Proceedings of the Twenty-Second International FLAIRS Conference (2009)

SlidesGen: Automatic Generation of Presentation Slides for a Technical Paper Using Summarization

M. Sravanthi, C. Ravindranath Chowdary and **P. Sreenivasa Kumar**
Department of Computer Science and Engineering
Indian Institute of Technology Madras
Chennai, India 600 036.
{sravanti,chowdary and psk}@cse.iitm.ac.in

Abstract

Presentations are one of the most common and effective ways of communicating the overview of a work to the audience. Given a technical paper, automatic generation of presentation slides reduces the effort of the presenter and helps in creating a structured summary of the paper. In this paper, we propose the framework of a novel system that does this task. Any paper that has an abstract and whose sections can be categorized under introduction, related work, model, experiments and conclusions can be given as input. As documents in LaTeX are rich in structural and semantic information we used them as input to our system. These documents are initially converted to XML format. This XML file is parsed and information in it is extracted. A query specific extractive summarizer has been used to generate slides. All graphical elements from the paper are made well use of by placing them at appropriate locations in the slides. These slides are presented in the document order.

Introduction

Slides have been an effective and popular means of presentation of information. In many conferences and meetings, a presenter takes the aid of slides to present his work in a systematic way (pictorial). In recent years with the availability of many software tools like Microsoft PowerPoint, Openoffice Presenter etc., for easy preparation of slides, their usage has increased tremendously. But these tools help only in the formatting of content (stylizing, bullet points etc), but not in preparing the content itself. A user has to start from scratch and it is a time consuming task. In this work, we propose a tool that generates slides for the presentation with important points and all necessary figures, tables and graphs from a technical paper. As it is evident, such kind of a tool saves time and reduces the effort by providing a basic presentation, which can be further tuned/upgraded as final presentation.

Slides contain the *summarized* version of a technical report. They contain the vital points of the report arranged in a systematic way, including graphic elements like figures and tables for easy illustration of the idea. Given a document, "Automatic generation of presentation slides" becomes a nontrivial task because of challenges like segmentation of document into multiple topics, summarizing content of each topic and aligning these topics to one or more slides and placing necessary graphical content like figures, graphs and tables in appropriate slides at appropriate locations.

In this paper we concentrate on generating slides for research papers that are in accordance with standards of conference/journal proceedings. By and large, conference papers have an almost similar structure. They have an abstract and the sections present in them can be broadly classified into presenting the introduction, the related work, actual work (model), the experiments, the conclusions and the bibliography. Most of the times, the presenter preserves the order of the paper in slides and each section is allotted one or more slides. A slide has a title and contains some bulleted points which are important in that section. Observing the similarity present between conference paper and human written slides for the paper, we address the problem of automatic generation of presentation slides by exploiting the structure of a conference paper. Here after we use terms "conference paper", "technical paper", "document" and "report" interchangeably.

Related Work

Very few papers directly address this problem of automatic slide generation. Masao et al.(Masao & Koiti 1999) tried to automatically generate a presentation from semantically annotated documents. The input documents are normal text documents which are semi-automatically annotated with GDA tagset[1] to infer semantic relations between sentences. The semantic relations include marking noun phrases and verb phrases etc., grammatical relations like subject, verb etc., thematic roles like agent, patient, recipient etc., or rhetorical relations like cause, elaboration etc. Initially topics of the document are identified and ranked. Highly ranked topics are taken as slide headings. Relevant sentences to the topics are extracted from documents and are processed to prepare bullet points.

Shibata et al. in (Shibata & Kurohashi 2005) prepares slides from raw text. Clauses and sentences are treated as discourse units and several relations like contrast, list, additive, topic-chaining, elaboration, cause, example etc., are identified between them. Some clauses are identified as

[1]http://i-content.org/GDA/tagset.html

topic parts and others are treated as non-topic parts. These extracted topic and non-topic parts are placed on the slides based on the discourse structure detected. Some heuristic rules have been proposed for generating slides from these topic and non-topic parts. In (Yoshiaki, Masashi, & Katsumi 2003; Miyamoto, Sakai, & Masuyama 2006), the authors tried to analyze LaTeX structure of technical document to generate a set of slides.

Masum et al.(Masum, Ishizuka, & Islam 2005; Masum & Ishizuka 2006) proposed a system called Auto Report to Presentation (ARP) which prepares a text report with text relevant to the users search query first and then a presentation from the report. The query is disambiguated using Wikipedia and a report for each sense of the query is prepared. The query along with its senses is given to multiple search engines and relevant web pages are retrieved from the internet. Headings and text-chunks from the retrieved web pages are extracted and the text-chunks are summarized. A report is initialized to the headings and text chunks present in the first link. Then contents of other documents are added to this report if they are not similar to previously entered contents. A presentation is built form this report by taking a maximum of 5 lines from each head-text tuple and slides for them are prepared. These 5 lines are randomly picked as two lines from the top, one in the middle and two from the end of text-chunk.

Though the approaches(Masao & Koiti 1999; Shibata & Kurohashi 2005) presented above have been domain and style independent, they are not language independent as they require language specific parsers to identify the specified relations. Moreover the technical documents are different from normal articles which are used as input in (Masum, Ishizuka, & Islam 2005; Masum & Ishizuka 2006; Masao & Koiti 1999; Shibata & Kurohashi 2005). In (Masum, Ishizuka, & Islam 2005; Masum & Ishizuka 2006) sentences are selected randomly for presentation from the report, which may result in a incoherent text. In contrast to above approaches, in this work we make use of the structure given by the author in the LaTeX document and proceed in a language and domain independent manner. We employ a query specific text summarization system that generates a coherent set of points by summarizing the text content under a section/sub-section.

System Overview

Our system framework is as shown in Figure 1. Details of each component are discussed in the following sections.

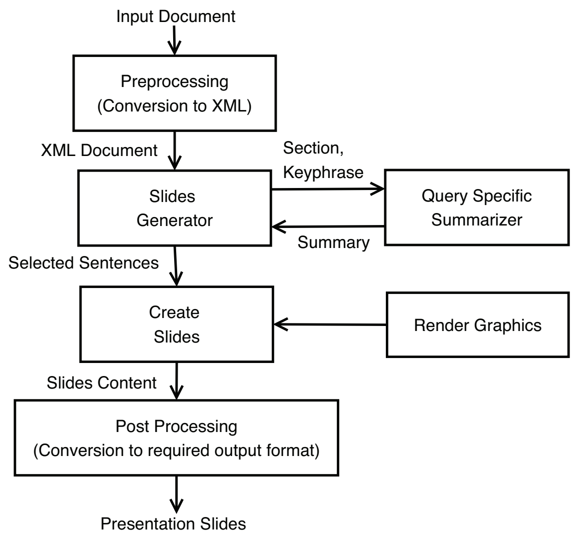

Figure 1: System Architecture

Pre-processing

Any technical document is normally divided into sections, sub-sections, paragraphs etc. For a better structural summary we need to preserve as much structural information as possible from the input document. There are many markup languages like HTML, XML, etc., which can encode the structural information of the document. LaTeX is one such document markup language that has been widely accepted by the academic community to write a technical paper because of the quality of typesetting that can be achieved. It has a rich set of features (tags) to encode the information related to the document structure and graphical elements present in the document. So, we chose documents written in LaTeX format as input. These documents are converted to XML using a publicly available LaTeX to XML converter called LaTeXML[2]. Appropriate converters can be used to convert and extract the required information from any document present in other formats like PDF, doc etc. This XML file is used for further processing.

Configuration File Generation

Generally a technical paper contains sections that can be broadly classified into introduction, related work, actual model, experiments and conclusions. As each section has different point of view and different information content and writing style, they are treated differently in our system for generation of slides. Initially as a first step, the sections are categorized and then key phrases are extracted for some sections of the paper. These details are saved in a file called “Configuration File”. Figure 2 presents the DTD of the configuration file. The rules used for identifying them are as follows.

If a section is one of the initial sections and title contains words like “Introduction”, “Motivation”, “Background”, “Problem Statement” etc., it is placed under “Introduction” category. A section containing considerably large number of <cite> </cite> tags or it’s title contains words like “Related Work”, “Literature Survey” etc. are categorized under “Related Work” category. The sections that present the actual idea of the authors which solves the problem at hand falls under model category. Sections not placed under any of the remaining categories are placed under “Model” category. If a section name contains words like “Experiments”, “Evaluations” etc., they fall under “Experiments” category. If a section has a title containing words like “Conclusions”,

[2]http://dlmf.nist.gov/LaTeXML/

```
<?xml encoding="UTF-8?>
<!DOCTYPE Configuration [
<!ELEMENT Configuration (Introduction, RelatedWork,
                         Model, Experiments, Conclusions)>
<!ELEMENT Introduction (Sections)>
<!ELEMENT RelatedWork (Sections)>
<!ELEMENT Model (Sections, Keyphrases)>
<!ELEMENT Experiments (Sections, Keyphrases)>
<!ELEMENT Conclusions (Sections, Keyphrases)>
<!ELEMENT Sections (SectionNo*)
<!ELEMENT Keyphrases (Keyphrase*)
<!ELEMENT SectionNo (#PCDATA)>
<!ELEMENT Keyphrase (#PCDATA) >]>
```

Figure 2: DTD of Configuration File

"Future Work" etc., it is placed under "Conclusions" category.

Extracting Key Phrases Generally most of the research papers have associated key phrases. They help in categorizing the content of the paper and contain important concepts introduced in the paper. Mostly these key phrases are related to the model and experimental sections. As these sections are long and un-predictable in nature, we use these key phrases in summarizing these sections. The extracted summary contains important information relevant to the key phrases. As key phrases indicate important information, the sentences in the extracted summary are also important and are put in slides.

The keywords at the beginning of the paper are added to the model and experimental categories in the configuration file. In addition to the keywords given in the paper, the titles in sections present under model and experiments categories are added as key phrases to the corresponding categories. For conclusions, the title of the paper and a set of phrases like "concludes", "present", "results", "prove" etc., as detailed in Section are used as key phrases. A user can also edit or provide this generated configuration file to customize the presentation as per his/her needs.

QueSTS Summarizer

In this section we present a brief overview of our query specific summarizer that has been used to summarize model, experiment and conclusion sections. In QueSTS we represent the input text in the form of an integrated graph (IG) where a sentence is made as a node, and an edge exists between two nodes if the sentences in them are similar. An edge weight is calculated as cosine similarity between connecting sentences. An edge having a weight above a $minSimilarity$ threshold is retained in the graph. Also edges between adjacent sentences in the document are retained irrespective of their edge weight. As this is a single document summary, it is assumed that there will not be any redundant sentences.

When a key phrase is given as a query to the system, a centrality based query specific node weight is calculated for each node as per the following equation(Otterbacher, Erkan, & Radev 2005).

$$w_q(s) = d\frac{sim(s,q)}{\sum_{m\in N} sim(m,q)} + (1-d)\sum_{v\in adj(s)} \frac{sim(s,v)}{\sum_{u\in adj(v)} sim(u,v)} w_q(v)$$

where $w_q(s)$ is node weight of node s with respect to query term q, d is bias factor, N is the total number of nodes and $sim(s_i, s_j)$ is cosine similarity between sentences s_i and s_j. Stop words are removed and remaining words are stemmed before computing similarity. First part of the equation computes relevancy of node to the query and second part considers neighbours' node weights. The bias factor d gives trade-off between these two parts.

The key phrase is initially tokenized, and for each token a node weight is calculated. Thus a node has as many node weight values as the number of tokens (other than stop words) in the key phrase. As node weight is computed using a centrality based approach, a high weight indicates a highly relevant sentence in the presence of highly relevant neighbours. Though a node doesn't has a query term, if it is connected to a node containing query relevant information it is relevant and gets a non zero node weight. Thus we have query independent edge weights and query specific node weights.

A summary is said to be good if it has good coherence and information. Coherence is the property that determines the readability of the text. Hence we make use of the edge weights to preserve coherence and the node weights to preserve query relevant information. For each query term q, from each node r a Contextual Tree($CTree_{rq}$) is constructed as follows. The neighbourhood of root node r is explored in BFS order, and at most $b(=3)$ nodes having higher h value calculated as ($h = \alpha w(e_{ij}) + \beta w_q(j)$), are selected and BFS traversal is continued from these selected nodes. The exploration from selected prominent nodes (at most b) is continued until a level which has a node containing a query term ($anchor\ node$) or maximum depth d is reached.

All nodes along the path from root to anchor node, along with their siblings are added to the $CTree_{rq}$. Siblings contain supporting information that is required to establish the context. When query term is not found until maximum depth d then $CTree_{rq}$ for that query term remains empty. If a root node has the query term then root and its adjacent "b" nodes are added to $CTree_{rq}$ and no further exploration is done. Once all CTrees rooted at a node r are constructed for each query term, they are merged (unioned) to form a summary graph ($SGraph_r$). All nodes and edges present in all CTrees are present in the $SGraph_r$. These $SGraphs$, generated one from each node in the integrated graph are ranked using a scoring model and the one with the highest rank is returned as summary. A SGraph with high node and edge weights gets high rank as per the scoring model.

For example, if we have an integrated graph as shown in Figure 3, for finding relevant nodes for query term $q1$, the neighbourhood of node h and then nodes g, d are explored.

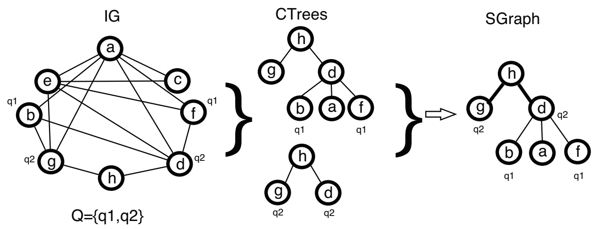

Figure 3: Generation of CTrees and SGraphs from node h

Exploration is stopped on reaching nodes b and f which contain $q1$. But b is found closer to d than g, so path from b to h through d, along with siblings a, f and g are added to $CTree_{hq1}$. Similarly for the query term $q2$, $CTree$ is generated. $SGraph$ is formed by merging these two trees. The edges shown in dark and nodes connected by them are common in both $CTrees$ and their weights are counted twice as per scoring model. All query terms are covered as $CTrees$ of all query terms are merged to form a $SGraph$. As the $SGraph$ is a connected component, we will have inter connected set of sentences in the summary and hence coherence is preserved. QueSTS summarizer is discussed in more detail in (Sravanthi, Chowdary, & Kumar 2008).

Slides Generation

The XML file and the configuration file are given as input to Slides Generator component and it generates slides for each section and sub-section.

Generating Introduction Slides Introduction contains background of the problem at hand, motivation and a brief overview of the contributions. Abstract also summarizes the complete work presented in the paper. So, each sentence of the introduction section is compared with complete abstract and sentences with high similarity are placed on the slides. We used cosine similarity measure to compute this similarity. For each sub-section in introduction section this procedure is repeated and top m sentences are extracted. For each section and sub-section, slides are created with their title as the slide's title. By default at most 2 slides with 4 sentences each are created for each section and sub-section of introduction category.

Generating Related Work Slides Related work presents the work done by the other researchers relating to the problem addressed in the paper. It contains citations to other related works. In LaTeX these citations are embedded using \cite{keylist}. In XML file it is encoded in <cite> </cite> tags. All sentences containing these cite tags are retrieved from related work sections. Similarity of these sentences with the text under introduction section is calculated and highly similar sentences are placed in slides.

Generating Model and Experiment Slides Model and Experiment sections are the important and long sections in the paper. The keywords extracted as specified in Section Configuration File Generation are used to identify and extract important sentences from these sections. We use the query specific summarizer QueSTS(Sravanthi, Chowdary, & Kumar 2008) described above for this purpose. The text under the specified section/sub-section and extracted/given key phrases are given as input to the QueSTS summarizer. The key phrases are used as the queries in the summarizer. Sentences of the summaries obtained for each key phrase are combined and are placed in slides in the order in which they appear in the paper.

Algorithm 1 Generating model/experiment slides

```
1:  Input: Section/sub-section $sec$ categorized under
    model/experiments
2:  Output: Slides presenting model/experiments
3:  $title$ = extract section title
4:  $Keyphrases$ = get key phrases for this category from
    config file
5:  $text$ = text under $sec$
6:  $points$ = {}
7:  for each keyphrase $k$ in the $Keyphrases$ do
8:      $points$ = $points \cup$ summarize($text$, $k$)
9:  end for
10: renderGraphics($sec, points$)
11: createSlides($title, points$){//using Algorithm 2}
12: if $sec$ has sub-sections then
13:     for each sub-section $sub\text{-}sec$ in $sec$ do
14:         generateModelOrExperimentSlides($sub\text{-}sec$)
15:     end for
16: end if
```

Generating Conclusion Slide A conclusions section concludes the paper by stating the work contributed and its consequences and implications. Usually it contains words like "contribution", "proposed", "conclude", "system", "model", "argue", "present", "experiments", "better", "results" etc., These keywords along with the title of the paper are used as the queries and the text under this section is summarized and slides are prepared as per Algorithm 1.

Rendering Graphics

Graphics are an important part of the presentation. All special environments used to highlight some portion of the text in LaTeX like definitions, theorems, tables, equations, figures etc., are treated as graphical elements and are extracted from the document. These elements are enclosed in special tags like <theorem>, <figure>, <equation> etc. and are extracted as they are. Normally in a paper figures, tables appear at the place where they are referred to in the text. Equations are present in-line within the text, and are referred at later points in the document. Theorems, definitions and examples are present in-line within the text. These elements which are present in-line within the text are called as "in-line graphical elements". The sentences just before these elements, containing statements like "... is defined as follows", "... is calculated using equation", "For example ..."

etc., are also important and they provide flow in the summary. So, all sentences that either refer a graphical element or are present before the in-line graphical elements are extracted and placed along with the element in the slides.

Aligning Sentences and Creating Slides

Once the important sentences and graphical elements are extracted, the slides can be prepared. For each section and sub-section there is at least one dedicated slide. The order of slides is same as the order of the sections/sub-sections in the paper. Depending upon the number of sentences selected, the number of slides is decided. These slides are assigned same title as their corresponding section/sub-section title. If in a slide there is a sentence that is referring to any graphical element, the corresponding element is displayed next to it. As all referring sentences are present in selected sentences, all graphical elements are captured. All sentences in the slides are ordered as per document order and any duplicate occurrences of elements are avoided. Algorithm 2 presents overview of this procedure.

Algorithm 2 Creation of slides

```
1: Input: Slide title, Selected sentences($points$)
2: Output: Sequence of slides
3: $noOfSentences$ = $|points|$
4: $n$ = maximum number of sentences per slide
5: $frac$ = ceil($noOfSentences/n$)
6: $i$ = 0
7: order sentences in $points$ as per document order
8: while $i < frac$ do
9:     create a new slide
10:    Set title of the slide
11:    for each sentence $s$ in the $points$ do
12:        if remaining sentences in $points < n/2$ then
13:            Add $s$ to previous slide
14:        else
15:            Add $s$ to the current slide
16:        end if
17:        if $s$ refers a graphical element or is before a in-line
           graphical element then
18:            Add the referred element to the slide after $s$
19:        end if
20:    end for
21:    append the slide to the output presentation file
22: end while
```

Experimental Results

We chose to do user evaluation by asking the authors of the papers to rate the presentation generated by our system. We collected source code(.tex file and figures) of eight papers and generated presentations for them. They were given to each one of the authors of the corresponding papers. The authors were asked to answer the following questions by giving a rating on a scale of 1 (bad) to 10 (good) for the presentations.

Q1 : How much information is covered in the presentation?

Q2 : What is the level of coherence in the slides?

Q3 : How much do you think this presentation can be a good starting point to prepare final presentation?

Q4 : How much of the presentation has to be changed to prepare final presentation assuming that no information from outside the paper is to be included in the slides? (Ans: 100%, 80%, 60%, 40%, 20%)

Q5 : What is your overall satisfaction level with the presentation?

Table 1: Evaluation Results of Presentation

User	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**
U1	10	10	9	20%	9
U2	9	10	9	40%	8
U3	10	9	8	20%	8
U4	10	10	8	20%	8.5
U5	8	8	8	40%	8
U6	8	8	9	40%	8
U7	10	9	9	20%	8.5
U8	9	8	8	40%	8
Average	9.25	9	8.5	30%	8.25

On the whole all authors have agreed on the point that the system indeed generates a presentation that can be a good starting point to prepare a final presentation and it can actually be presented only with some minor changes. The results establishing the fact are shown in the Table 1. They were all satisfied with the presentation generated. As per the users ratings the coherence and coverage of presentations have been good. In an informal discussion with the authors it was found that majority of the changes required for the presentation is to compress the sentences.

LaTeX is rich, with many constructs. As of now some environments like itemize, enumerated have not been handled in our system. Paragraph titles have not been used. Presentations usually don't contain algorithms. So we didn't extract content in algorithm and verbatim environments and footnotes have been discarded. The set of slides generated for section "QueSTS Summarizer" of this paper are provided in Appendix. As this section is part of model all section and sub-section titles in Section "System Overview" were considered as key phrases for these slides generation.

Discussion

In total 4 slides were generated for the Section "QueSTS Summarizer". Most of the information from this section is covered in slides along with the necessary equation and figure. But the details of $CTree$ construction are missing. This is because we chose to use only titles of sections and sub-section in this paper as key phrases. The most relevant key phrase to this section is its title "QueSTS Summarization component" and key phrase "summarizing section content". If the section specific keywords like "Integrated graph","CTree", "SGraph" etc., were added to the configuration file, the slides would have been with complete information. If the sections are small like Conclusions section, almost all information appears in the slides.

Conclusions

In this paper we propose a system that can generate a good quality presentation, from a technical paper given in LaTeX format. These slides provide a good starting point for the preparation of final presentation. We make a good use of the richness of the LaTeX document markup language and generate slides using only statistical processing. Future work includes usage of natural language processing techniques to compress the extracted sentences and to identify appropriate indentation structure for them so that presentation becomes much more appealing.

References

Masao, U., and Koiti, H. 1999. Automatic slide presentation from semantically annotated documents. In *Proceedings of the ACL workshop on Coreference and its applications*. Morristown, NJ, USA: Association for Computational Linguistics.

Masum, S. M. A., and Ishizuka, M. 2006. Making topic-specific report and multimodal presentation automatically by mining the web resources. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, 240–246. Washington, DC, USA: IEEE Computer Society.

Masum, S. M. A.; Ishizuka, M.; and Islam, M. T. 2005. Auto-presentation: A multi-agent system for building automatic multi-modal presentation of a topic from world wide web information. In *IAT '05: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, 246–249. Washington, DC, USA: IEEE Computer Society.

Miyamoto, M.; Sakai, H.; and Masuyama, S. 2006. Research on automatic generation of presentation slides from a latex manuscript of a paper (in Japanese). *Journal of Japan Society for Fuzzy Theory and Intelligent Informatics* 18(5):752–760.

Otterbacher, J.; Erkan, G.; and Radev, D. R. 2005. Using random walks for question-focused sentence retrieval. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, 915–922. Morristown, NJ, USA: Association for Computational Linguistics.

Shibata, T., and Kurohashi, S. 2005. Automatic slide generation based on discourse structure analysis. In *IJCNLP*, 754–766.

Sravanthi, M.; Chowdary, C. R.; and Kumar, P. S. 2008. QueSTS: A query specific text summarization approach. In *Proceedings of the 21st International FLAIRS Conference*, 219–224. Florida, USA: AAAI Press.

Yoshiaki, Y.; Masashi, T.; and Katsumi, N. 2003. A support system for making presentation slides (in Japanese). *Transactions of the Japanese Society for Artificial Intelligence* 18:212–22.

Sample slides

Table 2: Slides for Section

QueSTS Summarizer

- In this section we present a brief overview of our query specific summarizer that has been used to summarize model, experiment and conclusion sections.
- In QueSTS we represent the input text in the form of an integrated graph (IG) where a sentence is made as a node, and an edge exists between two nodes if the sentences in them are similar.
- An edge weight is calculated as cosine similarity between connecting sentences.
- Also edges between adjacent sentences in the document are retained irrespective of their edge weight.

QueSTS Summarizer

- When a key phrase is given as a query to the system, a centrality based query specific node weight is calculated for each node as per the following equation[].

$$Equation\ for\ NWeight\ deleted\ for\ space\ constraints$$

- where wqs is node weight of node s with respect to query term q, d is bias factor, N is the total number of nodes and simsisj is cosine similarity between sentences si and sj.
- Thus we have query independent edge weights and query specific node weights.
- Hence we make use of the edge weights to preserve coherence and the node weights to preserve query relevant information.

QueSTS Summarizer

- These SGraphs, generated one from each node in the integrated graph are ranked using a scoring model and the one with the highest rank is returned as summary.
- A SGraph with high node and edge weights gets high rank as per the scoring model.
- For example, if we have an integrated graph as shown in Figure , for finding relevant nodes for query term q1, the neighbourhood of node h and then nodes g, d are explored.

$$Figure\ 3\ deleted\ for\ space\ constraints$$

- Exploration is stopped on reaching nodes b and f which contain q1.

QueSTS Summarizer

- Similarly for the query term q2, CTree is generated.
- As the SGraph is a connected component, we will have inter connected set of sentences in the summary and hence coherence is preserved.
- QueSTS summarizer is discussed in more detail in [].