical Parsing, Summarization and Generation of Natural Language Texts*. PhD thesis, Toronto, Canada, 1997.
- [12] D. Marcu and A. Echihiabi. An unsupervised approach to recognizing discourse relations. In *Proceedings of the 40th Annual meeting of the Association for Computational Linguistics*, Philadelphia, PA, 2002.
- [13] G. Salton, S. A., M. Mitra, and C. Buckley. Automatic text structuring and summarization. *Information Processing and Management*, 33(2):193–207, 1997.