Project Reporting Management System with AI based Assistive Features for Text Summarization

Jin Boon Benjamin Tan, Quan Chen, and Chai Kiat Yeo

***Abstract*—This paper details a proof-of-concept system called Project Reporting Management System (PRMS) to manage the project reporting process in a typical research centre where the process can be manual for many centres. In fact, it is general enough to be scaled up and deployed for a large department or scaled down for a smaller setup in any organization which needs a simple and efficient project progress reporting system but does not entail the kind of complexity and cost of commercial project management systems. Using a research centre scenario, the progress of the individual projects has to be tracked through the periodic submission of progress reports by the Principal Investigator (PI) of the project. The centre will need to consolidate these individual reports manually into a consolidated report and an executive summary for higher management. PRMS automates the tracking of individual projects and reporting deadlines, sends reminders and allows online submission of reports by the PIs. PRMS also incorporates assistive and automated features exploiting Machine Learning (ML) and Natural Language Processing (NLP) techniques to generate the consolidated report and rank sentences of verbose report for assistive text summarization to facilitate the manual process of producing an executive summary.**

***Index Terms*—Web-based system, reporting process automation, machine learning, natural language processing, sentiment analysis, text summarization.**

I. Introduction

A typical research centre has many projects running at the same time. The progress of all projects has to be tracked by the management through periodic progress report submissions by the respective project principal investigators (PI). These individual reports have to be consolidated to produce an executive summary to facilitate onward reporting to higher management. In many centres, the process is manual as in the staff member in charge of project reports will have to email all PIs to submit their progress reports on a regular basis. Monitoring of reports submitted, the deadlines and reminders to PI is usually being carried out via emails. The staff will then have to browse through all the reports and gather the important progress so that they can be consolidated into an executive summary for higher management to peruse. Important information such as lack of progress and problems encountered has to be highlighted so that remedial action can be carried out to restore the projects back on schedule. The process is laborious and time-consuming.

A web-based proof-of-concept system called Project Reporting Management System (PRMS) is hence proposed to automate as much of the manual pipeline of project progress reporting as possible. PRMS automates the tracking of individual projects and reporting deadlines, sends reminders and allows online submission of reports by the PIs. PRMS also incorporates assistive and automated features to further automate the reporting process. It exploits Machine Learning (ML) and Natural Language Processing (NLP) techniques to rank sentences from the reports to facilitate the manual process of producing an executive summary. Although this paper uses a research scenario as the application example, the proposed PRMS is general enough to be deployed for a large department or for a smaller setup in any organization which needs a project progress reporting system but does not entail the kind of organization functional integration, complexity and cost of commercial project management systems such as Microsoft Project [1], Tiemchart [2], Monday [3] etc.

This paper is organized as follows: Section II provides an overview of PRMS and its implementation. Section III details the assistive and automated features to extract important sentences by ranking them for report consolidation and facilitate preparation of the executive summary. Section VI concludes the paper.

II. Overview and Development of PRMS

The requirements of PRMS is a fully functional enterprise system for management of research project reporting which include the design and development of the system, user interface and digitization of the process flow and the automation and optimization of areas within the reporting framework that are identified to be manually intensive or are prone to human-error or failure. Agile Project Management methodology is adopted in the development of PRMS.

A. System Requirements

Without an automated tool, administrators will need to manually consolidate all the individual project reports into a single report. Prior to this process, the individual reports must be reviewed and approved by the administrator to ensure they are of sufficient quality and adhere to standards set out. Owing to the pipelined nature of this process, administrators face the difficulty of tracking the status of individual reports across projects (as only approved reports are consolidated). Furthermore, the entire process of report consolidation is repetitive, tedious and prone to human error and formatting error. Therein lies the motivation for PRMS.

Manuscript received November 16, 2019. This work was supported in part by Grant No. NTU M4082227. The authors also wish to thank Singtel Cognitive and Artificial Intelligence Lab for Enterprises@Nanyang Technological University (SCALE@NTU) for providing the project reporting requirements of a typical research centre.

Jin Boon Benjamin Tan, Quan Chen and Chai Kiat Yeo are with School of Computer Science and Engineering, Nanyang Technological University, Singapore (e-mail: jtan346@e.ntu.edu.sg, qchen@ntu.edu.sg, asckyeo@ntu.edu.sg).

doi: 10.18178/ijmlc.2021.11.1.1009

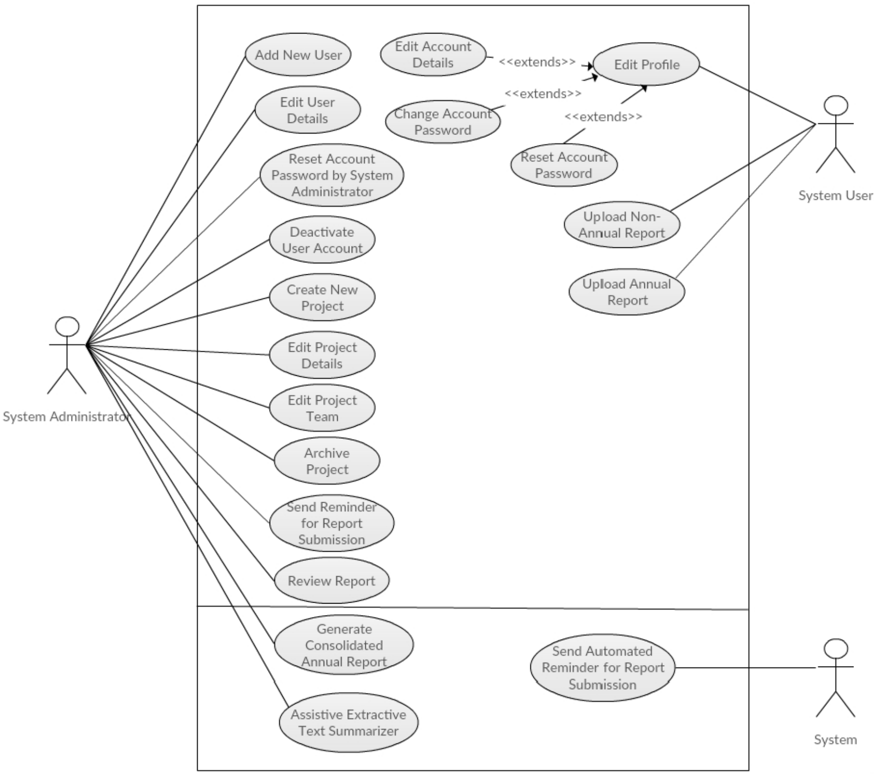


Fig. 1. Use case diagram of PRMS.

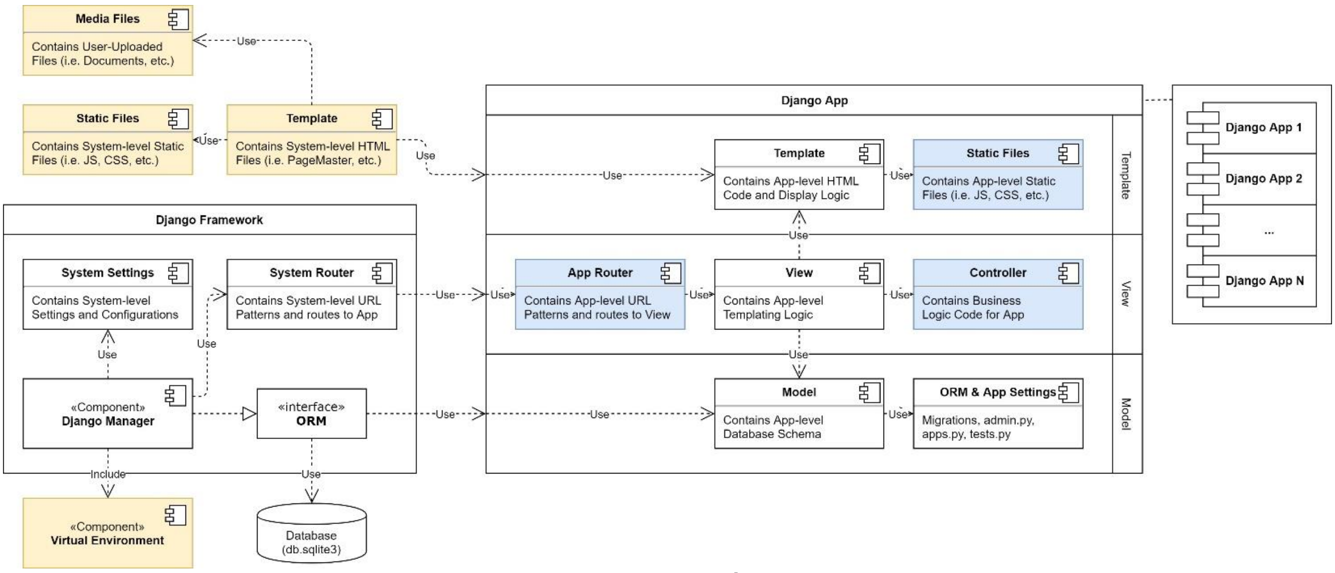


Fig. 2. PRMS system architecture.

Fig. 1 shows the use case diagram for PRMS. The functional requirements are as follows:

1) Provision of administrator and user account types with different levels of access privileges. PIs or project leads are able to log in with a user account, Provision of administrator and user account types with different levels of access privileges. PIs or project leads are able to log in with a user account, reset and change password and edit user profile and account information. System administrators are able to log in with an administrator account and such account must contain all the user functionality for account access except changing the user password. Administrator can create, activate, deactivate and delete user accounts.
2) Provision of project management functionalities with controlled access according to account types. Users can access projects according to their project membership such as PI/lead and members of project team and view, edit and upload reports. Besides details of the project, reporting requirements such as the schedule, type of reports, deadlines etc. are also included in the project profile. Administrators can perform project management tasks which will be a superset of those of the users. Administrators can define project access, assign the roles of the different users to the projects, add, delete or archive projects, edit project information, define the reporting types and deadlines. Administrator can define the report types needed for each project and their associate deadlines. Report types can be annual, bi-annual, quarterly, bi-monthly, monthly, fortnightly or weekly.
3) Provision of report management functionalities with controlled access according to the account type, project

role and project reporting requirements. Users can access their reports based on project role and project reporting requirements as well as uploading new reports. Administrators can perform all report management tasks for all projects, view and download reports including review actions, namely to approve or o reject with comments. Administrators can activate the automatic sending of email reminders.

4) Automatic report extraction for users to effectively preview the uploaded report prior to submission.
5) Assistive tools for automatic generation of consolidated report.

B. System Architecture

The system architecture of PRMS is modified from the Django framework [4] and is shown in Fig. 2. Extensions to the Django framework comprised system-level components (the yellow shaded blocks on the left of the figure) and app-level components (the blue shaded blocks on the right of the figure). The added system-level components include a virtual environment to store and manage packages, libraries and modules that are not part of the standard Python library as well as directories for system-level templates, static files and media files uploaded by the users. Added app-level components include a URL router, directories for static files (immutable files used in the templates) and controllers. The app-level controller directories contain functional codes and data (e.g. stored models from machine learning modules) relating to each app's business-logic. This structure will separate controller (business-logic) code from templating logic code in the app's views to facilitate system development, unit testing and code readability.

C. System Test

Acceptance Test Driven Development (ATDD) is used as the base stratagem as User Acceptance Tests (UAT) is carried out during the gathering of feedback for upcoming iterations of development [5]. The various tests show that PRMS has fulfilled the functional requirements set out in Section II-A. In addition, it is to be noted that the longest path to execute a user, project or report data manipulation function in PRMS is no more than 4 interactions within the application, including any prompts and messages for best user experience.

Fig. 3 shows an example of a PRMS screenshot. It displays the menu available to the Administrator (on the left tab), a listing of all the projects and the corresponding reporting requirements and deadlines. In particular, it shows that the administrator has the option to manually trigger email reminder to a project's PI which is in addition to the automated scheduled email reminders.

III. Assistive Automatic Tools for Text Extraction and Summarization

A. Toolkits Used

The libraries used to implement the NLP features are spaCy [6] and AllenNLP [7]. SpaCy offers a wide range of NLP-related functionalities, such as tokenization, POS tagging of extracted textual data and text preprocessing for deep learning tasks. AllenNLP facilitates design of deep learning tasks related to NLP. AllenNLP is used here to implement the learning algorithm for sentiment analysis.

B. Template Based Automatic Text Extraction

Text Extraction involves the extraction of key sections of the individual reports. This extraction is based on a standard report template. In our application scenario, five important sections are specified, namely, Milestones and Deliverables, Expenditures and Justifications, Report Body, Publications and Intellectual Properties. The first two sections are in table form while the rest are free text format.

The Python library, *docx2txt* [8], is used to extract the text from the Microsoft word documents for subsequent parsing and text extraction. Parsing of the entire report comprises Text parsing, Paragraph parsing and Section parsing. Text parsing refers to the parsing of individual tokens (or regular expressions) within a sentence or paragraph object in the document file. Paragraph parsing parses the individual paragraphs or token candidates within a paragraph object while Section parsing parses paragraphs and paragraph styles (headings) within the document to demarcate the sections. All words in the report body are tokenized using spaCy. Pre-processing is done to convert all words to lowercase and to remove all stop-words and non-alphabetical characters. Table I shows the report sections and the associated extraction strategy applied. The extraction accuracy is 100% in our tests on three sample reports since the extraction is template based. It is to be noted that any changes in the report template will require changes to this section of the program codes.

TABLE I: Report Template and Text Extraction Strategy

	Report Structure		
Report Section (In Sequence)	***Type***	***Extraction Strategy***	***Accuracy***
Milestones and Deliverables	Table	Text Parsing	100%
Expenditure and Justifications	Table	Paragraph Parsing	100%
Report Body	Free Text/Section	Paragraph, Section Parsing	100%
Publications	Free Text/Section	Text, Paragraph, Section Parsing	100%
Intellectual Properties	Free Text/Section	Text, Paragraph, Section Parsing	100%

C. Automatic Text Extraction for Report Preview

Using the mechanism described in Section III-A, an automatic report extraction feature is implemented for effective preview of the uploaded report prior to submission by the user. User can perform a quick check on the key sections of the report before uploading to ensure that important details are not being left out providing an optimal user experience for the report submission process. Fig. 4 shows an example of the report. Note that Fig. 4 only shows two sections out of the five sections spelt out in Table I.

D. Automatic Generation of Consolidated Report

The same mechanism detailed in Section III-A is applied to the individual reports to extract the sections automatically consolidating into a final report. Fig. 5 shows the screenshot. Python libraries, *docxtpl* [9] and *docxcompose,* [10] are used to facilitate the document generation process.

E. Assistive Extractive Text Summarizer

The assistive extractive text summarizer (aETS) using NLP techniques is designed to help administrators in report analysis and summarization, via the ranking of individual sentences. aETS extracts a set of the most significant sentences from a document verbatim and comprises three steps, namely (1) creating an intermediate representation of the original text (2) scoring of sentences and (3) obtaining the most significant sentences [11]. The design pipeline is shown in Fig. 6. Output from the text extraction mechanism described in Section III-A is fed into the Assistive Extractive Text Summarizer module. The first horizontal shaded block (in yellow) shows the use of word frequency to score the word-type in each sentence. The second shaded block (in orange) shows the named entity extraction and the last block (red) performs sentiment analysis for sentence scoring. A scoring formula is derived to score each sentence within the document. We avoid techniques which require domain-specific training data as we do not have a large report corpora and settle on word-frequency-based extractive text summarizing technique [8] which comprises four steps as shown in the first shaded block of Fig. 6.

1) Sentence Scoring using Word-Frequencies: After the tokenization and pre-processing as described in Section III-A, all word frequencies are then obtained by iterating through the remaining report text, where raw word counts for each word-type are tabulated. These word counts are then normalized to provide the score of each word-type in the report body. Lastly, all sentences in the report body are scored based on all the present word-types in the sentence. Finally, post-processing is carried out to manually assign scores to sentences with certain properties, e.g. report headers, subtitles, minimum sentence length, etc.
2) Named entity recognition (NER): NER extracts, locates and classifies named entities from unstructured text into pre-defined categories. Named entities are real-world objects generally represented by proper nouns [12]. NER is used to remove named entities from the text prior to sentiment analysis and to add static scores to sentences. [11] has shown that removal of named entities or proper nouns leads to an improvement in extractive text summarization. The spaCy library, which is trained on the 'OntoNotes5' dataset and boasts a 85.85% accuracy, is used to perform NER.
3) Sentiment Analysis: Sentiment Analysis is used to identify and classify the polarity or sentiment, of a given text, where sentiment refers to the opinion or emotion contained within the text [13]. We use deep learning models for sentiment analysis [14] using the 5-class classification model due to its wide polarity range. The Stanford Sentiment Treebank (SST) Dataset [15] is used for the training due to its state-of-the-art annotation scheme. SST is the standard benchmark for neural network models that can capture the syntactic structures of sentence [16]. In SST, sentiment labels are assigned to every phrase and word in sentences in a nested tree structure. This allows the study the complex semantic interactions between words and phrases.

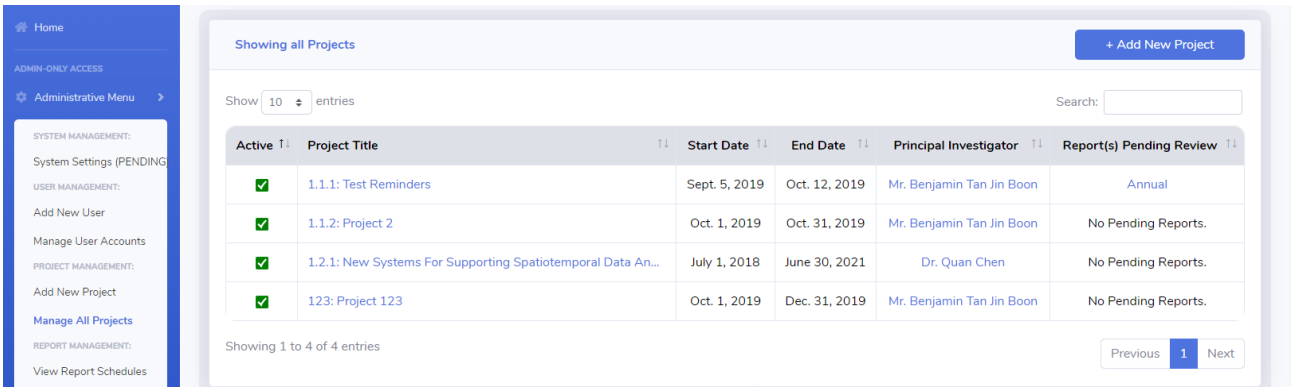


Fig. 3. Sample screenshot of PRMS.

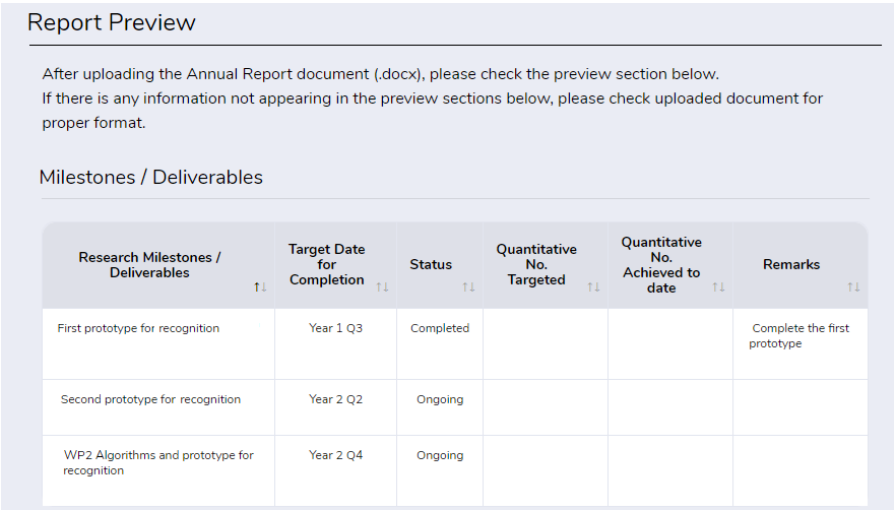


Fig. 4. Example of text extraction of sections of uploaded report for preview.

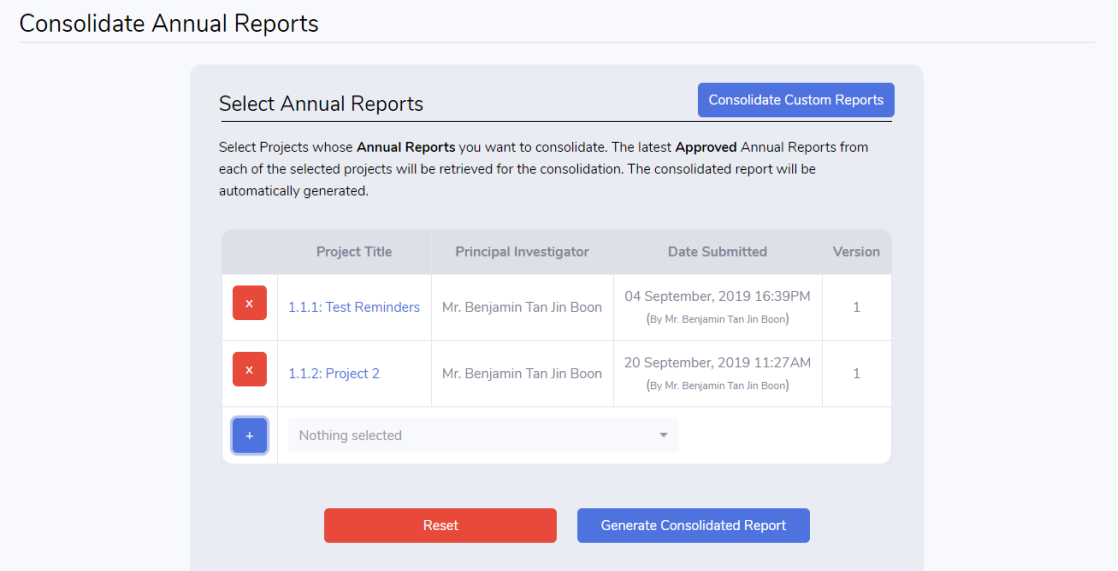


Fig. 5. A screenshot of the generated consolidated report page.

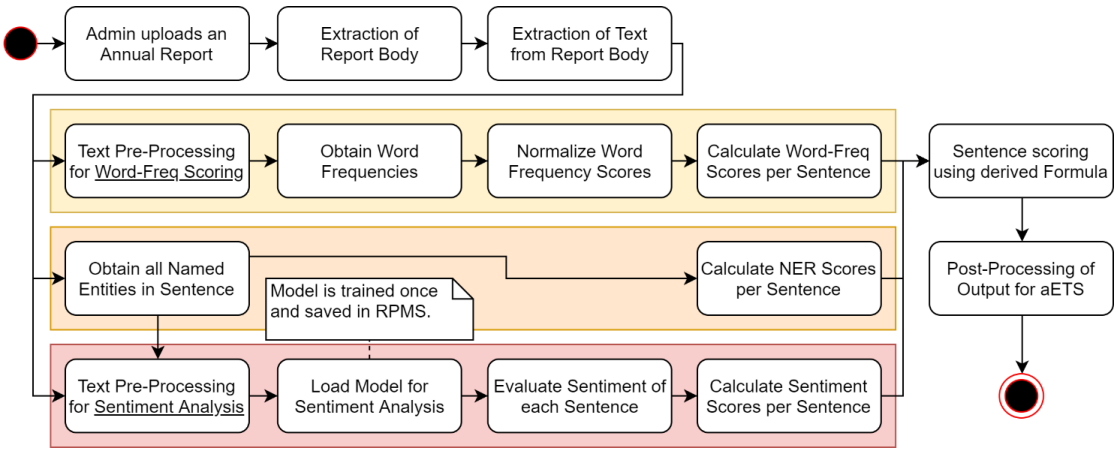


Fig. 6. Design of assistive extractive text summarizer.

The learning algorithm from AllenNLP library used here is the Biattentive Classification Network (BCN) and uses pre-trained word vector Embeddings from Language Models (ELMo) [17]. AllenNLP allows the pre-set learning algorithms to be fine-tuned to customize the models needed for aETS. While [17] reported an accuracy of 54.7% using the SST dataset, it only achieves 43.2% in aETS implementation.

Sentence scoring formula derived heuristically to improve accuracy: The output of the semantic analysis is a logit tensor containing the non-normalized output of the BCN+ELMo model. This tensor contains 5 logits, each corresponding to a class in the 5-class annotation of SST. The logit with the highest probability (argmax) is then used to classify the sentiment of the sentence. Sentiment analysis plays a huge role in the ranking of importance of sentences in PRMS.

Observations of logits produced show that while there are frequent cases of misclassification due to BCN-ELMo's low accuracy of 43.2%, there is a very high probability observed in misclassifications when (1) the ground truth is ±1 of the output class, and (2) the post-normalized logits have a low standard deviation with window size of 3 of the ground truth. Based on these observations and that the ground truth is unknown, a scoring formula is proposed as follows:

- Perform Min-max normalization of the logits
- Obtain window size of 3 by finding two neighbouring classes with the largest logit score and obtain the possible label
- Compute the confidence score of the result using the normalized difference between the score and the neighbouring classes' logit score

Compute the final static score depending on the output label versus the possible label, confidence score and flat score. This final score is then added to the computed score of the first two blocks in Fig. 6 as a static score instead of a multiplier. This is due to bias in the sentence scores of the former towards shorter sentences (i.e. a short negative sentence will have a lower score despite having a high, negative sentiment).

Fig. 7 illustrates the application of the aforementioned formula to a previously misclassified neutral sentence by BCN-ELMo to positive as can be seen from *Result* (Red box). The proposed scoring formula is able to detect the classification error and give the correct label of 'Neutral' shown as *Possible Output* (Green box).

F. Incorporation of aETS into PRMS

aETS is integrated into PRMS and Fig. 8 shows the user interface. The output of aETS is the ranked sentences which are displayed onto a user-friendly and intuitive interface focusing on assisting users in text summarization. Sliders on

the interface allow the user to freely adjust their top N number of significant sentences, which are displayed in bold.

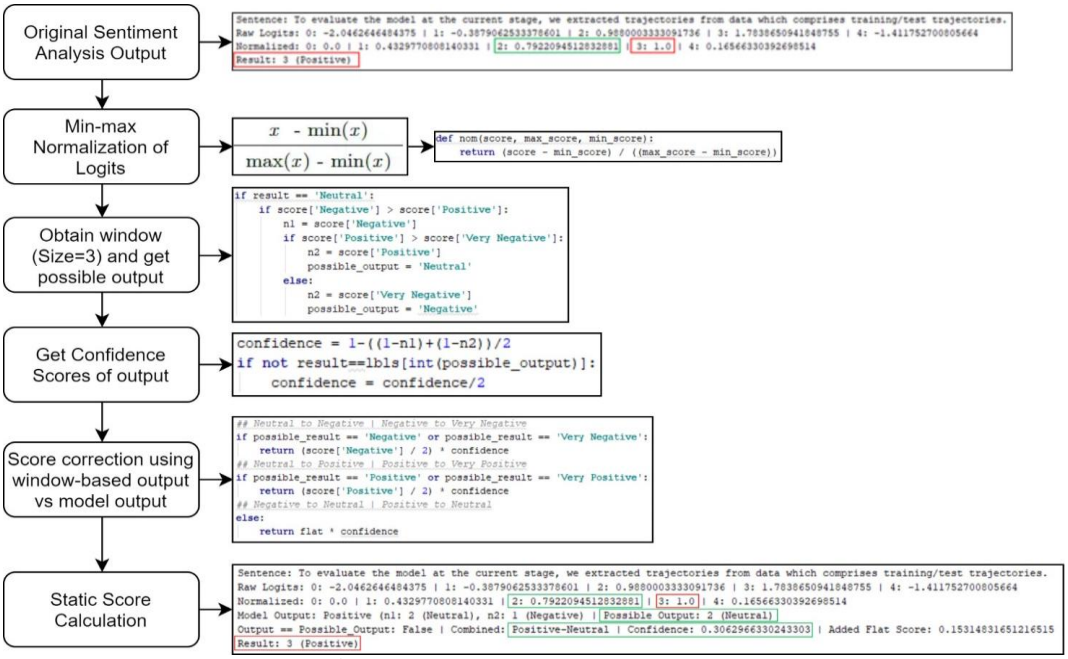


Fig. 7. Heuristic formula to compute sentiment static score to improve accuracy.

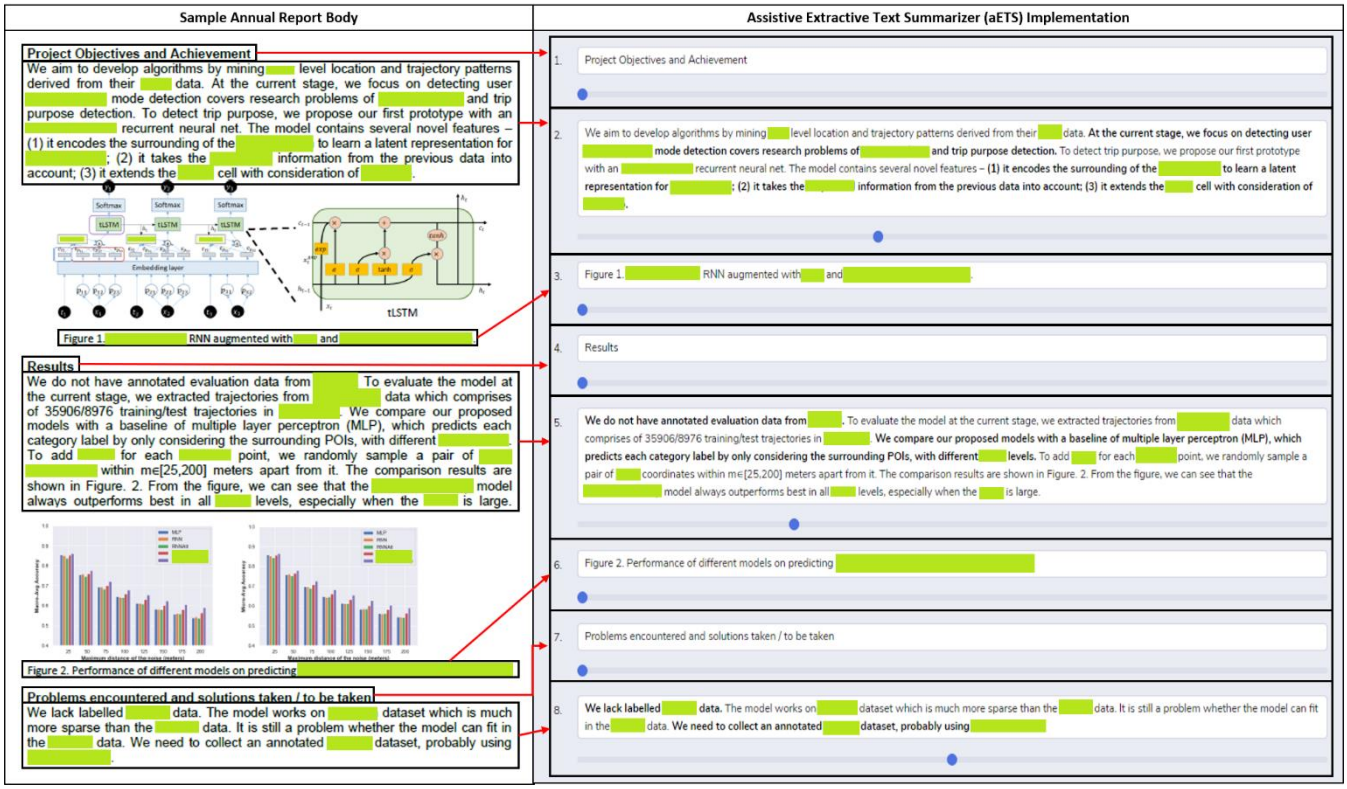


Fig. 8. User interface of aETS in PRMS.

IV. Conclusion

This paper details the development of a Project Reporting Management System (PRMS) to automate project reporting in the scenario of a typical research centre. In fact, the proposed PRMS is general enough to be scaled up and deployed for a large department or scaled down for a smaller setup in any organization which needs a simple and efficient project progress reporting system without incurring the complexity and cost of commercial project management systems. The progress of the individual projects has to be tracked through the periodic submission of progress reports and the administrator will need to compile these individual reports manually into a consolidated report and an executive summary for higher management. PRMS automates the tracking of individual projects and reporting deadlines, sends email reminders and allows online submission of reports by the PIs. PRMS also incorporates automated and assistive features exploiting Machine Learning (ML) and Natural Language Processing (NLP) techniques to generate the consolidated report. A proof-of-concept assistive extractive text summarizer has also been developed to rank the

sentences thereby facilitating the extraction of important portions of the reports to assist the administrator in preparing an executive summary.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Jin Boon Benjamin Tan designed and developed the system and conducted the research work involved. Quan Chen spelt out the system requirements, performed system analysis and conducted detailed testing and verification. Chai Kiat Yeo conceived the project, supervised the design, development as well as the research conducted and wrote the paper with inputs from all authors. All authors had approved the final version.

References

[1] Microsoft, Project. [Online]. Available: https://products.office.com/en-sg/project/project-management-software?rtc=1

[2] Tiemchart, Online Project Management Software.

[3] Monday, Project Management Software.

[4] Danjo, Danjo: The Web Framework for Perfectionists with Deadlines. [Online]. Available: https://www.djangoproject.com/

[5] ReQTest, *Agile Testing – Principles, Methods & Advantages*, 18 July 2018.

[6] spaCy. *SpaCy - Facts & Figures.* [Online]. Available: https://spacy.io/usage/facts-figures

[7] Allen Institute of AI. AllenNLP. [Online]. Available: https://allennlp.org/

[8] Python Software Foundation, Python Project. [Online]. Available: https://pypi.org/project/doc2text/

[9] Python Software Foundation, Python Project. [Online]. Available: https://pypi.org/project/docxtpl/

[10] Python Software Foundation, Python Project. [Online]. Available: https://pypi.org/project/docxcompose/

[11] R. Ferreiraa et al, "Assessing sentence scoring techniques for extractive text summarization," *Expert System Applications*, 2013.

[12] S. Li, "Named entity recognition with NLTK and SpaCy," *Towards Data Science,* August 17, 2018.

[13] S. Gupta, "Sentiment analysis: Concept, analysis and applications," *Towards Data Science*, January 8, 2018.

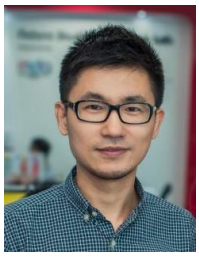
[14] "NLP-progress: Sentiment analysis," NLP-progress, [Online]. Available: http://nlpprogress.com/english/sentiment_analysis.html.

[15] Stanford University, Stanford Sentiment Treebank, *Deeply Moving: Deep Learning for Sentiment Analysis.*

[16] R. Socher *et al*., "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013.

[17] M. E. Peters *et al*., "Deep contextualized word representations," Allen Institute for Artificial Intelligence, Paul G. Allen School of Computer Science & Engineering, University of Washington, 2018.

Jin Boon Benjamin Tan is an undergraduate from Nanyang Technological University (NTU) and will receive his B.Eng in computer science with specialization in artificial intelligence and data science & analytics in 2020. His areas of interest include system design & development, machine learning, natural language processing, deep learning and information retrieval. He is particularly interested in the ideation and development of practical system tools using machine learning techniques.



Chen Quan received his PhD in computer engineering from Nanyang Technological University (NTU). He is a principal research fellow in NTU and his research interests include computer-assisted animation, computer graphics and game related technologies. He is also an associate director of Singtel Cognitive and Artificial Intelligence Lab for Enterprises (SCALE@NTU), managing various research projects in AI, data analytics, robotics and smart computing.

Chai Kiat Yeo received the B.Eng. (Hons.) and M.Sc. degrees both in electrical engineering, from the National University of Singapore and the Ph.D. degree from the School of Electrical and Electronics Engineering, Nanyang Technological University (NTU), Singapore. She was an assistant principal engineer with Singapore Technologies Electronics and Engineering Limited prior to joining NTU in 1993. She is an associate professor and was the deputy director of Centre for Multimedia and Network Technology (CeMNet) and the associate chair (academic) with the School of Computer Science and Engineering, NTU. She is currently the deputy director and programme lead of Singtel Cognitive and Artificial Intelligence Lab for Enterprises@NTU. Her current research interests include anomaly detection, machine learning, artificial intelligence, predictive operational analytics, ad hoc and mobile networks.