Doc2Cube: Automated Document Allocation to Text Cube via Dimension-Aware Joint Embedding

Fangbo Tao1, Chao Zhang1, Xiusi Chen1, Meng Jiang2, Tim Hanratty3, Lance Kaplan3, Jiawei Han1

1Dept. of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA
2Dept. of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, USA
3U.S. Army Research Laboratory, Adelphi, MD, USA
1{ftao2, czhang82, xiusic, hanjj}@illinois.edu
2mjiang2@nd.edu 3{timothy.p.hanratty.civ, lance.m.kaplan.civ}@mail.mil

ABSTRACT
Data cube is a cornerstone architecture in multidimensional analysis of structured datasets. It is highly desirable to conduct multidimensional analysis on text corpora with cube structures for various text-intensive applications in healthcare, business intelligence, and social media analysis. However, one bottleneck to constructing text cube is to automatically put millions of documents into the right cells in such a text cube so that quality multidimensional analysis can be conducted afterwards—it is too expensive to allocate documents manually or rely on massively labeled data. We propose Doc2Cube, a method that constructs a text cube from a given text corpus in an unsupervised way. Initially, only the label names (e.g., USA, China) of each dimension (e.g., location) are provided instead of any labeled data. Doc2Cube leverages label names as weak supervision signals and iteratively performs joint embedding of labels, terms, and documents to uncover their semantic similarities. To generate joint embeddings that are discriminative for cube construction, Doc2Cube learns dimension-tailored document representations by selectively focusing on terms that are highly label-indicative in each dimension. Furthermore, Doc2Cube alleviates label sparsity by propagating the information from label names to other terms and enriching the labeled term set. Our experiments on a real news corpus demonstrate that Doc2Cube outperforms existing methods significantly. Doc2Cube is a technology transferred to U.S. Army Research Lab and is a core component of the EventCube system that is being deployed for multidimensional news and social media data analysis.

1 INTRODUCTION
Text cube [16, 22, 31, 34] is a multidimensional data structure with text documents residing in, where the dimensions correspond to multiple aspects (e.g., topic, time, location) of the corpus. Text cube analysis has been demonstrated as a powerful text analytics tool for a wide spectrum of applications in bioinformatics, healthcare, and business intelligence. For example, by organizing a news corpus into a three-dimensional topic-time-location cube, decision makers can easily browse the corpus and retrieve desired articles with simple queries (e.g., (Sports, 2017, USA)). Any text mining primitives, e.g., sentiment analysis, can be further applied on the retrieved data for gaining useful insights. As another example, one can organize a corpus of biomedical research papers into a neat cube structure based on different facets (e.g., disease, gene, protein). Such a text cube allows people to easily identify relevant papers in biomedical research and acquire useful information for disease treatment.

Previous studies on text cube analysis [8, 12, 24, 32, 34, 35] assume the cube has already been constructed by data providers and focus on different text analytics tasks. Text cube construction, i.e., which automatically constructs a text cube from a text corpus, has remained largely overlooked. Specifically, given a text corpus $D$ and a pre-defined cube schema $C$, the task aims to allocate the documents in $D$ into the right cells in $C$. Figure 1 shows an example on a news corpus. Let $C$ be a pre-defined cube schema with three dimensions: topic, location, and time. The text cube construction
task is to assign each news article in the given corpus into a proper cube cell (e.g., (Sports, 2017, USA)), by choosing one label along each dimension to best match the textual content of the article.

Text cube construction is a multidimensional categorization problem in nature and closely related to document classification [1, 27, 28, 33]. Nevertheless, it has two unique challenges that prevent existing document classification methods from being applied: (1) The first challenge is the lack of labeled training data. The success of prevailing document classification methods largely relies on sufficient labeled document-label pairs to train reliable classifiers. For text cube construction, it is costly to manually annotate a large number of documents for classification, given that every document has to be assigned with multiple labels; (2) The second is to extract discriminative features for different dimensions. Existing document classification methods typically extract a set of lexical features, or learn distributed representations for textual units (words, sentences, or documents) to derive document representations. Either way, each document is represented as one fixed feature vector. In text cube construction, however, the categorization tasks along different dimensions often require different information from the same document. Continuing the news corpus example in Figure 1, the location dimension may favor location-indicative terms such as “Chicago” and “China” as features, while the topic dimension may favor semantics-telling ones such as “Super Bowl” and “Economy”. Existing text categorization methods derive fixed document representations and are dimension-agnostic. As a result, irrelevant terms are overemphasized in the representation, which often hurts the categorization performance.

We propose Doc2Cube, a method that constructs text cube from a given text corpus in an unsupervised way. Doc2Cube is a technology transferred to U.S. Army Research Lab and is a core component of the EventCube system[1] that is being deployed for multidimensional news and social media data analysis. Regarding label names as a small set of labeled seed terms, Doc2Cube first constructs a tripartite graph to encode the correlations among labels, terms, and documents. It then iteratively refines the graph structure and derives quality embeddings of labels, terms, and documents to uncover their inter-type similarities. During the iterative embedding process, Doc2Cube features two novel components to obtain discriminative joint embeddings: document focalization and label expansion.

The document focalization component gradually sparsifies the term-document sub-graph by emphasizing discriminative terms. As shown in Figure 2, a document is initially connected with all the terms appearing in it. The resultant document embedding is over-represented in the sense that many terms indiscriminative to the current dimension are encoded. To address this issue, Doc2Cube iteratively estimates the discriminativeness of terms for each cube dimension, and emphasizes discriminative ones to generate tailored document embeddings. As such, one document can have multiple representations—each tailored for one cube dimension by highlighting truly discriminative information.

The label expansion component iteratively densifies the label-term subgraph to address the label sparsity problem. As shown in Figure 2, as each label is only connected to its surface name in the beginning, the initial label embedding is under-represented because many other relevant terms are overlooked. To tackle this issue, Doc2Cube computes the correlations between labels and terms along different dimensions, and iteratively links each label with positively correlated terms. In this way, the information is propagated from label names to other semantically relevant terms for alleviating label sparsity.

Our contributions can be summarized as follows:

(1) We propose an unsupervised method for text cube construction. It does not require excessive labeled data, but simply leverages the surface names of different labels to achieve effective text categorization along different cube dimensions.

(2) We propose a novel dimension-aware joint embedding algorithm. It learns dimension-aware embeddings by focusing on discriminative terms and propagating information from label names to other terms to alleviate label sparsity.

(3) We have performed extensive experiments using a real-life news corpus. The results show that our method generates high-quality embeddings and significantly outperforms state-of-the-art methods.

---

[1] A video introduction of the system is available at https://goo.gl/dv2sCA
2 RELATED WORK

We examine related work in three aspects: text cube analysis, text categorization, and embedding learning.

Text Cube Analysis. Lin et al. [16] were the first to propose the text cube concept. They assumed the text documents have been organized in a neat multidimensional structure and studied how to efficiently compute different aggregation measures in the multidimensional space. Since then, text cube analysis has drawn much attention from the database and data mining communities [8, 22, 32, 34, 35]. Specifically, R-Cube [22] was proposed where users can specify an analysis portion by supplying some keywords and a set of cells are extracted based on relevance. TopCell and TEXplorer were proposed [8, 35] to support keyword-based ranking of text cube cells and facilitate interactive exploration of a text cube. A number of multidimensional analytical platforms [18, 31] were also developed to support end-to-end textual analytics. However, all these studies focus on the text analytics tasks, assuming the cube is already constructed by data providers. The text cube construction task, which aims at organizing massive text documents into a cube, has remained largely overlooked.

Text Categorization. Text construction is closely related to text categorization. Prevailing text categorization methods take a supervised approach. Relying on a sufficient amount of document-label training pairs, they learn reliable classifiers that are capable of predicting the label of any new document, including SVM [13], decision tree [1, 27], and neural networks [33]. Different from supervised text classification, the text cube construction problem does not involve excessive labeled data, but only a text corpus and a pre-defined cube schema. Such a setting makes our problem challenging and existing supervised methods inapplicable.

There are unsupervised or weakly-supervised approaches for text categorization. Ko et al. [14] used heuristic rules to generate training data, but the curated labels often need considerable feature engineering efforts to ensure the quality. OHLDA [6, 10] applies topic model with given labels to generate document classifiers, while leveraging external knowledge from Wikipedia to represent labels. The recently developed dataless classification methods [28] also use Wikipedia to perform explicit semantic analysis of labels and documents to derive vector representations. The common limitation of OHLDA and dataless models is their dependency on external knowledge bases. They suffer from limited performance if the given corpus is closed-domain or has limited coverage by external knowledge bases.

Embedding Learning. Text embedding has demonstrated enormous success in various text mining and NLP tasks. Researchers have developed techniques that learn vector representations for words [19, 21], sentences [20], and documents [15]. Such vector representations can be used as feature vectors for a wide range of classification tasks [2, 15]. Apart from text embedding, another line of work closely related to our method is graph embedding. Various methods have been proposed to learn vector representations for nodes in both homogeneous [23, 30] and heterogeneous graphs [9, 29]. While our proposed method also learns vector representations for labels and documents using graph embedding, there are notable differences between our method and existing techniques. Existing techniques generate static representations for documents and graph nodes. Such fixed embeddings are dimension-agnostic and may not be optimal for all the cube dimensions. In contrast, our method considers the characteristics of different dimensions and obtains dimension-aware document embeddings to achieve high accuracies for all the dimensions.

3 PRELIMINARIES

3.1 Problem Definition

Text cube [16] is a data model that enables multidimensional and multi-granular text analysis. Given a text corpus \( D \), the text cube for \( D \) is a multidimensional data structure. The multiple dimensions, which reveal important aspects (e.g., topic, location, time) of the corpus \( D \), uniquely define the schema of the text cube. Each document \( d \in D \) lies in one multidimensional cube cell to characterize the textual content of the document from multiple aspects. Formally, we define the concepts of cube dimension as follows:

**Definition 3.1 (Cube Dimension).** A cube dimension is defined as \( \mathcal{L} = \{l_1, l_2, \ldots, l_\text{\#}\} \), where \( l_i \in \mathcal{L} \) is a categorical label in this dimension.

Consider Figure 1 as an example. There are three cube dimensions for the given corpus: (1) \( \mathcal{L}_{\text{topic}} \) representing the topic aspect; (2) \( \mathcal{L}_{\text{loc}} \) representing the location aspect; (3) \( \mathcal{L}_{\text{time}} \) representing the time aspect. Then for each article, it should be associated with one label for each of the three dimensions, e.g., label "Economy" for \( \mathcal{L}_{\text{topic}} \), label "China" for \( \mathcal{L}_{\text{loc}} \), and label "2017" for \( \mathcal{L}_{\text{time}} \). The labels from different dimensions partition the space into cube cells.

**Definition 3.2 (Cube Cell).** Given \( n \) cube dimensions, \( \mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n \), a cube cell \( c \) is defined as an \( n \)-dimensional tuple \( (l_1, l_2, \ldots, l_n) \), where \( l_i (1 \leq i \leq n) \) is a label in dimension \( \mathcal{L}_i \).

**Definition 3.3 (Text Cube).** A text cube for a text corpus \( D \) is a \( n \)-dimensional structure \( C = (\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n) \), where \( \mathcal{L}_i \) is the \( i \)-th cube dimension. Each document \( d \in D \) resides in a cube cell \((l_{i_1}, l_{i_2}, \ldots, l_{i_n})\) in \( C \), where \( l_{i_1} \) is the label of \( d \) in dimension \( \mathcal{L}_{i_1} \).

We study the problem of constructing a text cube \( C \) from a text corpus \( D \). In tradition data cube literature [5], this process is also called cube instantiation or cube loading. We formally define this problem in the following.

**Problem 1 (Text Cube Construction).** Let \( C \) be an \( n \)-dimensional text cube with dimensions \( \mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n \), and \( D \) be a corpus of text documents. For any document \( d \in D \), the text cube construction problem is to allocate \( d \) into a \( n \)-dimensional cell in \( C \). This is equivalent to assigning \( n \) labels \( l_{i_1}, \ldots, l_{i_n} \) for \( d \), where label \( l_{i_i} \in \mathcal{L}_{i_i} \) represents the category of \( d \) in dimension \( \mathcal{L}_{i_i} \).

3.2 An Overview of Our Method

The major challenge for applying document classification methods is that there are no labeled documents for training reliable classifiers. Instead, one needs to perform document categorization along different dimensions using only label names and document content. Our method Doc2Cube uses label names to form a small set of seed labeled terms, and use them as weak supervision signals for document categorization. At the high level, Doc2Cube is an approach that learns distributed representations of labels, terms,
and documents. Instead of using bag-of-words as document representation, it learns low-dimensional document embeddings by discovering the correlations among terms. 

As shown in Figure 2, Doc2Cube initially constructs a tripartite label-term-document graph to encode the relationships among labels, terms, and documents along different dimensions, and embeds them into the same latent space. While the initial embeddings encode the seed information and the occurrences of terms in documents, they suffer from two drawbacks: (1) the document embeddings are over-represented in the sense that many terms indiscriminative to the current dimension are encoded; and (2) the label embeddings are under-represented because many other relevant terms are overlooked. To address the above challenges, Doc2Cube features two novel components for learning discriminative joint embeddings in an iterative fashion: (1) the document focalization component that emphasizes different terms for different dimensions, thus deriving dimension-aware document representations; and (2) the label expansion component that propagates information from label names to other terms for alleviating label scarcity.

4 GRAPH-BASED JOINT EMBEDDING

In this section, we describe the joint label-term-document embedding step. For a given dimension $L$, it first constructs a Label-Term-Document tripartite graph (Section 4.1) and then embeds different data types into the same latent space (Section 4.2).

4.1 Label-Term-Document Graph

To model the relationships among labels, terms and documents, we construct an Label-Term-Document (L-T-D) correlation graph. Since different dimensions have different label spaces, we construct an L-T-D graph for each dimension separately. As shown in Figure 2, for each cube dimension, there are three different node types: labels, terms, and documents. The initial graph $G_{LTD}$ is designed to capture two types of relationships: (1) the seed information between label names and terms; and (2) the occurrence information between terms and documents. Hence, we induce two different edge types to encode these relationships: label-term edges and document-term edges. The resultant L-T-D graph is a heterogeneous tripartite graph defined as follows.

**Definition 4.1 (L-T-D Graph).** The L-T-D graph for a dimension $L$ is a tripartite graph $G_{LTD} = (V_{LTD}, E_{LTD})$. The node set $V_{LTD}$ contains all the labels in $L$, terms in $T$, and documents in $D$. The edge set $E_{LTD}$ consists of two types of edges: (1) $E_{TL}$ is a set of edges between labels and terms. There is an edge between term $t_i$ and label $l_j$ if and only if they strictly match each other, and the weight $w_{t_il_j}$ is set to 1; (2) $E_{TD}$ is a set of edges between terms and documents. There is an edge between term $t_i$ and document $d_j$ if $t_i$ occurs in $d_j$, and the edge weight $w_{t_id_j}$ is set to $\log(1 + \text{count}(t_i, d_j))$.

4.2 Graph Embedding

The L-T-D graph $G_{LTD}$ encodes the information from seed terms as well as the co-occurrence relationships between terms and documents. Based on the constructed L-T-D graph, we proceed to learn initial vector representations of labels, terms, and documents. This is achieved by embedding all the nodes in the L-T-D graph into a $D$-dimensional vector space, such that their structural proximities in the graph are preserved. Here, $D$ is a parameter that specifies the dimensionality of the embedding space, e.g., $D = 200$.

The L-T-D graph $G_{LTD}$ is a tripartite graph between labels, terms, and documents. We design the graph embedding task to preserve the information from both the label-term edges $E_{TL}$ and the term-document edges $E_{TD}$. For this purpose, we define the probability of observing a term $i$ given a label $j$ as follows:

$$p(u^T_i | u^L_j) = \frac{\exp(u^T_i \cdot u^L_j)}{\sum_{u^T_i \in T} \exp(u^T_i \cdot u^L_j)},$$

where $u^T_i$ and $u^L_j$ are the $D$-dimensional embeddings of term $i$ and label $j$, respectively. Similarly, we define the probability of observing a term $i$ given a document $j$ as follows:

$$p(u^T_i | u^D_j) = \frac{\exp(u^T_i \cdot u^D_j)}{\sum_{u^T_i \in T} \exp(u^T_i \cdot u^D_j)}.$$  

Now given the L-T-D graph $G_{LTD}$, we learn the embeddings of labels, terms, and documents by collectively preserving the structures of the two bipartite graphs $E_{TL}$ and $E_{TD}$. This is achieved by minimizing the following objective function:

$$O_{lt} + O_{td},$$

where

$$O_{lt} = - \sum_{(i,j) \in E_{TL}} w_{t_il_j} \log p(u^T_i | u^L_j),$$

$$O_{td} = - \sum_{(i,j) \in E_{TD}} w_{t_id_j} \log p(u^T_i | u^D_j).$$

The above objective function is expensive to optimize due to the large amount of terms in the vocabulary. To efficiently learn the joint embeddings, we use the negative sampling strategy [19] with stochastic gradient descent (SGD) for optimizing Equation 3.

5 DIMENSION-AWARE UPDATING

In this section, we present the dimension-aware embedding updating step. Taking the joint embeddings as initialization, the updating step iteratively derives dimension-aware document embeddings by focusing on discriminative terms for each dimension, and expands the initial labeled seed terms to address label sparsity.

5.1 Measuring Term Discriminativeness

Although the joint embeddings capture the co-occurrence information among labels, terms, and documents, the resultant embeddings suffer from two problems. First, the document embedding is fixed for all the dimensions. In text cube construction, different dimensions require different representation for the same document. For instance, the location dimension may favor terms that captures location-related information, such as “super bowl” and “economic growth”. Second, the scarcity of labeled terms makes label embeddings not comprehensive enough to cover the semantics of the target category. For example, with the provided seeds, the label “Sports” is only linked to the term “sports”. However, the scope of “Sports” is quite broad, information such as

...
“nba”, “nfl”, and “soccer”. Consequently, the initial joint embeddings over-represent documents while under-represent labels.

The key to tackling the above two problems is to estimate each term’s discriminative power \(w.r.t\) a dimension and a label. The computed discriminative scores can address the over-represented document embedding problem by emphasizing discriminative terms and understating indiscriminative ones. In the mean time, for the under-represented label embedding problem, the discriminative scores of terms allow for expanding each label to highly relevant terms. In what follows, we define the label-focal score and the dimension-focal score of a term \(t\) and describe how we compute these two measures.

### 5.1 Focal Scores

#### 5.1.1 Label-Focal Score

The label-focal score of a term \(t\) \(w.r.t\) a label \(l\) in dimension \(\mathcal{L}\), denoted as \(f(t, l)\), aims at quantifying the discriminative power of the term \(t\) for the label \(l\). The higher \(f(t, l)\) is, the more exclusively the term \(t\) belongs to the label \(l\).

Our strategy for measuring the label-focal score \(f(t, l)\) is to leverage the documents containing \(t\) to derive the distribution of term \(t\) over all the labels in dimension \(\mathcal{L}\). Specifically, with the document embedding matrix \(\mathbf{U}^D\) and the label embedding matrix \(\mathbf{L}\), we first compute the label-document similarity matrix as:

\[
\mathbf{R}^{(D,L)} = \mathbf{U}^D \mathbf{L}^T.
\]  

(4)

In the above, \(\mathbf{R}^{(D,L)}\) is a \(|\mathcal{D}| \times |\mathcal{L}|\) matrix that gives the similarities between documents and labels in the embedding space. Combining it with the term-document subgraph, we are able to further compute the similarities between labels and terms. Specifically, let \(\mathbf{A}^{(T,D)}\) be the adjacency matrix for the term-document subgraph in \(\mathcal{G}_{LDT}\), we compute the term-label similarities as:

\[
\mathbf{R}^{(T,L)} = \mathbf{A}^{(T,D)} \mathbf{R}^{(D,L)},
\]  

(5)

where \(\mathbf{R}^{(T,L)}\) is a \(|\mathcal{T}| \times |\mathcal{L}|\) matrix keeping the similarities between terms and labels. Base on \(\mathbf{R}^{(T,L)}\), we apply row-wise softmax function to derive the probability distribution of each term over the labels. Finally, we define the label-focal score \(f(t, l)\) as the probability of assigning term \(t_i\) to label \(l_j\). Namely,

\[
f(t_i, l_j) = \mathbf{R}^{(T,L)}_{ij}.
\]  

(6)

#### 5.1.2 Dimension-Focal Score

We proceed to define the dimension-focal score of a term. Informally, the dimension-focal score of a term \(t_i\) \(w.r.t\) dimension \(\mathcal{L}\), denoted as \(f(t_i, \mathcal{L})\), aims to quantify how discriminative the term \(t_i\) is for the categorization task along dimension \(\mathcal{L}\). The higher \(f(t_i, \mathcal{L})\) is, the more useful term \(t_i\) is for deciding the label in dimension \(\mathcal{L}\).

We measure the dimension-focal score \(f(t_i, \mathcal{L})\) based on the distribution of term \(t_i\) over all the labels in dimension \(\mathcal{L}\). Recall that the matrix \(\mathbf{R}^{(T,L)}\) gives the label distribution of term \(t_i\). We compute its normalized KL-divergence from the uniform distribution of \(t_i\) over all the labels as the dimension-focal score. Formally, the dimension-focal score \(f(t_i, \mathcal{L})\) is given by:

\[
f(t_i, \mathcal{L}) = \frac{\sum_{l=0,\ldots,|\mathcal{L}|} \mathbf{R}^{(T,L)}_{ij} \log |\mathcal{L}| \mathbf{R}^{(T,L)}_{ij}}{\log |\mathcal{L}|},
\]  

(7)

where \(\log |\mathcal{L}|\) is a normalization term.

### 5.2 Document Focalization

The document focalization component uses the dimension-focal scores of terms to address the over-represented problem of document embeddings. The rationale is that the fixed document representation encodes the information from all the terms in the vocabulary, even those that are not relevant to the categorization task in the target dimension. With dimension-focal scores, it becomes possible to emphasize discriminative terms while understating irrelevant ones. Consider Figure 2 as an example. As shown, for the topic dimension, the first document is connected to topical terms such as “football” and “sports”, as well as time-indicative terms like “september” and background terms like “report”. Those irrelevant terms in the document can act as background noise and make the categorization task more difficult. Document focalization remedies this problem by emphasizing discriminative terms and generating dimension-tailored document representations, e.g., lowering the weights of “september” and “report” for that document.

To obtain dimension-tailored document embeddings, we use the dimension-focal scores to re-weigh the term-document matrix \(\mathbf{A}^{(T,D)}\), and compute the weighted average of term embeddings. Formally, we update the document embedding matrix \(\mathbf{U}^D\) as:

\[
\mathbf{U}^D = \left(\mathbf{A}^{(T,D)} \circ \left[ f_{\mathcal{L}} \cdots f_{\mathcal{L}} \right]_{|[\mathcal{T}] \times |\mathcal{D}|} \right)^T \mathbf{U}^D \circ \mathbf{L}.
\]  

(8)

where \(\circ\) is the Hadamard product between two matrices; and \(f_{\mathcal{L}}\) is a length-\(|\mathcal{T}|\) vector representing the dimension-focal scores of all the terms along dimension \(\mathcal{L}\). In this formula, the dimension-focal score of each term places a penalty in the range of \([0, 1]\) on the original weight in the matrix \(\mathbf{A}^{(T,D)}\). The document embedding is then an aggregation of term embeddings with penalized weights. The higher a term’s dimension-focal score is, the more it is emphasized when computing the document embedding.

Observing from Equation 4 and 8, it is apparent that the computations of the focal scores \(f_{\mathcal{L}}\) and the document embeddings \(\mathbf{U}^D\) are dependent on each other. These two measures can mutually enhance each other: (1) better document representations lead to more accurate labeling of the documents and thus better estimations of term focal scores; and (2) more accurate focal scores surface terms that are important to the dimension and result in more discriminative document embeddings. Consequently, we design an iterative process that updates \(\mathbf{R}^{(D,L)}\), \(f_{\mathcal{L}}\) and \(\mathbf{U}^D\) alternatively until they stabilize. We will describe the iterative process shortly in Section 6.

### 5.3 Label Expansion

The label expansion component is designed to solve the under-represented problem of label embeddings. The intuition behind it is to link each label with other positively correlated terms in addition to its surface name. For example, in Figure 2, it is reasonable to expand the label “Sports” to the term “football”, and the label “Economy” to the term “stock”. As such, the label-term subgraph is enriched and the obtained label representations encode the semantics of relevant terms more comprehensively.

To ensure the quality of the expanded terms, we consider two factors: (1) the label-focal score of a term; and (2) the popularity of a term. The label-focal score is critical to determining the correlations between a term and the considered label. However, we observe that
only using the label-focal score could link the label to many low-quality terms during the label expansion process. This is because many terms that have high discriminative power are infrequent in the corpus. Expanding labels to them not only covers few extra documents, but also suffers from their inadequately-trained embeddings. Hence, we design the expansion criterion by combining the label-focal score and the term popularity. Given a term $t_i$ and a label $l_j$, we compute the expansion score of term $t_i$ for label $l_j$ as:

$$e(t_i, l_j) = \frac{f(t_i, l_j) - \log(1 + |D|)}{\log(1 + |D|)} > \eta$$ (9)

where $d f(t_i)$ is the document frequency of term $t_i$. The second term thus reflects the normalized popularity of term $t_i$. In Equation 9, $\eta > 0$ is a pre-defined threshold for label expansion. Any term-label pairs with the expansion scores higher than $\eta$ are connected and the adjacency matrix $A(L \times T)$ is updated accordingly. After the expansion, we compute the label embedding as:

$$u^L = A(L \times T)u^T.$$ (10)

Since the label expansion process changes label embeddings, the label-focal scores of terms will be updated according to the newly computed $R(D \times L)$ and $R(T \times L)$. As label-focal scores are updated, a new label expansion operation could further benefit generating high-quality label embeddings. We design an iterative process to perform label expansion and focal score computation in turn, which will be described shortly.

6 THE OVERALL ALGORITHM

In this section, we put different pieces together and summarize the entire procedure of Doc2Cube for text cube construction. There are three major steps in Doc2Cube: (1) joint embedding of labels, terms, and documents; (2) dimension-aware embedding updating; and (3) label assignment. Algorithm 1 sketches the overall process of Doc2Cube. As shown, given the corpus, we first build the L-T-D tripartite graph and compute the joint embeddings of labels, terms, and documents (lines 2 - 8). Then we iteratively update the embeddings based on Algorithm 2 to derive dimension-aware document embeddings (line 9). Finally, we assign the max-scoring label to each document for the target dimension (line 10 - 11). The label assignment step is achieved by directly measuring the cosine similarity between label embedding and document embedding.

Algorithm 2 presents the iterative embedding updating process for document and label embeddings. Starting with the initial embeddings for labels ($u^L$), terms ($u^T$), and documents ($u^D$), we iteratively perform document focalization and label expansion to obtain more discriminative dimension-aware embeddings. In the document focalization component (lines 2 - 5), we compute the dimension-focal scores of terms, and update the document embeddings according to Equation 8; while in the label expansion component (lines 6 - 8), we compute the label-focal scores of terms, and update the label embeddings according to Equation 10.

Time Complexity. The total time cost of Doc2Cube involves two parts: (1) the initial joint embeddings; and (2) the iterative updating. For the first part, along each dimension, Doc2Cube needs to sample $M$ edges for graph embedding. For each sampled edge, Doc2Cube generates $K$ negative samples to update the $D$-dimensional embeddings. The time cost is thus $O(nMKD)$. For the second part, Doc2Cube performs $T$ iterations for updating the embeddings. In each iteration, computing the focal scores takes $O(n \cdot |T| \cdot |D| \cdot |L|_{max} \cdot D)$ time where $|L|_{max}$ is the maximum cardinality of the label set for all the dimensions. Once the focal scores are computed, Doc2Cube updates the embeddings with time complexity $O(n \cdot |T| \cdot |D| \cdot |L|_{max} \cdot D)$. The overall time complexity of Doc2Cube is $O(nMKD + nT \cdot |T| \cdot |D| \cdot |L|_{max} \cdot D)$. Note that the variables $n$, $K$, $D$, $T$, and $|L|_{max}$ are usually small in practice.
7 EXPERIMENTS

7.1 Experimental Setup

7.1.1 Dataset. We use a real news dataset in our experiments. Our dataset, named NYT, is a collection of New York Times articles. We crawled 13,080 articles using New York Time API\(^2\) in 2015. The articles in the corpus cover 29 topics and 14 countries, and each article contains exactly one topic label and one country label. Accordingly, two dimensions are involved for constructing a text cube for the NYT corpus: Topic and Location. The annotations of different articles along these two dimension are used as ground truth. Before applying different methods on this dataset, we use an existing phrase mining tool\(^3\) to segment each article into phrases. Furthermore, we remove all the stopwords and the phrases that appear less than 10 times. Our code and data are available at https://github.com/fangbo-tao/doc2cube.

7.1.2 Baselines. To demonstrate the effectiveness of Doc2Cube, we compare it with multiple baselines that can perform document categorization in an unsupervised or weakly-supervised way. We describe these baseline methods as follows. (1) IR [25] treats each label as a keyword query and performs categorization based on the BM25 retrieval model. Using BM25, the label that achieves the highest query relevance is assigned to the considered document. (2) IR + Expansion (IR+QE) [7, 26] extends the IR method by expanding label names using Word2Vec [19] and using the expanded term set as queries. (3) Word2vec (W2V) [19] first learns vector representations for all the terms in a given corpus, and then derives label and document representations by aggregating their member terms. Finally, the most similar label for a document is assigned based on cosine similarity. (4) Word2vec + Focalization (W2V+DF) extends W2V with our document focalization component. Instead of simply aggregating term embeddings for document representation, we leverage term dimension-focal scores to compute document representations. (5) Paragraph2vec (P2V) [15] directly learns vector representations of documents, by embedding documents and terms into the same semantic space. (6) Topic Model (TM) [3] trains the LDA model with the given corpus. When assigning labels along each dimension, we use the likelihood of observing a label name given the document to choose the most likely label. (7) Semi-Supervised Topic Model (Semi-TM) [17] extends the PLSA model [11] by using labels as guidance and forcing the learned topics to align with the provided labels. (8) Dataless Classification (Dataless) [4, 6, 28] is an unsupervised algorithm that utilizes Wikipedia as external knowledge base. It leverages Wikipedia and Explicit Semantic Analysis (ESA) to derive vector representations of labels and documents. (9) PTE [29] is a semi-supervised method that jointly embeds documents, terms, and labels into the same latent space and directly uses the embeddings for categorization.

Besides the above baseline methods, we also design two ablation algorithms to evaluate the separate effects of document focalization and label expansion during the joint embedding process in Doc2Cube: (1) D2C-DF updates document embeddings for each dimension using document focalization. However, the label embeddings are not updated with the label expansion component. (2) D2C-LE updates label embeddings iteratively with the label expansion component. However, it does not include document focalization for deriving dimension-aware document embeddings.

7.1.3 Evaluation Protocol. For our used dataset, there are two dimensions for the given corpus, and each document has one label along each dimension. To evaluate the performance of different methods, we use them to allocate all the documents in the corpus, and measure the F1 scores along different dimensions.

We set the parameters of different methods as follows. There are three major parameters in Doc2Cube: (1) the latent embedding dimension \(D\); (2) the number of iterations for embedding updating \(T\); and (3) the correlation threshold for label expansion \(\eta\). After tuning, we set these parameters as \(D = 200\), \(T = 3\) and \(\eta = 0.8\). We will also show the performance of Doc2Cube when these parameters vary. For the baseline methods, we set the embedding dimensions for W2V and PTE to 200 to ensure fair comparison with Doc2Cube; we set the number of topics to 200 for TM; and we set the number of Wikipedia concepts to 500 for Dataless.

7.2 Effectiveness Evaluation

In this subsection, we demonstrate the effectiveness of different methods, and also study the effects of different parameters on their performance.

7.2.1 Performance Comparison. In the first set of experiments, we demonstrate the effectiveness of different methods on the NYT dataset. As shown in Table 1, we report the micro-F1 and macro-F1 scores of all the methods along different dimensions. Comparing the performance of different methods, one can observe that Doc2Cube outperforms all the baselines in both dimensions. Semi-TM is the strongest baselines along the topic and location dimensions in terms of micro-F1. However, Doc2Cube outperforms it by more than 16.2% in the topic dimension and 37.3% in the location dimension. It is also interesting that all the methods perform better in the topic dimension on NYT. Our investigations into the data reveal two reasons for this phenomenon. First, the majority of the articles in the dataset contain topic-indicative terms, while a smaller portion of documents include keywords that indicate the location of the event. Second, quite a few categories in the location dimension have a small number of documents (~100). As a result, the respective label embeddings are not learned sufficiently due to data scarcity, and the overall F1 scores are harmed by those categories.

From Table 1, one can clearly observe the necessity of learning dimension-aware embeddings to achieve good performance across all the dimensions. We can see while certain dimension-agnostic methods (e.g., W2V and TM) can achieve reasonably good performance in the topic dimension, their performance drops drastically in the location dimension. In contrast, Doc2Cube achieves strong performance on both the topic and location dimensions, which validates the benefits of our design of learning dimension-aware document representations.

Comparing the different ablations of Doc2Cube, we can observe the benefits of the document focalization and label expansion components. On the NYT dataset, the inclusion of document focalization (D2C-DF v.s. PTE) improves the micro-F1 score from \(~0.69\) to \(~0.78\) in the topic dimension; and the inclusion of label expansion...
imposes a stricter condition when connecting the label to relevant terms. As shown in Figure 3d, as $\eta$ varies from 1.0 to 0.4, the micro-F1 scores for both dimensions first increase and then decrease rapidly. This phenomenon is reasonable. When $\eta$ is large, a slightly smaller $\eta$ can include more terms to enrich the semantics of label embeddings. However, when $\eta$ is too small, noisy terms that are not very correlated with the label could be included and deteriorate the performance.

7.3 Case Study

In this subsection, we first examine the computed dimension-focal scores of different terms on the NYT dataset. For this purpose, we pick five terms in the vocabulary and show their dimension-focal scores in the topic and location dimensions in Table 2.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>economic</td>
<td>soccer</td>
</tr>
<tr>
<td>beijing</td>
<td>new york state</td>
</tr>
<tr>
<td>chinese consumer</td>
<td>0.999</td>
</tr>
</tbody>
</table>

From the results, we can see that: (1) The first two terms, “economic growth” and “soccer”, both have very high focal scores in the topic dimension but low scores in the location dimension. This is intuitive as these two terms are quite topic-indicative but do not naturally reflect the location of a given article. In the joint embedding procedure, these terms are emphasized when generating topic-aware representations and de-emphasized when generating location-aware representations. (2) The terms “beijing” and “new york state” are only discriminative for the location dimension. These terms do not carry topical semantics but are very useful signals for deciding the locations of news events. (3) There are also terms that have high focal scores in both the topic and location dimensions, such as “chinese consumer”. It makes sense as one can easily tell the topics and locations of news articles from such terms.

We proceed to demonstrate the empirical results of the label expansion component. In Table 3, we choose four labels in each of the topic and location dimensions and show the label expansion results in three rounds. The results clearly show why the label expansion is useful. Starting from the surface name of a label, Doc2Cube is capable of discovering other terms that are highly correlated with the label and include them for generating label embeddings. For example, for the label “movies” in the topic dimension, Doc2Cube iteratively discovers correlated terms such as “films”, “director”, and “hollywood”. Similarly, in the location dimension, Doc2Cube expands the label “China” by including terms like “chinese”, “beijing” and “shanghai”. One can imagine that, although many documents describing “China” may not explicitly use the term “china”, the label expansion component will enrich the semantic coverage of the label “China” and give high scores to those using “chinese”, “beijing” and “shanghai”. Such a property effectively reduces label sparsity and improves the text cube construction performance.

8 CONCLUSION

We proposed a novel method that automatically constructs a text cube from a text corpus to facilitate multidimensional text analytics.
Our proposed method, Doc2Cube, requires only the label names for document allocation. It leverages label names as weak supervision signals and iteratively performs joint embedding of labels, terms, and documents to uncover their semantic similarities. It features a document focalization component that learns dimension-aware document representations by selectively focusing on discriminative terms; as well as a label expansion component that propagates information from label names to other terms for alleviating label sparsity. Our experiments validate the effectiveness of Doc2Cube and its advantages over a comprehensive set of baseline methods.

**REFERENCES**


