Document Chunking and Learning Objective Generation for Instruction Design

Khoi-Nguyen Tran  
IBM Research  
Australia  
khndtran@au1.ibm.com

Jey Han Lau  
IBM Research  
Australia  
jeryan.lau@au1.ibm.com

Danish Contractor  
IBM Research  
India  
dcontrac@in.ibm.com

Utkarsh Gupta*  
IBM Research  
India  
utgupta3@in.ibm.com

Bikram Sengupta  
IBM Research  
India  
bsengupt@in.ibm.com

Christopher J. Butler  
IBM Research  
Australia  
chris.butler@au1.ibm.com

Mukesh Mohania  
IBM Research  
Australia  
mukeshm@au1.ibm.com

ABSTRACT
Instructional Systems Design is the practice of creating instructional experiences that make the acquisition of knowledge and skill more efficient, effective, and appealing [18]. Specifically in designing courses, an hour of training material can require between 30 to 500 hours of effort in sourcing and organizing reference data for use in just the preparation of course material. In this paper, we present the first system of its kind that helps reduce the effort associated with sourcing reference material and course creation. We present algorithms for document chunking and automatic generation of learning objectives from content, creating descriptive content metadata to improve content-discoverability. Unlike existing methods, the learning objectives generated by our system incorporate pedagogically motivated Bloom’s verbs. We demonstrate the usefulness of our methods using real world data from the banking industry and through a live deployment at a large pharmaceutical company.

1. INTRODUCTION
Recent estimates suggest that on average, an organization spends nearly $1200 per year, per employee for training. Apart from the costs incurred in delivering training, significant costs are associated with instruction design activities such as sourcing and preparation of course materials. Currently, most of these activities are very human-intensive in nature, and they rely on the experience and expertise levels of instruction designers and intense reviews by subject-matter experts (SMEs) to achieve acceptable quality levels.

*Utkarsh carried out this work during his employment with IBM Research.


1.1 Course Creation: Workflow and Challenges
Figure 1 shows the typical steps involved in creating a new course. In the first step, instructional designers search for existing learning content that can be used for reference while developing the course. The learning objectives of the new (to be designed) course informs this search process. Reference materials may include existing courses and resources as well as other informal learning materials, such as those available in the form of media articles, blogs etc.

In the next step, the new course is designed and implemented by: extracting the relevant parts of the selected reference content, transforming them appropriately, and combining with newly developed materials to meet the overall training objectives. The new course content is finalized with SME review and approval. Finally, the course is uploaded to a repository for access by end users such as instructors and employees.

The average time taken to produce an hour of material this way can vary between 50 to 300 hours depending on the nature of the course being created. The efficiency with which a new course can be assembled rests on two critical factors: (a) the ability to quickly locate an existing reference material, which is relevant to a learning objective that is part of the planned new course; and (b) the ability to identify (and eventually extract) appropriate parts of this material for use within the new course.
1.2 Contributions

In this paper, we present the first system of its kind that helps reduce the effort associated with sourcing reference material and course creation. We present algorithms for document chunking and automatically generating learning objectives from content as well as creating descriptive content meta-data that improves content-discoverability. Our novel methods for document chunking incorporate syntactic and stylistic features from text as well as a semantic vector-based representation of document text to identify meaningful chunks. Each chunk is physically persisted and a learning objective consisting of Bloom’s verb [3] along with a descriptive keyphrase is generated and associated with each chunk. To the best of our knowledge, we are the first to generate learning objectives incorporating Bloom’s verbs and our system is the first of its kind that directly addresses the challenges in instruction design.

We describe experiments using real-world data from two industries: banking and pharmaceutical. Our results on data from the banking industry shows that our document chunking methods are useful for instruction designers. We report an average user rating of 2 out of 3 in a blind study to assess the quality of chunks and an F1 score of 0.62 computed against expert generated gold standard chunks. Furthermore, in the challenging problem of generating learning objectives, the output from our system has an F1 score of 0.70 for predicting Bloom’s verbs with an average user rating of 2.2 (out of 3) for the associated keyphrase. We also present details of a live deployment of our solution at a large pharmaceutical company.

2. RELATED WORK

To the best of our knowledge, our system is the first (commercial or prototype) that can automatically chunk/segment learning material and label them with system-generated course objectives. We highlight some related work directly relevant to the subcomponents of document chunking and learning objective generation.

Document chunking: Broadly, most methods for chunking/segmentation of text rely on detecting changes in vocabulary usage patterns [11, 14, 15], identifying topical shifts [6, 7, 23], or employing graph based techniques to identify boundaries [9, 28]. The TextTiling [11] document segmentation algorithm uses shifts in vocabulary patterns to mark segment boundaries. Works such as Riedl and Biemann [25] adapt the TextTiling algorithm to work on topics generated by Latent Dirichlet Allocation. Glavis et al. [9] use a graph based representation of documents based on semantic relatedness of sentences to identify document segments. More recent work [1, 2] uses semantic distance computed based on vector embeddings to identify chunk/segment boundaries. Our work on document chunking is based on this direction of research. We use file format specific APIs to physically persist document chunks, retaining any stylistic and presentation elements from the original document.

Learning Objective generation: Most learning management solutions either rely on user provided learning objectives or automated methods to label documents with existing learning objectives specified in curricula [4]. Methods such as Bhartiya et al. [2] and Contractor et al. [5] use a curriculum hierarchy to label learning material with learning objectives. Milli and Hearst [22] simplify the problem of generating course objectives by directly using document keyphrases as learning objectives. Similarly, Lang et al. [16] and Rouly et al. [26] simplify generating objectives using topic modeling to identify candidate learning objectives, where Lang et al. [16] also suggest a system to match topics with Bloom’s verbs. In contrast, we associate keyphrases with Bloom’s verbs [3] and rerank them to select the best candidates for use as learning objectives. To the best of our knowledge, we are the first to generate pedagogically motivated learning objectives incorporating Bloom’s verbs.

3. DOCUMENT CHUNKING

Course materials can often be very large and monolithic, covering a great number of topics and learning objectives, which makes consumption difficult. To make these course materials more discoverable, we automatically segment courses into smaller chunks that can persist independently in the course repository. We present three chunking approaches in the following sections.

3.1 Structure guided (SYNTACTIC-CHUNKER)

Section headings are often the most natural chunk boundaries as they reflect the organization of content by the document creator. Formats such as Microsoft Word have an underlying XML structure that allows us to create these natural chunks easily. However, for PDF documents, there is no encoded document structure information, but we can recover the section titles by analyzing the font sizes of text. To build the SYNTACTIC-CHUNKER, we use a combination of Apache PDFBox for PDF documents, Aspose APIs for Microsoft Office documents and Apache Tika for all other document formats.

Algorithm 1 details the syntactic chunking algorithm where we do not have markers for the section headings. The algorithm aims to find the font size of the largest heading in the document for chunking. The SYNTACTIC-CHUNKER first groups the lines in the document by their font size (sequentially). For each of these font groups, the algorithm gathers statistics on the chunks that would be created for each group’s font size. The largest font size (i.e. the top most section titles) is then chosen from the groups that satisfies the heuristics given in the chunking hyperparameters. An example heuristic is whether the number of chunks created by this font size is between 3 and 20, which is the number of sections or subsections we expect a document or a chapter to contain on average. The significant heuristics/hyperparameters for this algorithm are given in Table 1.

Finally, the line indices marking the start of the section headings are recovered through the font groups created earlier. These starting line indices are then further processed in the main algorithm for creating the physical chunks or storing the metadata.

---

4https://pdfbox.apache.org/
5https://docs.aspose.com/dashboard.action
6https://tika.apache.org/
3.2 Topically guided (semantic-chunker)

Some document styles have ambiguous semantic separation of content, such as presentation slides, informal articles, and blogs. These document styles often have repeated font sizes and text that do not provide distinguishing characteristics for syntactic chunking. For example, presentation slides often have repeated font sizes for slide titles, causing the syntactic-chunker to create a separate chunk for each slide. For these documents, their text content is more useful for inferring chunk boundaries than syntactic markers.

To chunk these documents, we use a divide-and-conquer approach based on topical or content shifts. We represent the content using mean bag-of-word embeddings, which are pre-trained word2vec embeddings [20, 21]. We tokenise words using whitespace, and discard common symbols such as con-
mas and periods. When computing the mean embedding, stopwords are excluded.\textsuperscript{9} The divide-and-conquer method first identifies a boundary that separates a document into two partitions that have the maximum cosine distance using the vector embeddings (providing topical diversity), and then recursively creates subpartitions until a minimum text length is reached. The search strategy is simpler compared to dynamic programming and iterative improvement techniques typically used in the literature [1] but we found this divide-and-conquer strategy performs encouragingly.

The pseudocode and hyperparameters for the \textsc{semantic-chunker} algorithm with modifications to create the \textsc{hybrid-chunker} are in Algorithm \ref{alg:hybrid-chunker} and Table \ref{tab:hyper}, respectively. Both algorithms share similar hyperparameters and similar divide-and-conquer logic but on different data structures.

\subsection{Hybrid method (\textsc{hybrid-chunker})}

The \textsc{semantic-chunker} relies purely on content information for chunking, ignoring potentially usable structural information. From preliminary experiments, we observed that the \textsc{semantic-chunker} occasionally partitions documents at arbitrary positions in the text. For example, a few lines after the start of a new section where the topical shift should be stronger. To resolve this, we developed a hybrid method that uses both structural and content information. Similar to the \textsc{syntactic-chunker}, we record font sizes for each line, and gather lines that share a similar font size into a data structure. With these data structures, we apply the same divide-and-conquer approach used in the \textsc{semantic-chunker} to recursively partition the document into multiple chunks. This forces the chunker to create partitions at natural text boundaries, when this information is available.

\section{Learning Objective Generation}

Traditionally, learning objectives associated with courses are generated manually and are presented in a sentence-like structure. An example from a K-12 Science curriculum in the US: Conduct an investigation to determine whether the mixing of two or more substances results in new substances.\textsuperscript{9} Automatically generating these objectives can be posed as summarization problem where the task is to identify the “learning skill” imparted by the document. However, inferring a skill requires an in-depth understanding of the concepts presented, how they relate with each other, and in courses–such as those that teach soft-skills or behavioural skills–the relationships may be more abstract. Thus, in order to generate tractable yet usable learning objectives, we generate short sentences that are prefixed by a verb from the Bloom’s taxonomy followed by a keyphrase. Recent work such as Milli and Hearst \cite{22} contends with simply using learning skill as learning objectives.

\subsection{Candidate Keyphrase Selection}

Existing methods for keyphrase extraction use a variety of different approaches. Some methods rely on supervision to extract keyphrases \cite{13, 27, 29}, while unsupervised methods often rely on graph-based ranking \cite{19} or topic-based clustering \cite{10, 17}. For our work, we rely on an accessible and effective keyphrase extraction method: IBM Watson Natural Language Understanding (NLU)\textsuperscript{10} to extract keyphrases. NLU is one of many commercially available general purpose keyphrase extraction methods that performs effectively in general keyphrase extraction tasks \cite{8, 12}. We also evaluated other methods such as a variant of TextRank \cite{19}, which has been used in extracting keyphrases from educational material \cite{4}. We chose NLU for the rest of this paper after a blind user study on 243 document chunks indicated a strong preference for these keyphrases as compared to the method employed by Contractor et al. \cite{4}. Table \ref{table:useful} shows the proportion of useful keyphrases\textsuperscript{11} for two keyphrase extraction methods. Further details and results are given in Section 5.3.

As seen from Table \ref{table:useful}, not all keyphrases extracted are useful for inclusion in learning objectives. Thus, to select candidate keyphrases for learning objectives from a general keyphrase list, we rank and select them using a combination of factors:

1. \textbf{Keyphrase score (}: A score between 0-1 returned by the NLU indicating the importance of a keyphrase (1 = most important).
2. \textbf{N-gram TF-IDF score (}: We compute an N-gram level TF-IDF score for each keyphrase using a large domain specific background corpus for IDF score computation.
3. \textbf{Inverse chunk frequency (}: We compute a chunk-level modified IDF score for each keyphrase where the IDF score is computed at the keyphrase level using sibling chunks of a given chunk.
4. \textbf{Google N-gram score (}: The Google Books N-gram service\textsuperscript{12} returns the log-likelihood of a given N-gram from a language model trained on the Google Books corpus. We use the (normalized) rank for a keyphrase within a chunk as the N-gram score.
5. \textbf{Word token level overlap with document section titles (}: Tokens in a section title are likely to contain mentions of important concepts and this acts as a useful signal for selecting keyphrases for learning objectives.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
\textbf{Method} & \textbf{\% Useful Keyphrases} \\
\hline
\textsc{Watson NLU} & 66 \\
\textsc{Modified TextRank} \cite{4} & 51 \\
\hline
\end{tabular}
\caption{Percentage proportion of keyphrases identified by instructional designers as being “useful” for possible inclusion in learning objectives}
\end{table}

\textsuperscript{9}Sources: \url{https://www.cs.ox.ac.uk/teaching/courses/2015-2016/ml/}, \url{https://www.nextgenscience.org/topic-arrangement/structure-and-properties-matter}.

\textsuperscript{10}\url{https://natural-language-understanding-demo.mybluemix.net/}

\textsuperscript{11}“Usefulness” is defined in terms of possible inclusion of a keyphrase in a learning objective, and not in terms of the “quality” of a keyphrase in a general keyphrase extraction task.

\textsuperscript{12}\url{https://books.google.com/ngrams}.
Let weight \( w_i \) be associated with each scoring factor \( f_i \), where there are \( N \) factors. The weights of each factor is normalized to sum to 1.0 (i.e. \( \sum_{i=0}^{N} w_i = 1.0 \)). Let \( K_{s}^{(j)} \) denote the set of top-\( k \) keyphrases selected by the system for the \( j \)-th chunk (based on decreasing order of the score \( \sum_{i=0}^{N} w_i f_i \)). Let the average user rating (see Section 5.3) associated with the keyphrase set \( K_{s}^{(j)} \) be denoted by \( s_j \). Our goal is to select values of \( w_i \) that maximises \( s_j \) for all training examples:

\[
\max \frac{\sum_{j=1}^{M} s_j}{M}
\]

where \( M \) is the number of training examples. The values for \( k \) and parameters \( w_i \) are estimated using grid search.

The tuned hyperparameters for the keyphrase selection are given in Table 4. We found that \( \gamma \) is not useful in these data sets, but maybe useful in other document collections where learning objectives are derived from a few chunks.

### 4.2 Bloom’s Verbs Association

Bloom [3] proposes a taxonomy for promoting learning instead of rote memorization. Bloom’s taxonomy aims to capture the whole pedagogy of learning, teaching, and processing information in a list of “action” verbs. These verbs (referred to as Bloom’s verbs) characterize the activity involved in learning concepts.

Figure 2 shows a representative view of Bloom’s taxonomy. For example, the verb \textit{knowledge} has a list of child verbs such as \textit{identify} and \textit{select}. Similarly, other top-level verbs have their own set of verbs. We experiment with a subset of 10 verbs, as recommended by SMEs. We also explore another more condensed list as suggested by the same SMEs to investigate the potential of hierarchical options. We collapse the 10 verbs belonging to the same parent, resulting in 4 higher-level verb classes in Bloom’s taxonomy. The verb

<table>
<thead>
<tr>
<th>( \alpha )</th>
<th>( \beta )</th>
<th>( \gamma )</th>
<th>( \phi )</th>
<th>( \theta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>bank</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>pharma</td>
<td>0.26</td>
<td>0.32</td>
<td>0</td>
<td>0.32</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Original List</th>
<th>Collapsed List</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>identify</td>
<td>542</td>
<td>bank</td>
</tr>
<tr>
<td>define</td>
<td>85</td>
<td>12</td>
</tr>
<tr>
<td>recall</td>
<td>knowledge</td>
<td>35</td>
</tr>
<tr>
<td>recognize</td>
<td>select</td>
<td>6</td>
</tr>
<tr>
<td>list</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>describe</td>
<td>understand</td>
<td>144</td>
</tr>
<tr>
<td>explain</td>
<td>36</td>
<td>11</td>
</tr>
<tr>
<td>outline</td>
<td>analyze</td>
<td>127</td>
</tr>
<tr>
<td>determine</td>
<td>apply</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 4: Hyperparameter values for \textit{bank} and \textit{pharma} data for keyphrase re-ranking: \( \alpha \) original keyphrase score, \( \beta \) N-gram TF-IDF score, \( \gamma \) Inverse Chunk Frequency, \( \phi \) Google N-gram score, \( \theta \) Overlap with words in section titles.

Table 5: Bloom’s verbs used for generating Learning Objectives and their distribution from a random sample of 100 chunks. Each chunk often has more than one keyphrase describing it, requiring the SMEs to suggest a matching Bloom’s verb.

To associate a verb from Bloom’s taxonomy with a keyphrase learning objective, we train a multilayer perceptron (MLP) to predict a verb given a document (or chunk) and a candidate keyphrase. Thus, the MLP consists of two fully connected (dense) layers with ReLU activation functions [24] in each node. The input of the network is the mean bag-of-words embedding of the document text and the keyphrase.

Word embeddings are pre-trained \textsc{word2vec} embeddings [20, 21] trained on the English Wikipedia. Word embeddings are kept static and not updated during back-propagation.\(^{13}\) This approach of predicting bloom verbs was found to be very effective as shown in Section 5.3.

Two examples of generating learning objectives are shown in Table 6. They show the pairing of a Bloom’s verb with various keyphrases. These pairings are presented to SMEs to evaluate, where their ratings allow us to determine the final rankings to select the most appropriate candidates as learning objectives for a piece of text. Note that the text in the examples (from a document chunk) has been truncated for presentation.

### 5. EXPERIMENTS

#### 5.1 Data sets

We evaluate our chunking and learning objective systems on real-world documents from two industries: banking and finance (henceforth \textit{bank}) and pharmaceuticals (henceforth \textit{pharma}). Table 7 summarizes the word statistics of the two document collections used in our experiments.

The \textit{bank} data set serves as our initial dataset for tuning and testing our methodology, which has a mix of 15 “formal” (e.g. Microsoft Word style) documents and 15 “informal” (e.g. HTML, MediaWiki style, Microsoft PowerPoint slides) documents.

The \textit{pharma} data is a set of client-provided documents with a

\(^{13}\)We also experimented with updating the embeddings (Facebook’s \textsc{fastText}), but found little improvement and thus chose the simpler static model with fewer parameters.
compute the average F1 score on their list of chunk boundaries resolved) based on their understanding of the starting line/page number for each chunk. To illustrate the evaluation method, we give an example:

\[
\text{system chunks} = [1, 4, 4] \\
\text{human chunks} = [1, 3, 4]
\]

where each number in the list denotes the starting page number of a chunk. We omit the first chunk, yielding:

\[
\text{system chunks} = [4, 4] \\
\text{human chunks} = [3, 4]
\]

Precision of the system is therefore $1/2 = 0.5$ (the second starting page number “4” is penalised), the recall is $1/2 = 0.5$, and thus $F1 = 0.5$.

There are a number of hyper-parameters for our chunking methods, which are available in Tables 1 and 2. We tune them manually based on the F1 score using a small labeled development set. Given the tuned models, we apply them to the bank and pharma data sets.

From the chunking performance in Table 8, we found that for formal documents, the SYNTACTIC-CHUNKER (relying on the font size to detect natural chunk boundaries) has the highest accuracy for formal content. In contrast, for the informal content, where structural information may not be very indicative of natural chunk boundaries, we find that the SEMANTIC-CHUNKER gives better results as expected.

In order to qualitatively assess the results of our systems, we also evaluate them with a blind user study. Two expert instructional designers were presented the output of chunks by different chunking algorithms in random order and without information on the underlying algorithm. Each designer was asked to rate a chunk output with 1 (poor), 2 (acceptable), and 3 (good) based on their quality and usefulness from an Instructional Design point of view. Due to complexity and unsupervised nature of the task, ratings above 1 are strongly indicative of natural chunk boundaries, we find that the SEMANTIC-CHUNKER gives better results as expected.

As seen in Table 8, the average ratings for all our best systems is greater than 1.5 indicating our system generated chunks could be acceptable and useful for instructional designers. Furthermore, we find that the scores from the user study reinforce the assessment that formal content (with well structured natural chunk boundaries) are reliably chunked using the SYNTACTIC-CHUNKER algorithm while informal content is better chunked using the SEMANTIC-CHUNKER algorithm.

Surprisingly, we find that the HYBRID-CHUNKER chunking algorithm performs poorly on informal content compared to

<table>
<thead>
<tr>
<th>Bloom’s Verb</th>
<th>Keyphrase</th>
<th>Avg. Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>describe</td>
<td>ach payments</td>
<td>3</td>
</tr>
<tr>
<td>explain</td>
<td>ach transaction flow</td>
<td>2.5</td>
</tr>
<tr>
<td>describe</td>
<td>ACH transactions</td>
<td>2.5</td>
</tr>
<tr>
<td>identify</td>
<td>ACH network</td>
<td>2</td>
</tr>
<tr>
<td>identify</td>
<td>ACH networks</td>
<td>2</td>
</tr>
<tr>
<td>identify</td>
<td>ACH payment request</td>
<td>2</td>
</tr>
<tr>
<td>describe</td>
<td>ACH payments industry</td>
<td>2</td>
</tr>
<tr>
<td>explain</td>
<td>internal ach transaction</td>
<td>2</td>
</tr>
<tr>
<td>identify</td>
<td>traditional ACH payments</td>
<td>2</td>
</tr>
<tr>
<td>identify</td>
<td>ACH</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bloom’s Verb</th>
<th>Keyphrase</th>
<th>Avg. Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>explain</td>
<td>consumer payments</td>
<td>3</td>
</tr>
<tr>
<td>define</td>
<td>Large value payments</td>
<td>2</td>
</tr>
<tr>
<td>describe</td>
<td>payments industry</td>
<td>2</td>
</tr>
<tr>
<td>define</td>
<td>Small value payments</td>
<td>2</td>
</tr>
<tr>
<td>identify</td>
<td>consumer bill payments</td>
<td>1.5</td>
</tr>
<tr>
<td>recall</td>
<td>consumer payments operations</td>
<td>1.5</td>
</tr>
<tr>
<td>identify</td>
<td>corporate-to-corporate payments</td>
<td>1.5</td>
</tr>
<tr>
<td>identify</td>
<td>interbank payments</td>
<td>1.5</td>
</tr>
<tr>
<td>explain</td>
<td>payments</td>
<td>1.5</td>
</tr>
<tr>
<td>identify</td>
<td>banks</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6: Examples of generating learning objectives and their average ratings from SMEs.

5.2 Evaluation: Document Chunking

For tuning and evaluation, we require gold standard chunks for the bank documents. To this end, we ask SMEs to chunk these documents manually, resulting in 243 chunks in total for the 30 documents. The documents were chunked by SMEs (with inter-annotator disagreements of the chunk boundaries resolved) based on their understanding of the subject from an instructional design perspective. The SMEs opted for page level chunks and thus we build our measure of quality at the page level.

Table 7: Data set statistics.

<table>
<thead>
<tr>
<th>Bank</th>
<th>Pharma</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. Documents</td>
<td>30</td>
</tr>
<tr>
<td>No. Word Tokens</td>
<td>376,570</td>
</tr>
<tr>
<td>Vocabulary Size</td>
<td>32,598</td>
</tr>
</tbody>
</table>

To measure the quality of our system against SMEs, we compute the average F1 score on their list of chunk boundaries. We omit the first chunk boundary as it always starts at page 1, and penalise duplicate page numbers (i.e. multiple sections on the same page).
we ask the judges to agree on a particular verb if there is discrepancy. To generate gold standard for the condensed verbs (4 classes), we map the original 10 classes to the 4 classes, as given in Table 5.

### 5.3.2 Selecting Bloom’s Verbs

Given a document and its verbs from the Bloom taxonomy, we train an MLP and optimise its hyperparameters based on 10-fold cross-validation at the chunk level. We use the evaluation metric of mean F1 score over the 10-folds.\(^\text{16}\) We use 2 test sets: (1) all keyphrases and (2) top-5 keyphrases predicted by our system. Note that in each fold, the training data remains the same, but test set (2) is a subset of (1).

We present the classification performance of Bloom’s verbs in Table 10. As expected, the performance in the 4-class prediction task is better than the 10-class prediction due to less confusion amongst classes. Baseline experiments where we assign the majority class for all predictions show a consistent 0.10 drop in F1-score for both the 4-class and 10-class prediction scores.

\(^{15}\)We want our system to select only good quality keyphrases.\(^{16}\)For a particular fold, we compute weighted F1, where it is weighted by the number of true instances for each class.

## 6. Related Work

Our work is related to various aspects of chunking, keyphrase extraction, and Bloom’s verbs. We discuss these in the following sections.

### 6.1 Document Chunking

Document chunking is a key step in many NLP tasks, such as summarization and extracting keyphrases. We use the chunking boundaries provided by the syntactic-chunker, the semantic-chunker, and the hybrid-chunker. The results in Table 8 show that the hybrid-chunker performs the best in formal and informal documents, while the syntactic-chunker performs the best in combined documents.

### 6.2 Keyphrase Extraction

Keyphrase extraction is a crucial step in many NLP tasks, such as information retrieval and topic modeling. We use the top-k keyphrases selected by the system as candidates for the learning objectives of that chunk. From Equation 1, the tuning of factor weights is based on the average user rating of the selected keyphrases, and Precision@N defined as

\[
P@N = \frac{k_p \cap k_s}{|k_s|}\]

where \(k_p\) is the set of gold standard keyphrases that have an average user rating of at least 1.5,\(^\text{15}\) and \(k_s\) is the set of top-\(k\) keyphrases selected by the system. This measure shows whether our selection methods are returning the keyphrases that are relevant for each chunk as determined by the SMEs.

From Table 9 our keyphrase selection method has a P@5 of 0.45 with a high average user rating. This means that 45% of the top 5 keyphrases selected contain the gold standard keyphrases.

### 6.3 Bloom’s Verbs

Bloom’s taxonomy is a well-known framework for classifying educational objectives. We use Bloom’s verbs to select keyphrases that are relevant for each chunk as determined by the SMEs. The results in Table 10 show that the performance in the 4-class prediction task is better than the 10-class prediction due to less confusion amongst classes. Baseline experiments where we assign the majority class for all predictions show a consistent 0.10 drop in F1-score for both the 4-class and 10-class prediction scores.
Proceedings of the 11th International Conference on Educational Data Mining

Table 10: bank: Bloom’s verb (BV) prediction performance. “KP” denotes keyphrase.

<table>
<thead>
<tr>
<th>Test Set</th>
<th>4-Class P@1</th>
<th>4-Class P@3</th>
<th>4-Class P@5</th>
<th>10-Class P@1</th>
<th>10-Class P@3</th>
<th>10-Class P@5</th>
</tr>
</thead>
<tbody>
<tr>
<td>All KP</td>
<td>0.69</td>
<td>0.53</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Top-5 KP</td>
<td>0.70</td>
<td>0.53</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 11: pharma: Candidate Keyphrase Selection for Learning Objective Generation

<table>
<thead>
<tr>
<th>P@1</th>
<th>P@3</th>
<th>P@5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. Rating</td>
<td>1.24</td>
<td>1.35</td>
</tr>
<tr>
<td>Precision</td>
<td>0.1</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Table 12: pharma: Bloom verb prediction performance. “KP” denotes keyphrase.

<table>
<thead>
<tr>
<th>System</th>
<th>Avg. Time Per Document (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTACTIC-CHUNKER</td>
<td>0.41</td>
</tr>
<tr>
<td>SEMANTIC-CHUNKER</td>
<td>0.40</td>
</tr>
<tr>
<td>HYBRID-CHUNKER</td>
<td>0.49</td>
</tr>
<tr>
<td>KEYPHRASE</td>
<td>0.02</td>
</tr>
<tr>
<td>KEYPHRASE_RERANKING</td>
<td>0.03</td>
</tr>
<tr>
<td>BLOOM-VERB</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Table 13: Throughput: Document Chunking, Keyphrase generation, candidate keyphrase selection, and bloom verb prediction (in seconds)

For Bloom’s verb prediction (Table 12), we see a marginally lower performance, but the trend largely remains the same.

6. DEPLOYMENT

Making content discoverable is a key challenge faced by talented development teams in organizations worldwide. Our system addresses this challenge and is currently being piloted at one of the world’s largest pharmaceutical companies to help organize their learning content.

Experiments and Tuning: Using the pharma data shared by the pharmaceutical company (statistics in Table 7), we repeated the bank data set experiments on this data. The pharmaceutical SMEs only wanted generation of document level learning objectives, and not document chunking. Thus, we describe only the experiments for this task. As with the bank experiments, we ask SMEs to rate predicted keyphrases and select the appropriate verb (from the Bloom’s taxonomy) given a document17 and keyphrase.18 For learning objective keyphrase selection and Bloom’s verb prediction, we train and tune the systems with 10-fold cross-validation as before.

Keyphrase selection and Bloom’s verb prediction performance for pharma are presented in Table 11 and Table 12. We find that our candidate keyphrase average rating and precision is lower than what was seen for banking data. We hypothesize a reason for this is due to the extremely dense and domain specific content as well as the requirement of complete documents without chunking when generating learning objectives.

Furthermore, many documents from the pharmaceutical company refer to chemical compounds and chemical formulae, which resulted in skewed TF-IDF weights while selecting candidate keyphrases. Our hypothesis is also backed by the score weights for TF-IDF become less important for pharma data as compared to bank data. We note that the Google N-grams scores were useful for re-ranking keyphrases in both domains. The results also suggest that domain-specific adaption of keyphrase extraction methods (eg. supervised methods) may be required for learning objective generation in content that is very technical.

17These were the original documents and were not chunked.
18We collect annotations for a random 25% subset of the 408 (original) documents, as SMEs simply did not have the time to evaluate all documents due to their length.

6.1 Commercial Deployment

A collection of over 20,000 learning courses have been labeled with learning objectives generated by our system and are being imported into existing learning management systems used by the organization. This is to help the organization retrieve courses efficiently, identify similar course material and prioritize new course development as it allows them to identify gaps in their course material by checking course objectives not covered existing in course material. We briefly describe the architecture of our full system as this is the eventual deployment goal.

6.2 System Architecture

Broadly, the system consists of three subsystems (see Figure 3): (1) UI and Business logic layer, which exposes interfaces for search and enforces business logic for user access; (2) Data Analytics layer, which are Web services for document chunking, keyphrase extraction, learning objective generation. Additional web services that generate different metadata can be easily plugged in and integrated into our system; and (3) Data Storage and Search, where we use Apache Solr to store all generated metadata and document text and to enable search. An illustration of the architecture is presented in Figure 3. Physical documents can either be stored locally or can be accessed via remote requests to learning management systems. Data ingestion from formal course repositories as well as informal sources (web based or Intranet) are supported.

We use document format specific APIs to physically persist document chunks in their original file formats. Our system exposes a simple search interface by which users can query the system using learning objectives. The system allows refinement of search results and also defines user workspaces where course packages can be created and shared.
Table 13 summarizes the average throughput for each of our components (computed on an Intel i5 6300 2.4 Ghz CPU with 8 GB RAM), demonstrating its speed and ease of scalability for large scale processing.

7. DISCUSSION AND CONCLUSION
In this paper, we presented the first system that automatically chunks learning material and generates learning objectives derived from content. It consists of modular subcomponents that require little training data for adaptation. The cloud based web service architecture enables effective use of each of its capabilities.

Our system uses a state-of-the-art embedding-based approach to chunk learning material into meaningful chunks. It also uses generic structural features from the document to guide chunking. It employs a novel methodology for generating learning objectives, which combines automatically generated verbs from Bloom’s taxonomy and extracted keyphrases.

Our system’s capabilities are being used by a large pharmaceutical company to organize learning material. We present detailed experiments on two different domains that demonstrate the applicability of our work.

In future work, we look to extend the work with improvements to our document ingestion capabilities, such as supporting images and videos using OCR and extracting headers and footers, and tabulated data. We would also like to add capabilities that aid instructional designers with other aspects of course design, such as discovering similar courses, summarizing documents, and improving learning objective generation to support a wider set of verbs from Bloom’s taxonomy as well as supervised approaches for keyphrase generation in highly technical domains.

Acknowledgements
We thank our colleagues from IBM Global Business Services for leading this project on the business side. In particular, many thanks to Prasanna C Nair, Sandra Misiaezek, Madhusmita P Patil, Narasimhan K Iyengar, Renjith K Mathew, Partha S Guha, Pinaki Chakladar, Richa Sethi, Tulasi S Maneapalli, Anindita Gupta, and Vinod Uniyal. Our research would not have been possible without their vision, support, data, expertise, and client engagements.

8. REFERENCES


