Sphinx: An Automated Generation System for English Reading Comprehension Assessment

Saad Khan (saad.khan@act.org) ,Yuchi Huang (yuchi.huang@act.org), Scott Pu, Vladimir Tarasov, Alejandro Andrade, Richard Meisner, Dave Edwards, Alina von Davier

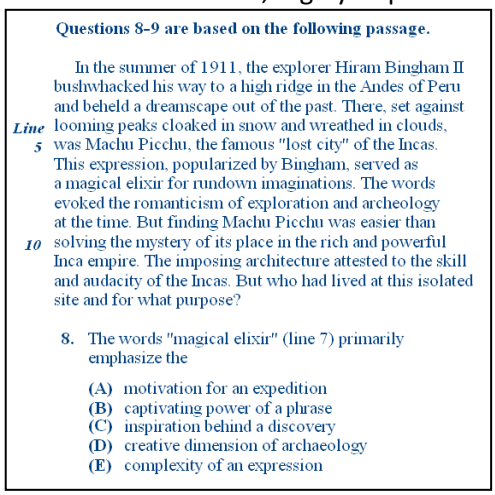
ABSTRACT: We discuss Sphinx, a human-AI hybrid system for scalable production of reading comprehension passages in English from writers' samples/prompts to be used in in a variety of learning and assessment. To the best of our knowledge, Sphinx is the first natural language generation system designed to create reading passages in a computationally efficient manner and can be used in a plethora of learning and assessment contexts. In Sphinx, we integrate state-of-the-art NLP approaches with the reasoning ability of writers to process text from a multiple of sources and produce industry-grade quality narratives and original content at the same time. We utilize highly capable NLP transformer models such as BERT, GPT2 and USE to encode text data and automate writer's tasks including, but not limited to topic modeling, auto-summarization, sentence recommendation and ranking, and paraphrasing. Furthermore, we integrate AEGIS (ACT English Item Generation System) into Sphinx to repeatedly produce items from source and composed essay text. Along with questions of quality and rigor, we pay special attention to the issues of parallelization and scalability that are pressing in the learning and assessment industries.

Format: Practitioner Presentation

Keywords: Natural Language Processing, Automated Content Generation

1 INTRODUCTION

Educational assessment, learning, and publishing companies dedicate significant resources for the creation of original text-based content for use in formative and summative tests, as well as in classroom learning or open educational resources. This process can be laborious, highly dependent on domain expertise and difficult to scale up. Furthermore, the manual generation of content and assessment items heightens the risk of incomplete, duplicate and/or redundant content. Automating educational content generation such as assessment items and in particular English reading passages (see figure 1 for a sample) can result in cost savings, quality standardization, and open new possibilities for personalized learning experiences.



Questions 8–9 are based on the following passage.

In the summer of 1911, the explorer Hiram Bingham II bushwhacked his way to a high ridge in the Andes of Peru and beheld a dreamscape out of the past. There, set against looming peaks cloaked in snow and wreathed in clouds, was Machu Picchu, the famous "lost city" of the Incas. This expression, popularized by Bingham, served as a magical elixir for rundown imaginations. The words evoked the romanticism of exploration and archeology at the time. But finding Machu Picchu was easier than solving the mystery of its place in the rich and powerful Inca empire. The imposing architecture attested to the skill and audacity of the Incas. But who had lived at this isolated site and for what purpose?

Line 5 10

8. The words "magical elixir" (line 7) primarily emphasize the

(A) motivation for an expedition
(B) captivating power of a phrase
(C) inspiration behind a discovery
(D) creative dimension of archaeology
(E) complexity of an expression

Figure 1: Sample reading passage and associated item. Manual creation of such passages can be a costly and inefficient process.

Classical natural language processing (NLP) work in this area dates back to John Wolfe's seminal work (Wolfe, 1977) that demonstrated the feasibility of automatically generating natural language questions. In recent years there has been a revival in interest, spurred in part by advances in dialogue systems such as Amazon Alexa. While

traditional approaches to NLP-based educational item generation involve a pipeline of modules such as content selection, template design and item realization (Gierl et al., 2012;), these have been criticized for being rigid and too reliant on arbitrary heuristic rules (Heilman, 2011). There is growing interest in developing end-to-end deep neural network based approaches that do not require customized, hand crafted rules and are better

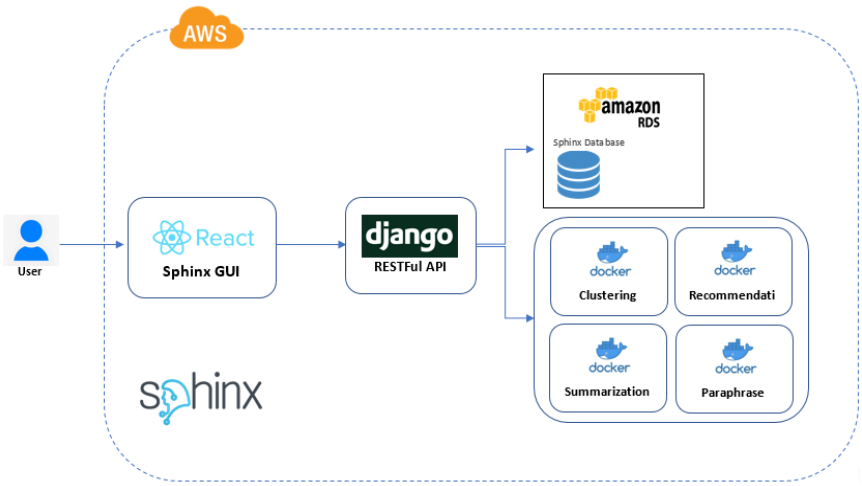


Figure 2: Sphinx system architecture is designed to be modular with distributed services hosted on AWS..

equipped to generalize across content areas (Cervone et al. 2019). A key element of such approaches is leveraging large text content databases and well annotated datasets such as BookCorpus (Zhu et al. 2015), SQuAD (Rajpurkar et al., 2016) and Wikipedia.

In this paper we discuss Sphinx, a scalable system that utilizes advanced NLP models to help expert or novice writers interactively create English reading comprehension passages from writers' samples/prompts. To the best of our knowledge, Sphinx is the first natural language generation system designed to create reading comprehension passages in a computationally efficient manner. Passages created by Sphinx could be used in a variety of learning and assessment applications such as formative and summative assessments of reading and comprehension and real-time adaptive learning. The system is designed to integrate state-of-the-art NLP approaches with the reasoning ability of writers to process text from a multiple of sources and produce industry-grade quality narratives and original content at the same time. Highly-capable NLP transformer models such as BERT (), GPT2 () and USE () are utilized to encode text data and automate writer's tasks including, but not limited to topic modeling, auto-summarization, sentence recommendation and ranking, and paraphrasing. The recommendations of passage content made by NLP models will always be evaluated by human users before inclusion. This interactive feature enables quality control for improved content validity as well as collecting training data for underlying machine learning models. We believe such human-AI hybrid systems can be the best of both worlds by utilizing the reasoning ability of subject matter experts while processing large amounts of input text to automate portions of the writing process. Furthermore, we integrate AEGIS (ACT English Item Generation System) into Sphinx to repeatedly produce items from source and composed essay text. Along with questions of quality and rigor, we pay special attention to the issues of parallelization and scalability that are pressing in the learning and assessment industries.

In the following, we describe the system architecture and the technology stack used in development. Especially, we present detailed framework of the Sphinx system, including its core NLP algorithmic modules: automated summarization, topic clustering, sentence recommendation and paraphrasing. We also introduce AEGIS (ACT English Item Generation System) at the end of the second section, and followed by the conclusion and discussion on the future work finally.

2 TECHNICAL MODULES

Figure 2 shows an overview of the Sphinx system architecture. Sphinx is designed as a distributed system with three main components. The first is a React JavaScript-based graphic user interface (Figure 3). Through the interface, users can upload or download passages, organize passages into folders, create projects, and use Sphinx 's NLP sub-modules to compose new passages. Users can also choose whether to enable the scaffold feature, so that new article composition will be divided into three parts: Introduction, Body and Ending. The user interface is linked to the second component, which is a Django REST API (https://www.django-rest-framework.org/). The API serves mainly as a gateway to the NLP machine learning algorithms. It authenticates user visits, manages processing requests and access to the system database. Sphinx's NLP algorithms form the third core component and include text summarization, topic modeling, sentence recommendation and paraphrasing. As shown in Figure 4, for most functionality modules we provide different NLP algorithms for users to choose from. Each algorithm is wrapped as a web service in a docker container (https://www.docker.com/resources/what-container) and remains dormant unless a processing request is received through the Django REST API.

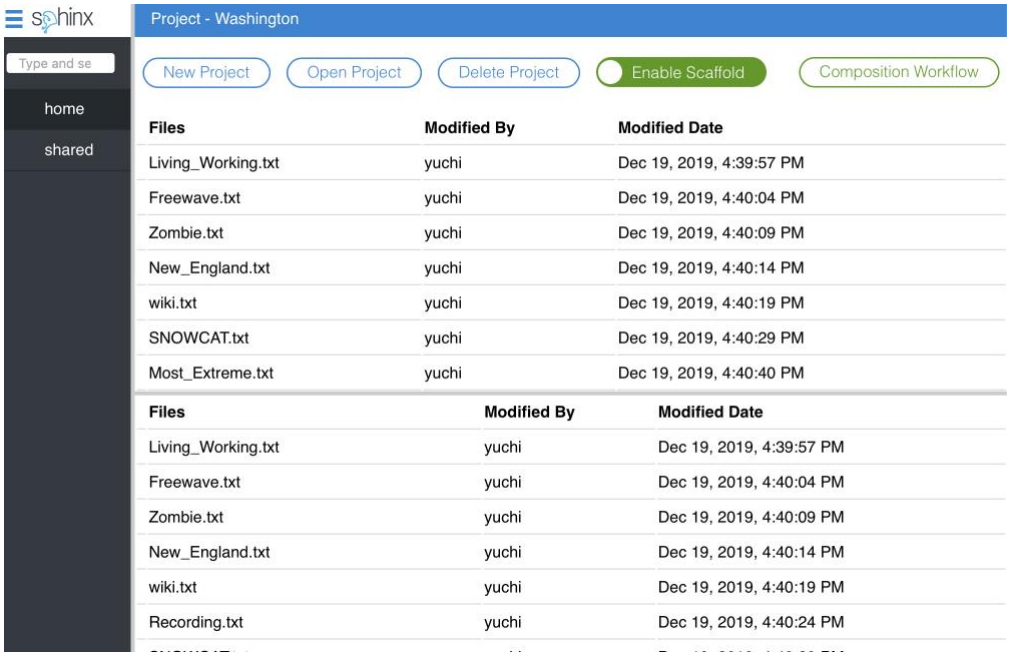


Figure 3: Through the interface, users can upload or download passages, organize passages into folders, create projects, and use Sphinx 's NLP sub-modules to compose new passages.

Currently, Sphinx's server and algorithms are all deployed on AWS EC2 instances. AWS provides auto-scaling services that can add or remove EC2 instances dynamically according to real (or predicted)

demand. AWS also comes with services like Elastic Loading Balancing that automatically distributed income traffic to different EC2 instances according to their current workload. Technologies like Elastic Beanstalk can take care of auto-scaling, load balancing, application health monitoring, and more according to configuration.

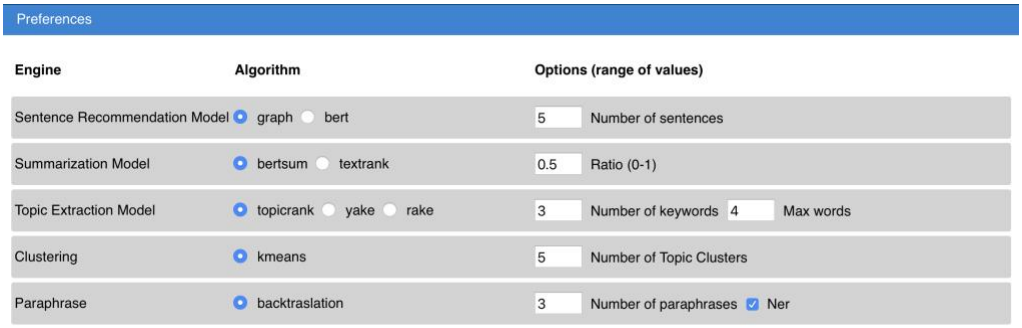


Figure 4: For most functionality modules we provide different NLP algorithms for users to choose from.

In the following we describe details of the NLP algorithmic core of Sphinx, functionalities that enable expert or novice writers to process raw digital text from a multitude of sources into new, coherent narratives and original reading content. We also introduce AEGIS (ACT English Item Generation System) which has been integrated to Sphinx as an essay-based item generation tool.

2.1 Automated Summarization

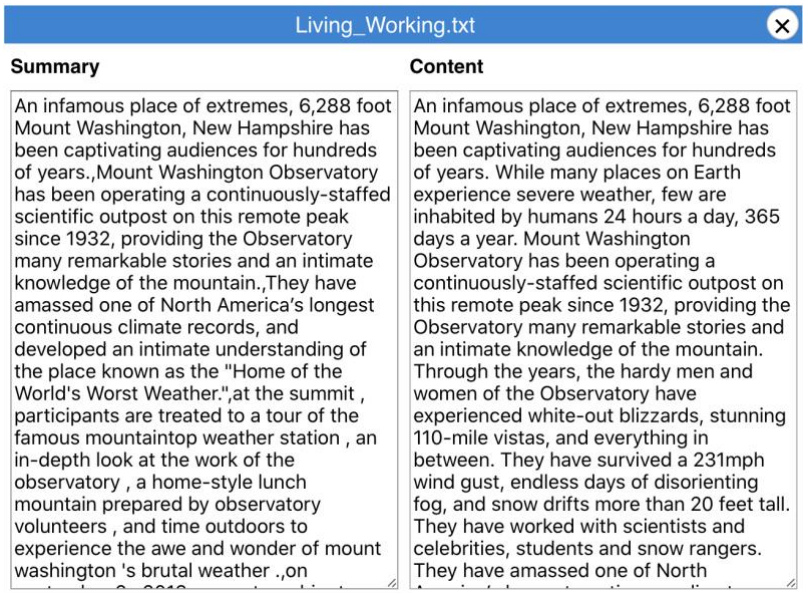


Figure 5. Generated extractive summary (left) and a source document (right) shown in Sphinx interface.

A key feature of Sphinx is automated text summarization (Figure 5). At the outset users can upload a variety of original articles that can then be readily summarized into prototype passages for faster understanding and even use as draft text for new compositions. Text Summarization is an area of Natural Language Processing (NLP) which is bound to have a huge impact on a lot of applications such as media monitoring, newsletters, social media marketing among others. In this project, we focus on

extractive method in which a shorter paragraph is created by extracting and concatenating a subset of spans (usually sentences) from a document, so that the summarized information is as close to the original text as possible. Let *a* denote an article containing several sentences [s*1*, s*2*, · · ·, s*m*], where s*i* is the *i*th sentence in the document. Our problem is defined as the task of assigning a label *y*i ∈ {0, 1} to each *s*i, indicating whether the sentence should be included in the summary. In our system, we adopted two extractive approaches. The first one is the most important early work and baseline for extractive summarization, named 'TextRank' (Mihalcea and Tarau, 2004), in which a graph-based ranking model similar to Google's PageRank (Brin and Page, 1998) was proposed to extract core sentences from text in real time. The second approach, BERTSum (Liu, 2019), is a state-of-the-art method which outperforms previous work on the CNN/Dailymail dataset (Hermann et al., 2015). BERTSum is fine-tuned on top of the famous BERT (Bidirectional Encoder Representations from Transformers) pretrained model (Devlin et al. 2019), which have recently advanced a wide range of natural language processing tasks. To apply BERT on text summarization, BertSum made the following changes to Bert: 1) Encoding multiple sentences in the input level by inserting a [CLS] token before each sentence and a [SEP] token after each sentence; 2) using interval segment embeddings EA (if *i* is odd) or EB (if *i* is even) to distinguish multiple sentences within an article; 3) On the sentence representation vectors Ti (the vector of the ith [CLS] symbol) of the top BERT layer, adding a linear classifier and using a sigmoid function to get the predicted score. After fine-tuning the pretrained BERT model on CNN/DailyMail news dataset, the BERTSum system is able to create high-quality extractive summarization for input articles.

2.2 Topic Modeling

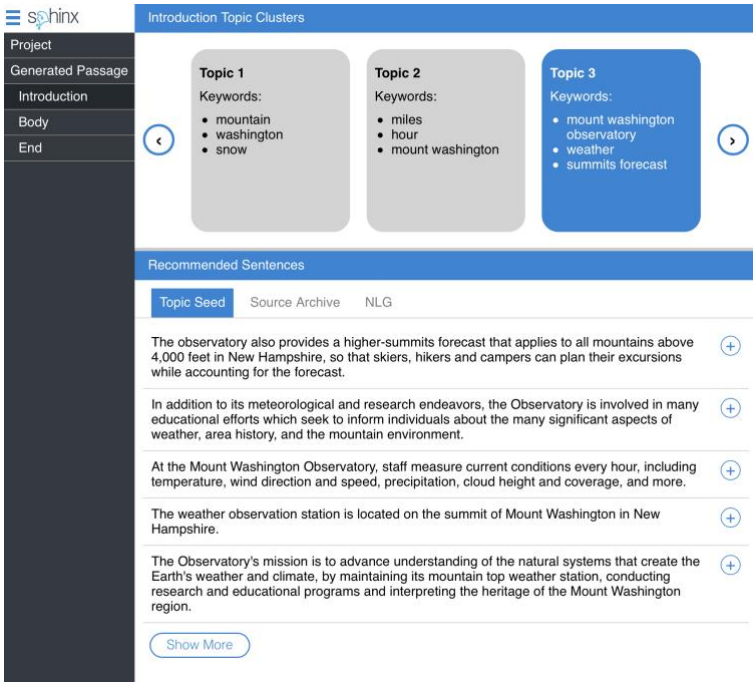


Figure 6. Sphinx processes all sentences of raw material documents to generate topic clusters with their keywords (above) and corresponding core sentences (shown as 'Topic Seed' below).

In addition to creating summaries, Sphinx analyzes the raw material articles/documents input by the user to automatically identify and extract latent topics and related core sentences. Since new articles in Sphinx

are composed at sentence level, we perform clustering on sentences of raw articles and infer topic phrases on formed clusters. These topics serve as a guidance for writers to ensure comprehensive content coverage and the starting point for new composition. Our topic modeling approach begins with utilizing Google's Universal Sentence Encoder (Cer et al. 2018) to encode the sentences of all input articles into a 512-dimensional vector. This transformer encoder was trained from a large corpus composed from a variety of data sources with the aim of accommodating a wide variety of natural language understanding tasks such as text classification, sentimental analysis etc. Second, K-means clustering is conducted only on the embedded vectors of those summarized sentences of all articles. To avoid the influence of the trivial description in the original articles on the topic modeling, we only perform the clustering on extractive sentences generated from the text summarization step. Third, the encoded vectors of all sentences are projected into computed clusters and only top sentences closest to each cluster center are kept, ranked and presented to users. At last, three unsupervised topic extraction methods are integrated into our system for users to choose from: graph-based method TopicRank (Bougouin et al. 2013), YAKE (Campos et al, 2020) and RAKE (Rose et al. 2010). Each of three approaches can be employed on top sentences of a cluster to extract key phrases which cover the major topics depicted in those clustered sentences. If the scaffold mode is enabled, the topic sentences are reranked so sentences from specific part (e.g. the introduction) of raw documents are presented to users at a higher priority.

2.3 Sentence Recommendation

Once the user selects a topic cluster, Sphinx recommends a list of sentences of that cluster (the quantity is user configurable) from which the user can choose a seed sentence, as shown in Figure 6. In article composition, we adopt an interactive and recursive strategy in Sphinx to integrate the writing skills of human users and language processing abilities of machine learning algorithms. For the second sentence of a new composition, besides selecting a new sentence from the topic clusters, users can also choose from sentences recommended by Sphinx to fit the content (Figure 7). The recommended sentences could come from archived sentences of source documents (shown as 'Source Archive' in this figure) or new sentences generated from the GPT2 (Radford et al., 2019) model (shown as 'NLG' in this figure). This composition process then repeats until the user is satisfied with the content of the passage draft. Sphinx provides three recommendation engines, of which the first method has lower time complexity, the second method returns higher recommendation accuracy from achieved source sentences, while the third method creates new sentences not from source documents. All three engines evaluate contiguity and cohesion to filter down the next sentence recommendation list to a user configured number, which ensures semantic meaning carries over in sentence transition. In the first engine, embedded vectors of all sentences are computed from Google's Universal Sentence Encoder and an affinity graph is created using cosine similarity measures. Based on the undirected manifold graph ranking algorithm (Zhou, 2004), the sentence recommendation problem is formulated as a sentence ranking problem given the query sentence (the last sentence in the composed new article). The second recommendation engine is based on a pretrained BERT (Devlin et al., 2019) model with the next sentence prediction objective, which was trained on pairs of sentences from a variaty of datasets to specifically model the relationship of two input sentences - whether they are next to each other. In a transfer learning setting, we get rid of the output layer of binarized next sentence classification and utilize the last hidden layer features of tokens of two sentences in a pair: (S_e and S_i where i = 1, 2, 3 … p). In this way, the problem of sentence recommendation becomes ranking the

similarities between the BERT features of *S*e and *S*i. Instead of simply averaging on all token features before computing the cosine similarity of Se and Si, we calculate a weighted sum of token features for each sentence according to the frequency of a word: less frequent words contribute more to sentence feature than frequent words (e.g., is and do) and more unique relatedness could be captured between two sentences. In the third recommendation engine, the famous language generation model GPT2 (Radford et al., 2019) is applied to generate brand new sentences different from archived source articles. The authenticity and language quality of GPT2 generated sentences will be evaluated by content specialists before adoption.

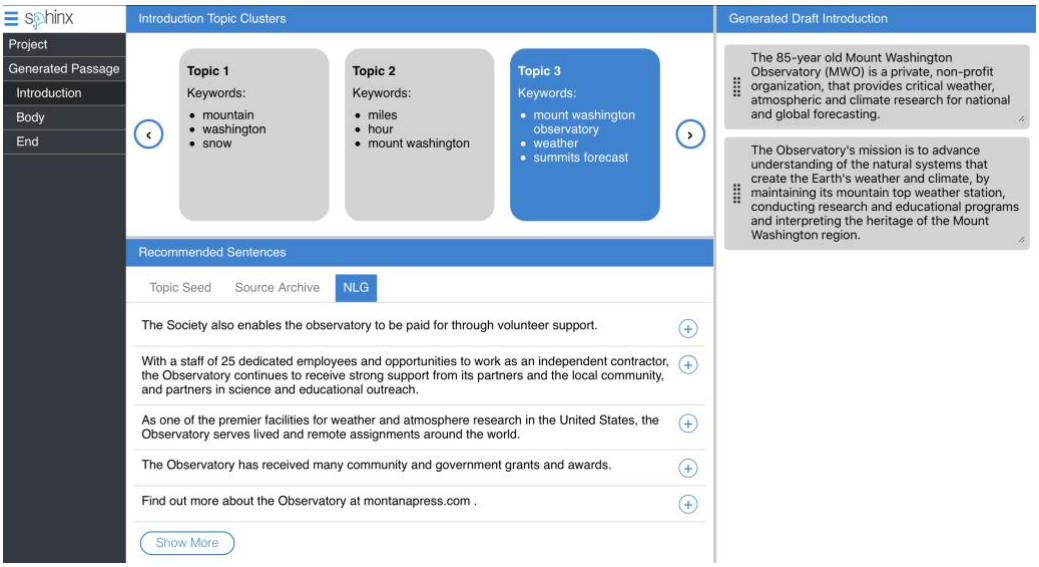


Figure 7. The recommended sentences could come from archived sentences of source documents (shown as 'Source Archive' in this figure) or new sentences generated from the GPT2 model (shown as 'NLG' in this figure).

2.4 Paraphrasing

At any given moment in draft composition, users have the option to use automated paraphrasing. Our sentence paraphrasing approach consists of three steps. First, a Named Entity Recognition and Discourse elements (NERD) filter masks all segments of the sentence that need to be kept intact, such as quotations and names of entities. Second, we use a back-translation approach whereby the sentence is translated to *n* different languages and then translated back to English using Google Translate (https://translate.google.com/) to generate paraphrase candidates. Third, each of the *n* sentence candidates are scored for their semantic similarity (USE cosine distance) and lexical-grammatical distance (Rouge Score, Conroy et al. 2006) with respect to the original sentence. A weighted average of these scores is used to rank list the paraphrased sentences and the user has agency to make a final selection. After all composed sentences are paraphrased, the user can continue to edit essays they work on before saving them (Figure 9).

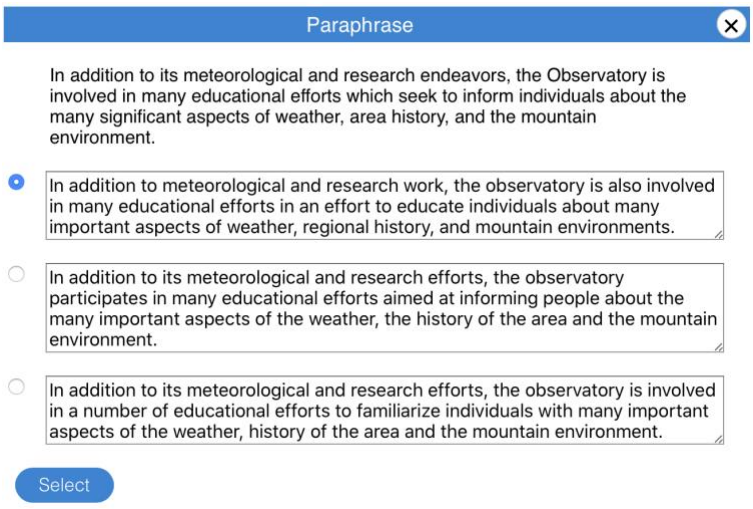


Figure 8: Sphinx's paraphrasing module based on back-translation.

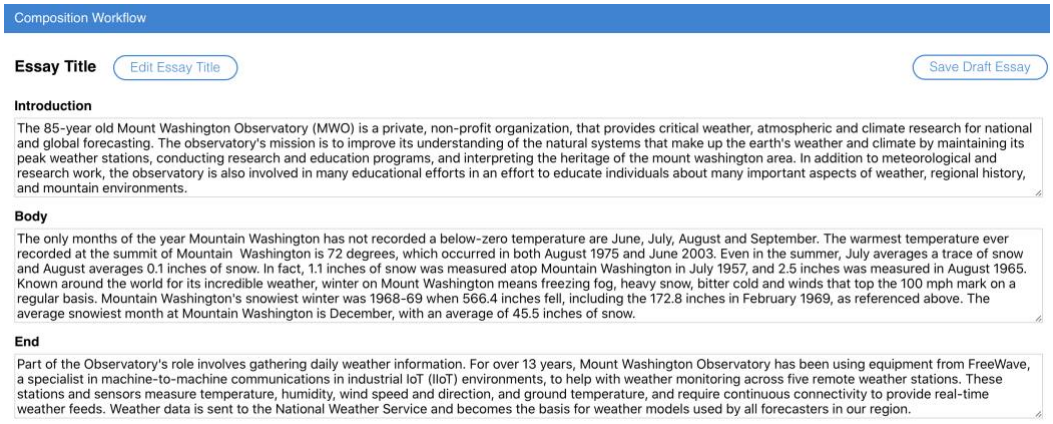


Figure 9: An example of a draft composition. Users can continue to edit essays they work on before saving them.

2.5 Automated Item Generation

Rapidly evolving new directions in computer-adaptive tests with increased numbers of forms, and expanded markets, require a significantly higher level of item production. Automated item generation is a promising avenue for facilitating item development, though it has traditionally been limited to math content. Initial attempt of automatic English reading item generation at ACT focused on discrete item generation, which relied on content staffs to produce item text substrings that would continue to make sense in the context of the newly generated items. Recently, AEGIS (ACT English Item Generation System) has been created and successfully applied in the development of various ACT English Tests. In AEGIS, the possible components (e.g., the nouns, names, verbs, adjectives) of the generated items are not manually derived by content experts but are scanned for in the essay and transformed in rule-based ways to generate the item's newly inserted essay linguistic error and item distractors. For example, Figure shows an essay excerpt and corresponding item with the item model classification SST-FOR-FRG, "Correcting rhetorically ineffective sentence fragments":

Architect Eero Saarinen, who created the design
that symbolized the memorial's theme of St. Louis
as the "Gateway to the West."

A. NO CHANGE
B. Saarinen, creator of
C. Saarinen created*
D. Saarinen creating

Figure 10: An example of essay excerpt item which is classified under the item model SST-FOR-FRG, "Correcting rhetorically ineffective sentence fragments".

The comma and the "who" do not belong in the underlined portion of the essay, so the key is "C". As of the end of the year 2019, over 200 item models has been developed and put in operational usage; over 1000 items have been produced for a number of ACT English tests.

Since the item model is always based on the linguistic patterns, errors, and rules characterizing the abstract structure of a parent item of known high quality, the need for content expert involvement is minimal – the software automatically handles all of the actual item generation. For the same reason, AEGIS is particularly suitable to be integrated into large-scale content generation systems like Sphinx. As shown in Figure, AEGIS can be applied on both the source documents input by users or the newly-composed articles to generate and publish supported item types in real time, as shown in Figure 11.

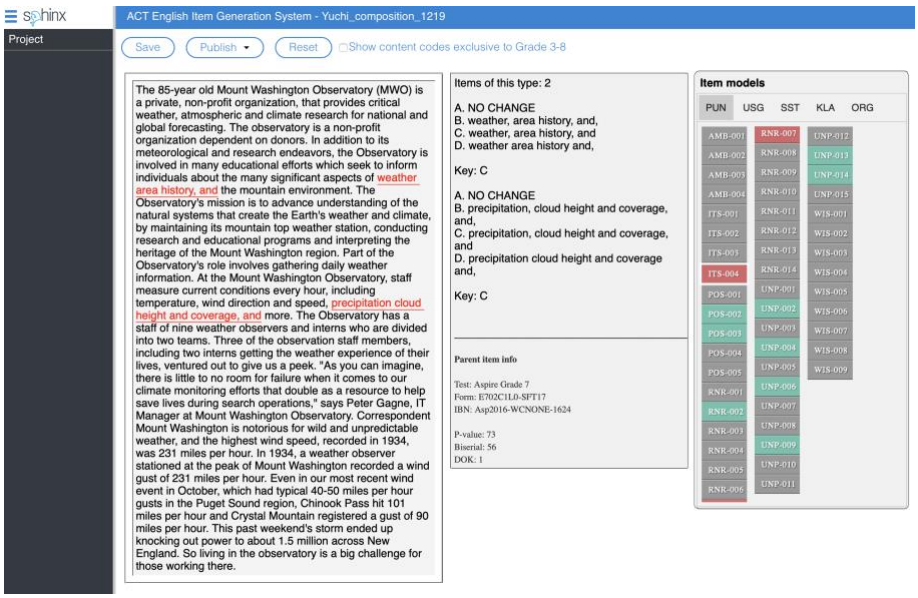


Figure 11: In AEGIS integrated into Sphinx, two items are generated under the item model 'ITS-004'.

3 CONCLUSION

In this paper we present Sphinx, a system that in a scalable and efficient manner helps writers compose English reading comprehension passages and corresponding items for use in educational

learning and assessment. We adopt an interactive and recursive strategy to integrate writing skills of human users and advanced NLP modules deployed in an auto-scaled manner. One of the benefits of our approach is that the human in the loop enables the AI to learn from expert writers as it accumulates useful data for future training of models, so make the large-scale learning and assessment more efficient, more effective and more adaptive. We believe the system can be used in delivering adaptive learning experiences as well as formative and summative assessments of English reading among other educational applications. Sphinx is currently in pilot trials and validity testing and we plan to introduce more advanced functionalities such as intelligent source document searching and organizing, structured article composition and automated figure generation among others.

REFERENCES

Campos, R., Mangaravite, V., Pasquali, A., Jatowt, A., Jorge, A., Nunes, C. and Jatowt, A. (2020). *YAKE!: Keyword Extraction from Single Documents using Multiple Local Features*. In Information Sciences Journal. Elsevier, Vol 509, pp 257-289

Cer, D., Yang, Y., Kong, S. Y., Hua, N., Limtiaco, N., John, R. S., & Sung, Y. H. (2018). *Universal sentence encoder*. arXiv preprint arXiv:1803.11175.

Chen, D., Bolton, J., & Manning, C. D. (2016). *A thorough examination of the cnn/daily mail reading comprehension task*. arXiv preprint arXiv:1606.02858.

Conroy, J. M., Schlesinger, J. D., & O'Leary, D. P. (2006, July). *Topic-focused multi-document summarization using an approximate oracle score*. In Proceedings of the COLING/ACL.

Cervone, A., Khatri, C., Goel, R., Hedayatnia, B., Venkatesh, A., Hakkani-Tur, D., & Gabriel, R. (2019). *Natural Language Generation at Scale*. arXiv preprint arXiv:1903.08097.

Gierl, M. J., Lai, H., & Turner, S. (2012). *Using automatic item generation to create multiple choice items for assessments in medical education*. Medical Education, 46, 757-765.

Heilman, M. 2011. *Automatic factual question generation from text*. Ph.D. thesis, Carnegie Mellon University.

Liu, Y. (2019). *Fine-tune BERT for Extractive Summarization*. arXiv preprint arXiv:1903.10318.

Mihalcea, R., & Tarau, P. (2004). *Textrank: Bringing order into text*. In Proceedings of the 2004 conference on empirical methods in natural language processing (pp. 404-411).

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). *The PageRank citation ranking*. Stanford infolab

Radford, Alec and Wu, Jeff and Child, Rewon and Luan, David and Amodei, Dario and Sutskever, Ilya. *Language Models are Unsupervised Multitask Learners*. Technical report, OpenAi, 2019

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). *Squad: 100,000+ questions for machine comprehension of text*. arXiv preprint arXiv:1606.05250.

Rose, S., Engel, D., Cramer, N., & Cowley, W. (2010). *Automatic Keyword Extraction from Individual Documents*. In M. W. Berry & J. Kogan (Eds.), Text Mining: Theory and Applications: John Wiley & Sons.

Serban, I. V., García-Durán, A., Gulcehre, C., Ahn, S., Chandar, S., Courville, A., & Bengio, Y. (2016). *Generating factoid questions with recurrent neural networks*: arXiv:1603.06807.

Wolfe, J.H. 1977. *Reading retention as a function of method for generating interspersed questions*. DTIC, 1977

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). *Aligning books and movies*. In Proceedings of the IEEE ICCV (pp. 19-27).

Zhou, D., Weston J., Gretton A., Bousquet O., and Schölkopf ,B. *Ranking on Data Manifolds*. NIPS 2004.