A Machine Learning Approach to Detecting Start Reading Location of eBooks

Sravan Babu Bodapati*, Sriraghavendra Ramaswamy†, Gururaj Narayanan†
*Amazon, Seattle, WA
sravanb@amazon.com
†Amazon Development Center, India
{sriragha, gururan}@amazon.com

***Abstract*—Machine Learning and NLP (Natural Language Processing) have aided the development of new and improved user experience features in many applications. We address the problem of automatically identifying the "Start Reading Location" (SRL) of eBooks, i.e. the location of the logical beginning or start of main content. This improves eBook reading experience by taking users automatically to the logical start location without requiring them to flip through several front-matter sections such as "Dedication" and "About the Author". Automatic identification of SRL is complex since many eBooks do not adhere to any well-defined convention with respect to section naming, formatting and layout patterns. We formulate SRL as a classification problem based on detailed rule-based and NLP-based classification schemes. Our models are being used in production for Kindle eBooks and have led to a 400% increase in coverage (number of books which had SRL stamped) compared to what could be achieved earlier through an entirely manual process, while also maintaining a high accuracy of 95%.**

I. Introduction

Providing a uniform user experience in an eBook offering is difficult due to lack of standardization in organizing, formatting and layout of content. For instance, readers of books typically skip a lot of front matter section such as "Dedications", "About the author", "Publishers note" etc, since they are not related to the mainline content of the book. Users of physical books could easily skip the front matter section due to the large form factor of books and ease of skimming though pages. With the rise of eBook offerings such as on Kindle and on mobile environments with small displays, navigating to the logical beginning of the e-books is a tedious process for the user. Hence, e-book offering services such as Kindle would want to mark the logical beginning (aka Start Reading Location, or SRL in short) of the eBooks to improve the reading experience for the users. During the early phase of Kindle, SRL was identified and set manually by auditors. However, with the rapid increase in the number of digital books, scaling this manual process to handle all books has become infeasible. We address the problem of automatically identifying the SRL for e-books in Kindle. For example, looking for a preview or downloading this very famous eBook [1] automatically opens at the SRL location, providing a much better reading experience.

The problem of automatically identifying SRL is challenging due to: (1) Lack of Standardization: In the past, typically organizing, formatting and layout standards were defined and enforced on authors by publishers. However, with the rise of internet and self publishing services [2], authors follow their own formatting standards. Also, there is no standardization of writing structure for books across publishers. (2) Missing Information: There are several popular e-books with basic section heading missing. Also, few authors use non-standard tools to generate / convert e-books which might remove valuable metadata information during generation / conversion. In this paper, we propose the following steps to identify SRL: (1) segment the eBook text into logical blocks (2) build a binary classifier to identify each block as "relevant" or "not relevant" (3) the beginning of the first relevant block is marked as SRL of the book. We describe NLP based classification scheme for identifying blocks of content as "relevant" or "not relevant" and compare it against the rule based existing production baseline scheme. Our classifier shows an 400% increase in coverage at 92% accuracy compared to an earlier rule-based system on random audit data and has been successfully deployed in production.

The remainder of the paper is organized as follows. Section II provides an overview of our automation pipeline and our techniques for extraction/identification of logical blocks. Sections III and IV explain the architecture of our 2-step classification scheme and the models used by the two classifiers. Section V explains the challenges we faced while developing the classifier models and the importance of various features. We present our experiments and results in Section VI, survey prior work in Section VII, and finally talk about Future work in Section VIII.

II. Overview of Automated SRL Detection

Our automated SRL identification system is illustrated in Figure 1. It involves four major steps each of which is outlined in the diagram.

A. Automated SRL Detection as a classification problem

We model SRL detection as a classification problem. Our system divides an eBook into "logically separable" blocks, or *LBCKs*, each of which is to be classified as either *BEFORE_SRL* or *AFTER_SRL*. The first LBCK from the beginning of the book that gets classified as *AFTER_SRL* is chosen as the SRL of the book, subject to additional requirements on the confidence of classification. We identify the Table of

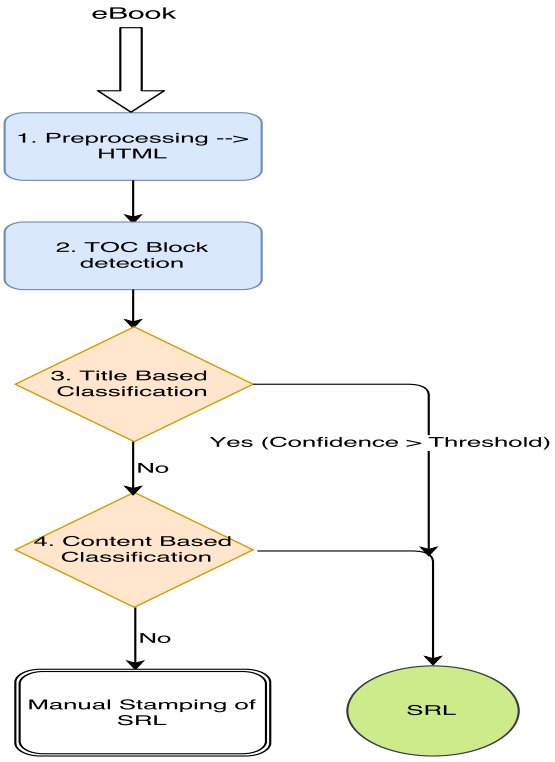

Fig. 1. SRL Automation Flowchart

contents (TOC) LBCK first and use the information about TOC LBCK to classify all the other LBCKs. Our system uses two different predictors - a "title text" based predictor and a "content text" predictor in a twostep classification scheme. Both predictors use language-dependent classification models. A cascading approach is used for prediction (ref. Figure 1). The title text based predictor is called first to classify an LBCK. If its result has a low confidence, content-based based prediction is attempted and its prediction is chosen as the SRL if its associated confidence score exceeds a threshold. If both classifiers fail to classify blocks with an acceptable degree of confidence, the eBook is queued for manual setting of SRL by human auditors. The two-step classification makes the system more robust and less prone to errors in identifying the SRL. We had the following choice while building the models: (a) use a Deep Neural Network (DNN) based architecture with handcrafted features derived from title text and content text fed to a single neural network model (b) use two separate classifiers, each built exclusively based on the information from title text and context-text. We chose (b) as it gives each of the models a fairly exclusive view of the input data and hence held promise for achieving a lower variance in prediction with less effort compared to (a).

B. Block Extraction

This step converts an eBook into a list of LBCKs, each of which corresponds to a distinct logical section. An example of LBCK is the content of the "Preface" or that of the "Chapter 1" section[1]. The eBook is first converted to HTML format. This is followed by extraction of an ordered list of text blocks along with their section/sub-section headings, if present. Customizable rules based on HTML tags are used for block identification/extraction. For e.g., a block may correspond to all text content between a start header tag and the corresponding close tag, or content from start of a header till a pre-defined end phrase is encountered. Each block is associated with following attributes in HTML format:

- *title_level* (level of the block's section header as in HTML, such as *h1*, *h2*),
- *title_text* (text of the block's section header. such as "Preface", "Chapter 1"),
- *title_font* (font size of the block's title text),
- *content_text* (text of paragraphs/lines within the block; not to be confused with title text)
- *list_of_images* (referring to images in the block)

Due to the use of irregular formatting by authors, or missing formatting/layout information in the source file of eBook, some books end up with a very poor representation in terms of HTML blocks - for e.g., not having headers, or all content text of a section incorrectly getting parsed into the title text. The block extraction algorithm was designed to be robust enough to handle correctly most such cases.

C. Identification of TOC block

A majority of the books tend to have their TOC near the beginning, but there are several books that have their TOC at the end or don't have a TOC page at all. We noticed a strong correlation between the relevance of a block to the main content, and its relative position w.r.t. the TOC as well the location of the TOC within the book. For e.g., if the TOC is near the beginning of the book, most of the blocks before the TOC tend to be irrelevant to main content and hence would come before the ideal SRL. This motivated us to identify the TOC LBCK in the eBook. We use the following algorithm to identify the TOC LBCK in an eBook.

Algorithm 1 TOC Identification

Input list of LBCKs
Output TOC LBCK
titleTexts $\leftarrow \{bk.titleText$ **for** bk **in** *list_of_LBCKs*$\}$
$nbck \leftarrow numberOfElementsIn(list\_of\_LBCKs)$
for $(currentLBCK$ **in** $list\_of\_LBCKs)$ **do**
  $contentText \leftarrow currentLBCK.contentText$
  $lines \leftarrow split(contentText, newline)$
  $frac \leftarrow match(lines, titleTexts)/nbck$
  if $frac \geq 0.65$ **then**
    $TOC \leftarrow currentLBCK$
    return TOC
  end if
end for
return null

[1]Refer to Appendix for a detailed example of LBCK

The algorithm labels an LBCK as TOC if its content text matches at least 65% of the title text of all LBCKs in the book. The match threshold was chosen empirically; a value of 0.65 minimized false positives and gave TOC identification an accuracy of 0.97.

The following TOC-based features are used in both title-based and content-based classifiers: (a) whether the block is before the TOC block or after the TOC block (b) categorical variable that indicates whether the TOC block is at the beginning of the book, near the end of the book or is absent (c) difference/offset between location of the block and location of the TOC block. We now move onto the task of labeling other LBCK's as relevant(BEFORE_SRL) or irrelevant(AFTER_SRL).

III. Title Text Based Classification (TBC)

In title text based SRL prediction, an LBCK is classified based on information from title text and TOC block alone – the content text of the LBCK is not used. The title text based classifier consists of a rule-based classifier and a machine-learnt classifier.

Rule-based classifier: This classifies a block based on whether the title text matches some predefined keywords or key-phrases which were chosen based on empirical data. A block is classified with a confidence score of 1.0 if the title text matches any of the keywords or pre-defined regexes. The algorithm moves on to the next block for classification if the current block was classified as *BEFORE_SRL*, or tags the current block as SRL and exits if it was classified as *AFTER_SRL.*

Machine-learnt classifier: This is triggered if the rule-based classifier encounters a block that cannot be classified as it doesn't match any of the pre-defined rules. A simple and efficient baseline for ML classifier is a Naive Bayes [3] or a RandomForest [4] classifier built from the bag-of-words (BoW) features and some features that are largely derived from title text, such as (a) presence of any number on the title block, since chapter headings often have numbers, while Preface, Acknowledgement etc. doesn't have numbers in general (b) number of words in the title, (c) whether the block is before the TOC block or after, (d) whether the title matches any line in the content of the TOC block, (e) length of title text and (f) fraction of characters in each of font types, upper case, lower case.

We found that we could achieve a significant improvement over the baseline accuracy by using an ensemble of models (shown in Figures 2 and 3). Figure 2 illustrates a model architecture based on LSTM [5] with pre-trained GloVe [6] embeddings as input. The output of BiLSTM [7] is fed to a dense layer along with other handcrafted features as explained above. Pretrained embeddings successfully captured the relationship with target as most of the words and phrases that belonged to relevant or irrelevant category had huge semantic overlap and are synonymous to each other. The weights in embedding layer are frozen and not updated during training. We picked GloVe embeddings of vector size 300 dimensions.

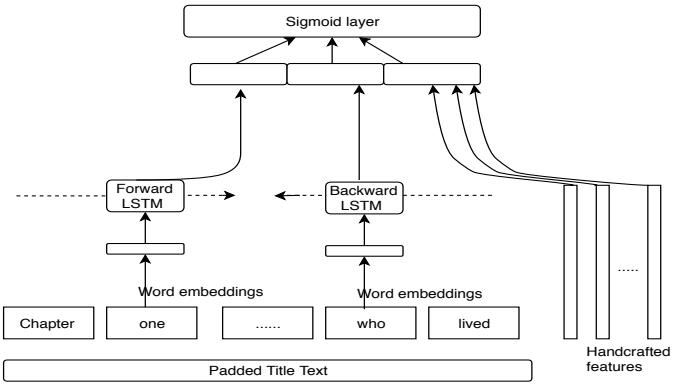

Fig. 2. BiLSTM based sharearchitecture for Title Based Classification

We input a fixed length sequence to the embedding layer by either truncating or padding the input with zeros. The length of input is chosen as 99 percentile of the title text, which came out to be 108 from our training data. The padded/truncated document with embeddings is fed into an BiLSTM layer with 64 units. The output from BiLSTM layer is then concatenated with other title text based features in a dense pre-final layer.

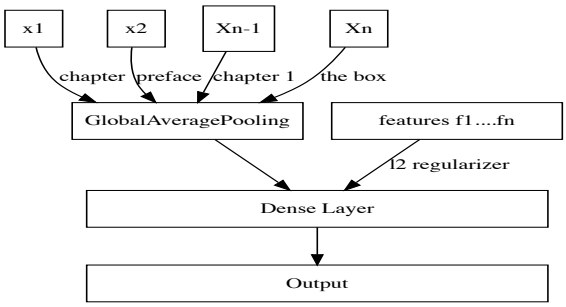

Fig. 3. FastText based architecture for Title Based Classification

Figure 3 illustrates the second model of our ensemble, whose architecture is inspired by fastText [8]. Similar to fastText paper, we extract unigram and bigram tokens from the title text and assign a unique embedding for each of those. Higher order n-grams help in capturing the local word order. These embeddings are then averaged using GlobalAverage-Pooling layer, which gives us a representation of the title text of the document for this classification task. The embedding layer isn't frozen – the embeddings for unigrams and bigrams are learnt during the training phase. The next layer has the input from GlobalAveragePooling layer along with other title-based features as explained in (a),(b),(c),(d) above. We use the sigmoid function to compute probability distribution over the classes.

The final classifier we use for TBC is an ensemble of the above two classifiers. We average the probability distributions from the two classifiers to get the ensemble TBC's probability

distribution over classes. We implemented both the models using keras[9] interfaces, with tensorflow[10] as the backend.

IV. Content Text Based Classification (CBC)

Content-based classification uses features derived from the content text of the block as opposed to the title text of the block. These comprise of several length-based features, named-entity (**NE**) based features, casing and font based features. The features are enumerated below:

- Length based features
  1) Number of words in content
  2) Number of lines in content
  3) Average Number of words per line
- Casing, Font and TOC Based Features
  1) Fraction of text that is upper-cased
  2) Fraction of lower cased letters
  3) Number of quotations in the text
  4) Proportion of text in the content that is quoted (indicates dialogues, speech)
  5) Number of fonts used in the text
  6) For each font, fraction of overall text using that font
  7) Is block before or after the TOC block
- NER Based features
  1) Number of named entities in the block (i.e. in the blocks content text)
  2) Number of people, place and organization category named entities in the content text
  3) Number of matches with top 10%, 20%, 30% and 40% named-entities in the book
  4) Ratio of matched count to overall occurrence count for named entities of types people, place and organization

We use Stanford CoreNLP NER models [11] for extracting named entities as the performance of these NER models on our e-book corpus was found to be very satisfactory. We restrict our named entity set to people, organizations and places. We use two classifiers built from the features enumerated above – a Random Forest classifier and an XGBoost [12] classifier. The outputs of these two models are combined with a weightage of 40% for RandomForest and 60% for XGBoost model to get the final output. These weights were determined empirically by building a logistic regression model on the outputs of the above classifiers.

V. Data Preparation and Training

We used two years worth of manually set SRL data to create training, cross-validation and test sets, in a split ratio of 70:15:15 respectively. This dataset had close to 35000 eBooks across all the languages.

A. Imbalanced class distribution

The average number of LBCKs per eBook when computed on 25000 eBooks from the training set is 19, whereas the SRL block on average is the fourth block. Intuitively, this makes sense as the front-matter typically has fewer number of text blocks than the rest of the book. Once the LBCKs are extracted, we can clearly see that including all the LBCKs after the manually-set SRL block as *AFTER_SRL* would result in huge skew of the class distribution, which would affect the performance of our classifiers. In order to mitigate the effects of this data skew and get an even distribution of *BEFORE_SRL* and *AFTER_SRL* blocks from every book, We started our experiments by picking as many blocks after the SRL LBCK as there are before the SRL LBCK. This resulted in our algorithm missing few important blocks and stamping the SRL after them, leading to destructive errors (Please refer to Appendix for explanation of these error types). We then tried constructing training data by picking 5 consecutive blocks after SRL block as AFTER_SRL and all blocks before SRL block as *BEFORE_SRL* for every eBook. We noticed that the classifier built from this data resulted in a much lower number of destructive errors. The value of 5 was chosen as it corresponds to one plus the average SRL block index of our training corpus. The final dataset we obtained is a relatively balanced dataset with classes in the ratio of 0.55:0.45, in favor of *AFTER_SRL*.

B. Feature Importance

This section looks in detail about informative features with high predictive power for each of the classifiers.

1) Title text based classifier: We computed Chi-square metric to find words that were highly relevant/useful to our title text based classifier. Table V-B1 shows the top-ranking words based on Chi-square test for each class.

AFTER_SRL	BEFORE_SRL
Chapter, Part	cast-of-characters, translator's note. Overview
1,2,3,4,9	2015,2014
Prologue, Incidents	Foreword,Fiction, Characters,coincidental
How to use	Imaginations, Rights,reserved
Introduction, Background	preface,Events, Publisher,published
Epigraph	author,remarks,alsoby
Episode, Section	Resemblance,places, persons,disclaimer
Ingredients	Acknowledgements, Praise,dedications

TABLE I
DISTINGUISHING WORDS IN CHI SQUARE TEST

Other important features from title text based classifier based on feature importance from RandomForest classifier are: (a) fraction of characters in each font types. (b) fraction of characters in upper case.

2) Content text based classifier: We studied the significance of various features based on the feature importance from RandomForest classifier. The following features were reported to be the most important features: (a) Average number of words per line, (b) Named Entities percentage match with top 20% named entities in the book, (c) Ratio of quoted characters to overall characters, (d) Number of lines in the content text,

(e)Count of upper case characters to overall characters in the book. We found

$$\frac{count(NE\ in\ current\ block)}{count(Top\ 20\%\ NE\ by\ count\ in\ entire\ book)}$$

to be a very informative feature in discriminating AFTER_SRL LBCKs'. It had a very strong positive correlation with the AFTER_SRL label. Similar features like match with the top 40% NE's didnt have such high predictive power or information gain. Few books had very few unique NEs with most of those NEs occurred in the "Also By" or "Dedications" sections that belonged to pre-SRL content. This led to lot of noise and irregularities in the match with top 40% NE's feature. Restricting the match to top 10% NE's of the book also could only help capture NEs corresponding to the protagonist and characters around the protagonist, but these also occurred in the "summary" section at the beginning of the book (i.e. pre-SRL). Match with top 20% turned out to be just right.

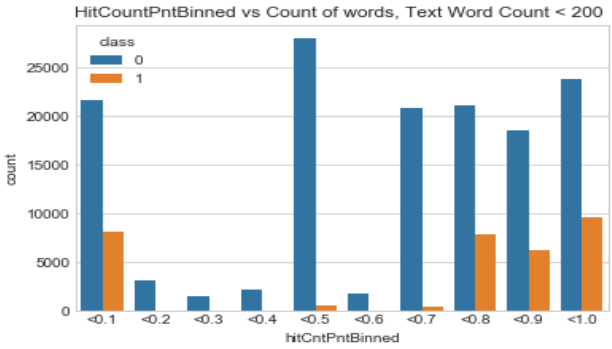

Fig. 4. Class-distribution in each bin of fraction match with top 20% NE, Word count less than 200

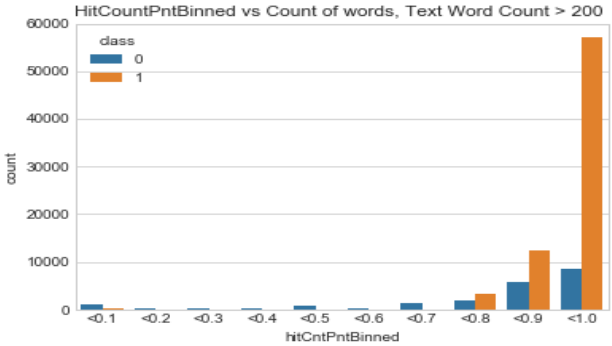

Fig. 5. Class-distribution in each bin of fraction match with top 20% NE, Word count greater than 200

Figure 4 shows distribution of classes for each bin of 20%-NE-match when the text word count is less than 200, and Figure 5 shows the distribution when text word count is greater than 200. The negative class had lesser text word count and less match compared to positive class which had large text word count and higher NE match. These two features together were very informative and gave our content text based classifier the ability to capture many blocks that were tagged unconfident by our title based classifier, especially the books which didn't follow proper naming conventions for the title text. This helped us greatly in enhancing the coverage of our automation efforts.

VI. Experiments and Results

In this section, we evaluate each of our classifiers based on their performance, along with the coverage obtained at different stages of automation. We also look at the agreement between the classifiers and the confidence threshold for the classifiers. Even though the classifiers have been observed to be doing extremely good at the block level, we can note that a single misclassification at the block level would render the SRL stamped for the book inaccurate. We hence tried to optimize the confidence thresholds for both the classifiers based on accuracy of the SRL stamped at the eBook level rather than accuracy at the block level.

A. Baseline

We choose the RandomForest model built from the title text based features as the baseline, and compare performances of our final TBC and CBC with reference to this. The ROC AUC value for this classifier is 0.82, which we will be using as a baseline.

B. Performance of title text based classifier and confidence-threshold

Figure 6 shows the ROC comparison of final ensemble of FastText and LSTM based classifier vs the baseline model. We can clearly see that the ensemble model is learning the relationship with target better than the baseline. Figure 7 shows

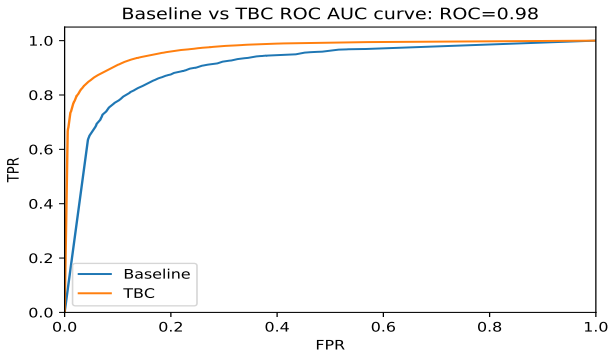

Fig. 6. Baseline vs TBC ROC AUC Comparison

the accuracy of the SRL stamped in eBooks with varying the confidence threshold needed that is required to accept the label from the TBC. We wanted an accuracy of above 0.91 for the SRL stamped, which corresponded to a confidence threshold of 0.76. When the probability is below the confidence threshold, we call CBC for tagging the label of the LBCK. We were able to achieve a coverage of 40% for the SRL on the eBook catalog by using just the TBC.

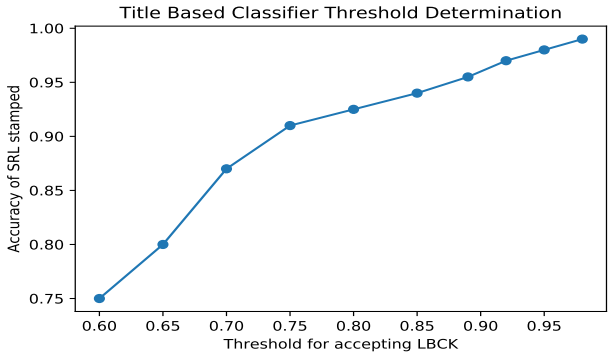

Fig. 7. Baseline vs TBC ROC AUC Comparison

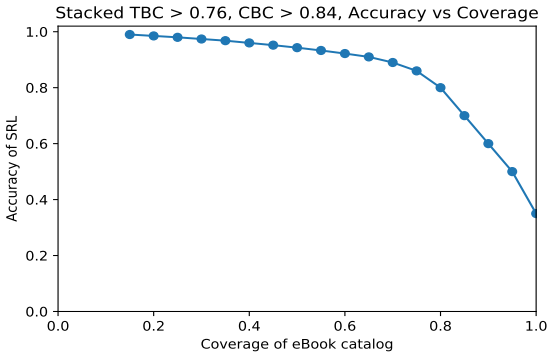

Fig. 9. Coverage vs Accuracy for Stacked Classifier

C. Performance of Stacked classifier

Figure 8 shows the ROC comparison of our CBC and TBC models. Though the two individual classifiers have a fairly similar performance in terms of AUC, we gained a huge improvement in coverage of books by stacking together the two classifiers. The confidence threshold for CBC was arrived

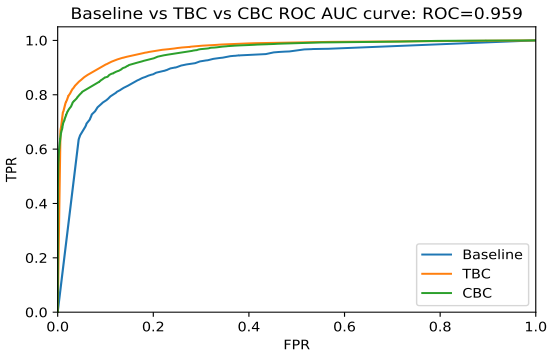

Fig. 8. TBC vs CBC ROC AUC Comparison

at based on the performance of the stacked classifier, i.e. we called CBC for every block whose TBC confidence is less than 0.76 and checked the accuracy at different values of confidence threshold for CBC. For the same accuracy of 0.91, we had to choose a confidence threshold of 0.84. This stacked classifier was able to achieve a coverage of 63% on the eBook catalog.

Figure 9 shows the coverage vs accuracy at various thresholds for the TBC vs the stacked classifier (stacked clasifier uses TBC's output if *confidence_score(TBC)* $\geq$ 0.76, else use CBC if *confidence_score(CBC)* $\geq$ 0.84).

D. Kappa statistic

We computed Kappa statistic [13] to compare the agreement between the two classifiers. Only the blocks that had TBC confidence $\geq$ 0.76 and CBC confidence $\geq$ 0.84 were picked for this analysis. As shown in Table VI-D, both the classifiers had kappa statistic value of approximately 0.89045. This added

	Title Based AFTER_SRL	Title Based BEFORE_SRL
Content Based AFTER_SRL	98992	5841
Content Based BEFORE_SRL	5050	89757

TABLE II
AGREEMENT BETWEEN CLASSIFIERS

more weight to our hypothesis that both our classifiers were able to capture the relationship at the block level quite well.

E. Performance of Automation

Figure 10 shows the overall success rate of our algorithm (considers all languages for which automation is enabled) for a period of over 2 weeks in April-May, 2018. The average success rate during that period is 86.5%. Weekly audits have shown the accuracy of auto-generated SRL to be approximately 90.5%.

The models were initially developed only for English language books, we were able to extend the same approach easily to German language books with excellent results (outlined below). For French, we couldn't get a highly accurate Named Entity Recognizer; hence we combined features relating to content text and title text and built a single classifier. We adopted the same approach for Italian as well. Table VI-E provides the observed coverage for various languages for each of the classifiers.

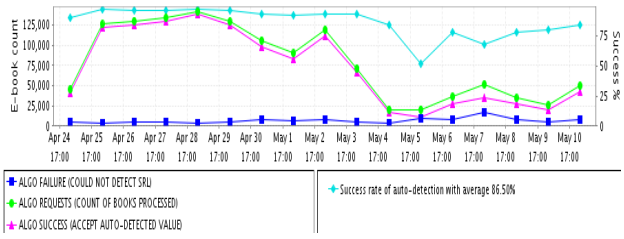

Fig. 10. Success rate of SRL auto-detection on all languages combined

German books had the highest coverage of SRL with accuracy of SRL at 95.4%, followed by English, with an accuracy of about 90.3%.

	en	de	fr	Overall
Baseline	29.1	33.2	30.1	31.6
TBC	41.1	53	48.1	45.2
Stacked Classifier	63.6	71.2	48.1	64.4

TABLE III
COVERAGE ATTAINED WITH EACH LEVEL OF CLASSIFICATION FOR DIFF LANGUAGES

VII. Related Work

Irrelevant content filtering techniques ([14]; [15]) try to find relevant content or filter out all unrelated content with reference to a query. Generating the summary for the entire eBook using multi-document summarization techniques, that can be used with above techniques [16] it'self is very hard and error prone. A minor miss in capturing the summary can result in a huge error in the final SRL. Other solutions like identifying and naming section boundaries in large text documents ([17]; [18]) wouldn't entirely solve our problem,as our problem goes much beyond identifying section headings. This work can be used in future though to expand our scope in extracting logical blocks, particularly for books that couldn't be rendered perfectly with html parsing, and therefore couldn't be classified by our algorithm. Topical segmentation techniques [19] can help to an extent in identifying the clusters of related paragraphs together, but it fails to provide a clear understanding about each section's relevance to central story-line of the book. There might be few sections like "summary" which are related to main content but these sections could be skipped as front-matter. While NLP is a well-established field, we are not aware of any prior work that addressed this specific problem in the context of eBooks.

VIII. Conclusion and Future Work

Our work demonstrates how ideas from NLP, with the aid of manually-labelled SRL data for eBooks from the past, was used to train classifiers that mimic the manual intelligence that goes into identifying the starting location for the main content of an eBook. This problem requires some understanding of the semantics behind the sectioning/chaptering structure of any eBook and is challenging because books may come from any genre, any author, and often do not follow any standard conventions with respect to styling, section-naming etc. We have also presented a novel way to frame the SRL identification problem, without which using any ML would have been really difficult. Future work should aim to extend our work to books in other non-European languages and increasing the coverage further without compromising on the accuracy. The overall coverage of Automation for English and European languages now sits at 65% and can be enhanced further to cover more than 90% of the catalog. We can also revisit our logic of stacking the classifiers to use a single layer neural-net based approach that could capture the relationship among the classifiers and target better.

References

[1] J. K. Rowling, "Harry Potter and the Sorcerer's Stone (Kindle Edition)," https://www.amazon.com/dp/B0192CTMYG.

[2] "Amazon Kindle Direct Publishing," https://kdp.amazon.com/en_US/.

[3] H. Zhang, "The optimality of naive bayes," *AA*, vol. 1, no. 2, p. 3, 2004.

[4] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[7] C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith, "Transition-based dependency parsing with stack long short-term memory," *arXiv preprint arXiv:1505.08075*, 2015.

[8] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

[9] F. Chollet *et al.*, "Keras," 2015.

[10] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[11] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit," in *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 2014, pp. 55–60.

[12] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016, pp. 785–794.

[13] J. Carletta, "Assessing agreement on classification tasks: the kappa statistic," *Computational linguistics*, vol. 22, no. 2, pp. 249–254, 1996.

[14] M. Chau and H. Chen, "A machine learning approach to web page filtering using content and structure analysis," *Decision Support Systems*, vol. 44, no. 2, pp. 482–494, 2008.

[15] R. E. Schapire, Y. Singer, and A. Singhal, "Boosting and rocchio applied to text filtering," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1998, pp. 215–223.

[16] J. Goldstein, V. Mittal, J. Carbonell, and M. Kantrowitz, "Multi-document summarization by sentence extraction," in *Proceedings of the 2000 NAACL-ANLP Workshop on Automatic summarization*. Association for Computational Linguistics, 2000, pp. 40–48.

[17] H.-J. Dai, S. Syed-Abdul, C.-W. Chen, and C.-C. Wu, "Recognition and evaluation of clinical section headings in clinical documents using token-based formulation with

conditional random fields," *BioMed research international*, vol. 2015, 2015.
[18] M. M. Rahman and T. Finin, "Understanding the logical and semantic structure of large documents," *arXiv preprint arXiv:1709.00770*, 2017.
[19] A. Kazantseva and S. Szpakowicz, "Hierarchical topical segmentation with affinity propagation," in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, 2014, pp. 37–47.

IX. Appendix

A. Effect of non-standard styling

Figure 11 shows the distribution of number of words per text block. It can be seen that some blocks may have an unusually huge word count (nearly 1000K words in some books). This happens due to authors not following standard formatting/layout styles, resulting in block extraction being unable to split the e-book into normal-sized sections.

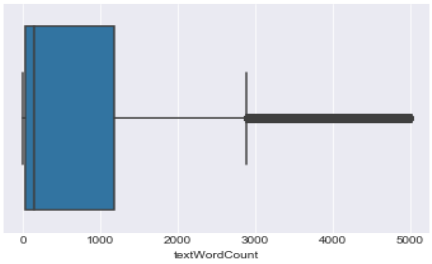

Fig. 11. Text Word Count Distribution in the corpus

B. Destructive vs Distractive Errors

We categorize LBCK classification errors into one of the following types:

1) **Destructive errors:** Auto-predicted SRL is greater than the expected/correct SRL (i.e. automation predicts the SRL to be further down the book that what a reader would expect)
2) **Distractive errors:** Auto-predicted SRL is less than the expected/correct SRL (i.e. automation predicts the SRL to be earlier in the book that what a reader would expect)

While both types of errors lead to poor reading experience (as a reader will be forced to flip through pages to reach the start of main content), destructive errors lead to much worse experience since reader might miss reading an important section of main content and yet not realize it immediately.

C. Sample LBCK

A sample LBCK is given here from the eBook "Harry Potter and the sorcerer's stone" [1]. The LBCK was generated from the first chapter of that book.

- *title_level* h1
- *title_text* Chapter 1 The Boy who lived
- *title_font* Times New Roman
- *content_text* Mr and Mrs Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people youd expect to be involved in anything strange or mysterious, because they just didnt hold with such nonsense...He couldnt know that at this very moment, people meeting in secret all over the country were holding up their glasses and saying in hushed voices: To Harry Potter the boy who lived!
- *list_of_images* /tmp/lbck2.1.png,