Generating Slides from HTML pages

Arun JVS[*]
201001079
arun.jvs[†]

Rohit Girdhar[*]
201001047
rohit.girdhar[†]

Sudheer Kumar[*]
201001149
sai.shanka[†]

Abstract

Authors of technical papers often use slides to effectively present their ideas in conferences and meetings. If there is an easy way of generating slides from a research paper, then it would be of great use to them. In this paper, we propose the framework of an automated system which takes the HTML version of a technical paper and generates a starting point presentation with organized headings, images, summarized text and references that can be further turned into a final presentation. HTML version of any paper which is in accordance with conference/journal proceedings can be given as input to out system. The HTML file is parsed, information in it is extracted and is converted into a predefined XML format. Then the data in this XML file is summarized using multiple heuristics to generate most informative content. The summarized content is then converted into a neat LaTeX presentation.

1. Motivation

Presenting information using slides is an effective way for audience retention. Presenters take the aid of slides for presenting research papers in conferences, for explaining them in classrooms and for other academic purposes. But creating slides from the research papers is a time consuming task for the presenter. So, it would be a great benefit for them if there is an easy way for generating the slides from the research papers. Also, sometimes people want to go through the research papers just to understand the overview of them. Slides provide an easier way for them to understand the overview of the topic. This gives rise to need for a tool which automatically generates slides from research papers. Automatic generation of slides from research paper is an easier task compared to generation of slides from some random document because research papers have an almost similar structure and we can address the problem of automatic generation of slides from research papers by exploiting the structure of the research papers. Once this problem is addressed, this work can be easily extended to automatically generate slides from other structured entities like book chapters etc.

2. Problem Definition

Given an HTML version of a research paper which is in accordance with the conference/journal proceedings, a slideshow is created which contains an summarized version of the paper. This Slideshow will contain the important points, tables, graphs, figures which helps to explain the paper. This may not serve as a final presentation but provides a good starting point for preparing it.

3. Related Work

There has been limited work that addresses the problem of automatic slides generation. Previous works include approaches like section-wise summarization [2], alignment methods for matching document regions with presentation regions [1], extracting topics and itemized elaborations from tagged documents. Many of these systems use multiple ideas from Information Retrieval, such as TF-IDF weights, query expansion, query-specific summarization, POS-tagging, etc.. Some of them are purely generative, while others learn models from an existing corpus of document-presentation pairs. These systems are tuned to work with different input formats such as PDF, XML, LaTeX and PPT, each of which preserve a varying amount of semantic and structural information about the original text. In the following two subsections, we review two papers [2],[1] which particularly deal with technical papers.

3.1. SlidesGen

The first work we review is by Sravanthi et al. [2]. They propose an novel framework for automatic generation of presentation slides for technical papers. They take LaTeX documents of research papers as input, and return the presentation slides. Their method depends on the assumption that by and large, conference papers have a similar structure: an abstract, followed by sections that can be broadly classified as introduction, related work, actual work

[*] IIIT, Hyderabad
[†] @students.iiit.ac.in

(model), experiments, conclusion/results and bibliography. A slide in their system contains a title and some bulleted points that are important in that section. They evaluate the system by surveying the response of people who use their system.

Their system is divided into multiple stages. The first stage is pre-processing stage. In this step, the LaTeX documents are converted into XML using a public domain converter LaTeXML.

Next, the generate what they call "configuration file", which contains configuration parameters for each section (since each section has a different point of view and writing style). This involves categorization of the section and extraction of key phrases from the section which are then stored in the configuration file. For example, a section with large number of cite tags, or with title containing words such as "related work" or "literature survey" is categorized as related works section.

The next step is of extracting key phrases. Most research papers have associated key phrases that can be help categorize the content in paper, and contain important concepts introduced in the paper. They are mostly related to model and experiments section, and can be used to summarize the same. So, the keywords given at the beginning are added to keyphrases for those sections in the configuration file. Also, few other phrases as the names of subsections etc are also added to the configuration file.

Next, they use QueSTS summarizer to summarize the model, experiment and conclusions section. QueSTS represents the text as a Integrated Graph where sentence is a node, and edge exists between 2 sentences if the sentences are similar. The edge weight is defined as the cosine similarity between the 2 sentences (above a given minimum threshold) Given the keyphrases computed previously, they are tokenized and a centrality based (to the query phrase) node weight is computed for the node corresponding to each token. Thus, they get query specific node weights and query independent edge weights.

From the above graph, they construct a Contextual Tree for each term q and node r. All the trees at node r are merged into a Summary Graph at r. The SGraphs generated from each node are ranked using a scoring model, and the one with best rank is returned as the summary.

The final step of this procedure is slide generation from the XML and configuration file. The introduction slides are generated by comparing introduction sentence with the abstract using consine similarity and ones with high similarity are placed in the slides. They give a similar method for generating slides from related work section as well. From model and experiments section, the slides are generated using the above QueSTS mechanism. Similarly, conclusion slides are generated using comparison with keywords such as "proposed", "concluded" etc.

Another important aspect in slides is graphics. Graphics are added along with sentences that either refer to it or is present along with it.

The paper finally discusses the issues in alignment of sentences and generation of slides. They also evaluate their slides manually with multiple users, taking their feedback. Most users gave a satisfaction level of more than 8/10 to the slides generated.

3.2. Alignment Methods

This paper by Brandon et. al. [1] takes a different approach to slide generation, inspired by the human throught process while making a slide out of a paper. They focus their efforts on *alignment* methods - first breaking up the document and presentation into regions and then performing a matching on them. *Slide regions* include bullets, headings, and other text spans, while *Paper regions* include paragraphs, section headings, and list items.

They generate their corpus of 296 paper-presentation pairs from workshops of technical conferences, through simple searching. The papers were PDF format, and presentation were a mixture of PDF and PPT. Before working with them, they convert them into custom XML formats, which represent relevant parts of the original data as logical regions (orthographic boundaries). They prefer such physical regions over semantic regions for its simplicity to implement and verify.

The alignment problem now reduces to an IR query, where the query is a slide region (which is the section/subsection the slide is related to), and the documents are the target regions from the paper. They compare two TF-IDF based scoring methods, with and without query expansion, resulting in 4 alignment methods.

The procedure of scoring is as follows. For each token in each region TF-IDF is computed, where TF is the frequency of the token in the region and DF is the number of regions containing the token's stem. The slide region is tokenized and POS tagged to remove non-content words. Each token in the query is stemmed and then may or may not be query-expanded depending on the method. A score is then calculated for each target region with the query. Two scoring methods were used - one uses the average TF-IDF score of the search terms relative to the target region, and the other uses the quantity of the matched terms.

The evaluation of their methods gives critical insights. First, a vast majority of the slide regions are not alignable (zero score with all target regions) - meaning that a lot of information in slides is not present in the paper - contrary to their hypothesis. Then they define an *alignable* accuracy against the *raw* accuracy, considering only alignable slide regions. They find that the best algorithm on an average gives an average 75% alignable accuracy, but only 50% raw accuracy. Query expansion seems to have little or negative

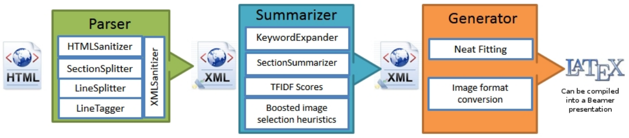

Figure 1. Block Diagram of the system

impact on the aligners and that the second scoring method is better than the first.

After more results, they conclude that the data indicates that the task of presentation generation is highly dependent on the end purpose the presentation will serve, as well as the target audience and other factors. Also query expansion generally degraded performance, possibly because authors tend to use wording in slides similar to their paper, and that using synonyms for query expansion is not aggressive enough, and may require hypernyms, immediate hyponyms, and other semantically related terms. Also a possible loss of accuracy could be polysemy of words, causing query expansion to be incorrect and insensitive to context.

4. Approach

Our approach consists of 3 stages, parsing, summarization and slides generation. Parsing converts the HTML into a clean, well-defined XML format that is used for further processing. Summarization prunes the XML using a series of heuristics, to select the sections/lines that would be most relevant for the presentation. Finally, generator stage converts the pruned XML into LaTeXBeamer presentation. Each of the stages have an intermediate output, that can optionally be tweaked manually by user to achieve better results later, such as tweaking the set of keywords extracted after parsing stage can help summarization work better. We discuss each stage in detail next.

4.1. Parsing

The parser is the frontend of the system as a whole, and requires to be robust against a wide range of input HTML styles. While broken html tag structure itself can be repaired upto some extent by tools like tidylib, the need to construct a structured xml from the HTML itself required us to write our own parser. This is because a standard parser will either output a stream of HTML tags or a DOM tree - both of which make it difficult for the parser to see a continuous context of data, split across too many tokens or tag depths. The parser would then become a complex state machine travelling the tag stream or the DOM tree back and forth, simply blowing up the complexity. Instead, our self-written parser does a constant number of stateless runs, each linear in the size of the HTML data. The parser can handle a variety of non-uniform heading hierarchies, match images with neighbouring captions, smoothly identify lines from paragraphs, and tag lines with images and references they refer to. In each of the continuing subsection we will see how each pass of the parser accomplishes one or more of these tasks.

4.1.1 HTML Sanitizing and Image Extraction

In this run we apply various filters to clean the input HTML. This includes removing comments, scripts, cropping to the body tag only, removing non-emphasizing font modifiers and condensing multiple whitespaces and whitespace tags. We also identify images and search for possible captions in their spacial neighbourhood. All these are kept in an in-memory list, containing their id, source path, their ”alt” text and the caption text. This list will be used later, to tag lines inside paragraphs with images they refer to.

4.1.2 Recursive Section Hierarchy Construction

In this run, we take the cleaned HTML, and search for all instances of heading tags (starting at Level1 - <h1>), extract the heading name, and range of text inside the heading. We start a new section tag in the xml, and write the list of lines under this heading, and then recursively do this for the next heading level (Level2 - <h2>), and then close the tag. The first section is searched for the title of the paper, as well as author details like their name, email, etc.. The keywords section is handled specially, and put at the end in a seperate keywords tag in the xml, due to its importance in summarization. The actual recursion is slightly more complex and two-dimensional due to the need to maintain two heading levels - the literal HTML tag level, and the structural level. This handles the non-uniform heading hierarchy problem.

4.1.3 Line Splitting and Tagging

After obtaining appropriate section splits, we tokenize them into lines, reducing the job of the summarizer to a selection algorithm on these lines, keeping the most informative and important content. Lines are first speculatively split using the string ”. ” (i.e. a dot followed by a space). However it causes broken lines in many situations - like after acronyms (e.g. , v.s.,), inside names (John L. Peterson), sentences inside quotations (”), numbered points (1.), etc.. A line smoothener uses various features like the length of previous sentence and current sentences, the number of words in previous sentence and current sentence, the parity of the count or the matching of quotations, apostrophies or paranthesis in the previous line, etc.. to suggest a merger of the two lines. This kind of smoothening ensures that only complete and sensible lines appear in the final presentation, adding to both its completeness and aesthetics.

Lines are also searched for references to images, by comparing them with the image alt texts or captions, and matching image IDs are added as their attributes. References inside lines (number sequences inside square brackets), are removed and also added as attributes.

4.2. Summarization

The second step in the slide generation process is that of summarization of the text as present in the complete research paper. The parsing step returns the structure of the paper in our predefined XML format, with all the lines lines, sections, subsections in the hierarchical order, and the objective of this stage is to select only those lines that are most informative, and would be most relevant for a presentation. We applied different heuristic techniques to summarize different parts of the paper, as discussed in following sections.

4.2.1 Summarizing Introduction

Introduction is an important section that introduces the problem, the motivation to solving it and gives an overall direction towards the solution, along with discussion an overview of the contributions. Abstract of the paper also summarizes the paper, albeit in a more succinct format. Hence, we use the approach as proposed in [2]. For each sentence in introduction, we compute it's TF-IDF based cosine similarity with the complete abstract text, and select the top n sentences with highest scores.

4.2.2 Summarizing Model

We consider the central part of the paper, with the approach, results, analysis etc. to be the model part of the paper, that contains the bulk of information content. The information itself may be divided into multiple subsections and bulleted points, and may contain images, tables, mathematical proofs/expressions etc.

Based on our experiments, we observed that the sections tree structure as extracted by the parser is an important part of the paper, and we include each of the sections and subsections in the final set of slides in the same hierarchy as detected in original paper. However, we summarize the text content in each of those sections recursively. We use the following set of heuristics for summarizing lines in each:

1. **Bulleted Points** Some sections in the papers itself contain a set of points in bulleted fashion, or with numbering. Such points might also be spread across with text in between, but have continuous numbering. In such a case, we only take those points to be representative text for that section.

2. **Other Text** In case the section does not contain bullets, we manually select a subset of the lines from that section. We determine a score for every line by computing its TF-IDF score with the set of keywords relevant to the paper. Lines with frequent occurance of keywords is usually more relevant, and we tend to select such sentences.

3. **Boosting image references** Since images are an important part of presentations, we boost the score of lines with image references. That increases the possibility of that line getting included, and hence of the associated image getting included in the slides as well.

Keyword Set Expansion The model section summarization depends heavily on the set of keywords, which we mostly extract from the keywords section of the paper itself. However, this set is usually very small, and sometimes ineffective for selection of best subset of sentences, hence we propose an algorithm to expand that set using the paper itself. We use co-occurrence probabilities to compute the other potential keywords, according to Algorithm 1.

```
Data: allLinesSet, keywordsSet
Result: expandedKeywordsSet
pairs = []
for line in lines do
    if line contains any keyword then
        for token in line.tokenized do
            $pairs \leftarrow (token, keyword)$
        end
    end
    for pair in pairs do
        if frequency $>$ threshold then
            $expandedKeywordsSet \leftarrow pair.token$
        end
    end
end
```

Algorithm 1: Keyword Set Expansion

4.2.3 Summarizing Conclusion

Summarizing conclusion requires us to select sentences that mention the major contributions of the paper, along with its consequences and implications. Here we use the approach as proposed in [2]. We define a set of words such as ["proposed", "shown", "contributed", concluded"] and so on, that are present in important sentences in conclusion. We compute TF-IDF based cosine similarity of each conclusion sentence with the above set, and select the top n.

4.2.4 Selecting images, references

We select graphics for display in the slides only if the sentence referring to that graphic gets selected. However, since images an integral part of presentations, we boost the score of

4.3. Slides Generation

The third step in the slide generation process is generation of slides using the content returned by the summarizing step. The summarizing step summarizes the content of the paper and returns the important sentences to be included in the slides. The Generator takes these sentences along with the images and references, which these sentences are referring to, and creates slides using them.

The number of slides will depend on the total number of characters in the sentences selected. We ensure that there is atleast one dedicated slide for each section and subsection. The order of slides is same as the order of sections/subsections in the paper. All the slides are assigned same title as their corresponding section title. If there is a slide having a sentence which is referring to any graphical element, then corresponding element will be displayed in the slide next to this. And if in a slide, there are lines referring to other documents/links, then these references are displayed at the bottom of the same slide. An overview of how the lines are included in the slides is presented in the Algorithm 2.

```
Data: selectedLinesSet, title
Result: sequenceOfSlides
$n$ = maximum number of characters per slide
$linesConsideredSet$ = []
$imagesSet$ = []
$referencesSet$ = []
for line in selectedLinesSet do
    if len(linesConsideredSet + line) $>$ n then
        Create a $slide$
        Set the $title$ of the $slide$
        Add the $linesConsideredSet$ to the $slide$
        Add the $referencesSet$ at the bottom of the $slide$
        Append the $slide$ to the output presentation
        for image in imagesSet do
            Create a $slide$ for the image
            Append the $slide$ to the output presentation
        end
    end
    $linesConsidered \leftarrow line$
    $referencesSet \leftarrow$ references, the line citing to
    $imagesSet \leftarrow$ images, the line referring to
end
if linesConsideredSet is not Empty then
    create a $slide$ for the lines remaining in the $linesConsideredSet$
    Append the $slide$ to the output presentation
end
```

Algorithm 2: Slide Generation

Table 1. User Satisfaction Survey

User	Q1	Q2	Q3	Q4
User 1	9	9	8	9
User 2	9	7	7	8
User 3	10	8	7	9
Average	9.33	8	7.33	8.33

5. Datasets

We used papers from SIGCHI'96 conference proceedings as our dataset. This dataset is given to us by Prof. Manish Gupta. In this dataset, there are 53 papers in HTML format. Out of these 53 papers, we used 40 papers for testing our system. The remaining 13 papers are not usable as some of them have corrupted HTML, some of them do not have the heading hierarchy, they were just flat text dumps without sections and some of them have irreparable errors in their HTML tag structure. Each HTML page is a full size research paper having sections Abstract, Keywords, Introduction, Methods, Experiments/Evaluation, Future Work, Conclusions, Acknowledgements, References with some exceptions. On an average, each paper in the dataset has 250 sentences. The intermediate output returned by the parsing step, on an average, has 80 sentences.

6. Experimental Results

what metric was used for evaluation, experimental settings, accuracy, plots (if any), tables.

Since the quality of the output of our solution is defined in highly subjective terms, we couldn't use any quantitative metric directly to evaluate our results. Hence we did user evaluation by reading the papers and rating the presentations generated for those papers. We selected a set of 10 papers randomly, and gave scores (out of 10) on following parameters:

- Q1: Information coverage by presentation
- Q2: Coherence between slides
- Q3: Closeness to the final presentation
- Q4: Overall satisfaction

The average response is summarized in Table 1.

6.1. Inter-Judge Similarity

We use the Fleiss' κ measure [3] to estimate the inter judge similarity (since the common Cohen's κ measure only works for binary categorization of input, whereas we want to give scores out of 10). Each cell of the table gives the number of users who gave score i on parameter j. Let $N = 4$ be the total number of papers considered and $k = 10$ be

Table 2. Inter-Judge Agreement

Score	P1	P2	P3	P4	Total	P_j
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0
7	1	1	0	2	4	0.33
8	2	0	3	0	5	0.41
9	0	1	0	1	2	0.16
10	0	1	0	0	1	0.083
P_i	0.33	0	1	0.33		

the number of ratings for each parameters, number of raters $n = 3$, and let n_{ij} represents number of raters who gave score j to paper i. For each row, P_j, proportion of papers getting j^{th} score, is defined as:

$$P_j = \frac{1}{Nn} \sum_{i=1}^{N} n_{ij}$$

For each column, P_i gives the extent to which the raters agree for i^{th} subject

$$P_i = \frac{1}{n(n-1)} \left[\left(\sum_{j=1}^{k} n_{ij}^2 \right) - (n) \right]$$

The above values are summarized in Table 2. Now, to calculate κ, we need,

$$\bar{P} = \frac{1}{N} \sum_{i=1}^{N} P_i = \frac{1}{4} (0.33 + 0 + 1 + 0.33) = 0.415$$

and

$$\bar{P_e} = \sum_{1}^{k} P_j^2 = 0 + 0.1089 + 0.1681 + 0.00689 = 0.2838$$

Hence,

$$\kappa = \frac{\bar{P} - \bar{P_e}}{1 - \bar{P_e}} = \frac{0.415 - 0.283}{1 - 0.283} = 0.1831$$

This value of κ shows there was slight agreement between the judges as per [3].

7. Analysis of Results

Analysis of our results shows that the entire system works convincingly well enough, serving our original objective of automated slide generation. The output generated by the system is a good starting point, with all structural information - like paper details, author details, section hierarchy, the summary of every section, relevant images and citations - in place. The parser was versatile enough to reconstruct proper structure from a majority of the HTML dumps of the papers from the dataset. Also as noted by [2], purely statistical methods were good enough to obtain a minimum level of coherence and information during summarization. The various methods of the summarizer like keyword expansion through co-occurance frequencies and evaluating importance of lines using TF-IDF like similarities with abstract and keywords, boosting lines with tagged images, etc. never produced irrelevant summaries. The combination of various deterministic as well herusitic rules we applied at each stage of our pipeline worked well together to give an output that is both information preserving as well as aesthetically pleasing.

8. Conclusion

In this paper, we proposed the framework of an automated system which generates a neat LaTeX presentation from a technical paper given in HTML format. The generated slides might not be exactly what the presenter wanted but they serve as a good starting point to convert it into a final presentation. The system we proposed generates the slides in 3 stages. At any stage, the users can optionally modify the content in the intermediate output like adding more keywords after the parsing stage, adding/removing sentences after the summarizing stage which will help the system to generate better presentations.

9. Future Work

Our approach and mix of papers we referred, all use statistical methods like measuring similarity between keywords, abstract, introduction and the inner sections to extract the content which is to be included in the slides. More natural and smoother summarization is possible using Natural Language Processing (NLP) methods.

References

[1] B. Beamer and R. Girju. Investigating automatic alignment methods for slide generation from academic papers. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 111–119, 2009.

[2] M. Sravanthi, C. R. Chowdary, and P. S. Kumar. Slidesgen: Automatic generation of presentation slides for a technical paper using summarization. In *FLAIRS Conference*, 2009.

[3] Wikipedia. Fleiss' kappa — Wikipedia, the free encyclopedia. [Online; accessed 20-Nov-2013].