Knowledge and Information Systems
https://doi.org/10.1007/s10115-018-1321-8

REGULAR PAPER

Learning concept embeddings for dataless classification via efficient bag-of-concepts densification

Walid Shalaby[1] · **Wlodek Zadrozny**[1]

Received: 8 December 2017 / Revised: 10 December 2018 / Accepted: 17 December 2018

Abstract
Explicit concept space models have proven efficacy for text representation in many natural language and text mining applications. The idea is to embed textual structures into a semantic space of concepts which captures the main ideas, objects, and the characteristics of these structures. The so-called bag-of-concepts (BoC) representation suffers from data sparsity causing low similarity scores between similar texts due to low concept overlap. To address this problem, we propose two neural embedding models to learn continuous concept vectors. Once they are learned, we propose an efficient vector aggregation method to generate fully continuous BoC representations. We evaluate our concept embedding models on three tasks: (1) measuring entity semantic relatedness and ranking where we achieve 1.6% improvement in correlation scores, (2) dataless concept categorization where we achieve state-of-the-art performance and reduce the categorization error rate by more than 5% compared to five prior word and entity embedding models, and (3) dataless document classification where our models outperform the sparse BoC representations. In addition, by exploiting our efficient linear time vector aggregation method, we achieve better accuracy scores with much less concept dimensions compared to previous BoC densification methods which operate in polynomial time and require hundreds of dimensions in the BoC representation.

Keywords Dataless classification · Concept categorization · Entity relatedness · Concept space models · Concept embeddings · Bag-of-concepts

1 Introduction

Vector space representation models are used to represent textual structures (words, phrases, and documents) as multidimensional vectors. Typically, those models utilize textual corpora and/or knowledge bases (KBs) in order to extract and model real-world knowledge. Once acquired, any given text is represented as a *vector* in the semantic space. The goal is thus to

✉ Walid Shalaby
wshalaby@uncc.edu

Wlodek Zadrozny
wzadrozn@uncc.edu

[1] Computer Science Department, University of North Carolina at Charlotte, Charlotte, NC 28223, USA

Published online: 17 January 2019

accurately place similar structures close to each other in that semantic space, while placing dissimilar ones far apart.

Explicit concept space models are one of these *vector-based representations* which are motivated by the idea that high level cognitive tasks such learning and reasoning are supported by the knowledge we acquire from *concepts*[1] [30]. Therefore, such models utilize concept vectors (aka bag of concepts (BoC)) as the underlying semantic representation of a given text through a process called *conceptualization*, which is mapping the text into relevant concepts capturing its main ideas, objects, events, and their characteristics. The concept space typically includes concepts obtained from KBs such as Wikipedia, Probase [35], and others. Once the concept vectors are generated, similarity between two concept vectors can be computed using a suitable similarity/distance measure such as *cosine*.

Similar to the traditional bag-of-words (BoW) representation, the BoC vector is a multidimensional *sparse* vector whose dimensionality is the same as the number of concepts in the employed KB (typically *millions*). Consequently, it suffers from *data sparsity* causing low similarity scores between similar texts due to low concept overlap. Formally, given a text snippet $T = \{t_1, t_2, \ldots, t_n\}$ of n terms where $n \geq 1$, and a concept space $C = \{c_1, c_2, \ldots, c_N\}$ of size N. The BoC vector $\mathbf{v} = \{w_1, w_2, \ldots, w_N\} \in \mathbb{R}^N$ of T is a vector of weights of each concept where each w_i of concept c_i is calculated as in Eq. 1:

$$w_i = \sum_{j=1}^{n} f(c_i, t_j), 1 \leq i \leq N \tag{1}$$

Here, $f(c, t)$ is a *scoring function* which indicates the degree of *association* between term t and concept c. For example, Gabrilovich and Markovitch [6] proposed explicit semantic analysis (ESA) which uses Wikipedia articles as concepts and the TF-IDF score of the terms in these article as the association score. Another scoring function might be the co-occurrence count or Pearson correlation score between t and c. As we can notice, only very small subset of the concept space would have nonzero scores with a given term[2]. Moreover, the BoC vector is generated from the top n concepts which have relatively high association scores with the input terms (typically few hundreds). Thus, each text snippet is mapped to a very sparse vector of millions of dimensions having only few hundreds nonzero values leading to the *BoC sparsity* problem [22].

Typically, the *cosine* similarity measure is used compute the similarity between a pair of BoC vectors $\mathbf{u}$ and $\mathbf{v}$. Because the concept vectors are very sparse and for space efficiency, we can rewrite each vector as a vector of tuples (c_i, w_i). Suppose that $\mathbf{u} = \{(c_{n_1}, u_1), \ldots, (c_{n_{|\mathbf{u}|}}, u_{|\mathbf{u}|})\}$ and $\mathbf{v} = \{(c_{m_1}, v_1), \ldots, (c_{m_{|\mathbf{v}|}}, v_{|\mathbf{v}|})\}$, where u_i and v_j are the corresponding weights of concepts c_{n_i} and c_{m_j}, respectively. And n_i, m_j are the indices of these concepts in the concept space C such that $1 \leq n_i, m_j \leq N$. Then, the similarity score can be written as in Eq. 2:

$$\text{Sim}_{\cos}(\mathbf{u}, \mathbf{v}) = \frac{\sum_{i=1}^{|\mathbf{u}|} \sum_{j=1}^{|\mathbf{v}|} \mathbb{1}(n_i = m_j) u_i v_j}{\sqrt{\sum_{i=1}^{|\mathbf{u}|} u_i^2} \sqrt{\sum_{j=1}^{|\mathbf{v}|} v_j^2}} \tag{2}$$

where $\mathbb{1}$ is the indicator function which returns 1 if $n_i = m_j$ and 0 otherwise. Having such sparse representation and using exact match similarity scoring measure, we can expect that two very similar text snippets might have *zero similarity* score if they map to *different but very related set of concepts* [28]. We demonstrate this fact in Table 1 (ESA column).

[1] A concept is an expression that denotes an idea, event, or an object.

[2] Unless the term is very common (e.g., a, the, some…etc) and carry no relevant information.

Table 1 Top-3 concepts generated using ESA [6] for two 20-newsgroups categories (Hockey and Guns) along with top-3 concepts of sample instances

Category	Top-3 concepts	Instance top-3 concepts	ESA	CCX	CRX
Hockey	• Detroit Red Wings • History of the Detroit Red Wings • History of the NHL on United States television	Instance (53,798) • History of the Detroit Red Wings • Detroit Red Wings • Pittsburgh Penguins	0.73	0.95	0.95
		Instance (54,551) • Paul Kariya • Boston Bruins • Bobby Orr	0.0	0.84	0.80
Guns	• Waco siege • Overview of gun laws by nation • Gun violence in the United States	Instance (54,387) • Overview of gun laws by nation • Waco siege • Gun politics in the United States	0.71	0.94	0.93
		Instance (54,477) • Concealed carry in the United States • Overview of gun laws by nation • Gun laws in California	0.33	0.80	0.75

Using exact match similarity scoring (as in ESA) results in low scores between similar instance and category concept vectors. When using concept embeddings (our models), we obtain relatively higher and more representative similarities

In this paper, we utilize *neural-based representations* to overcome the BoC sparsity problem. The basic idea is to *map* each concept to a *fixed size continuous vector*[3]. These vectors can then be used to compute concept–concept similarity and thus overcome the concept mismatch problem.

Our work is also motivated by the success of recent neural-based methods for learning word embeddings in capturing both syntactic and semantic regularities using simple vector arithmetic [17,18,23]. For example, inferring analogical relationships between words: *vec(king) − vec(man) + vec(woman) = vec(queen)*. This indicates that the learned vectors encode meaningful multi-clustering for each word.

However, word vectors suffer from significant limitations. First, each word is assumed to have a *single meaning* regardless of its context and thus is represented by a single vector in the semantic space (e.g., *charlotte (city)* vs. *charlotte (given name)*). Second, the space contains vectors of single words only. Vectors of multiword expressions (MWEs) are typically obtained by averaging the vectors of individual words. This often produces inaccurate representations, especially if the meaning of the MWE is different from the composition of meanings of its individual words (e.g., *vec(north carolina)* vs. *vec(north) + vec(carolina)*. Additionally, mentions that are used to refer to the same concept would have different embeddings (e.g., *u.s., america, usa*), and the model might not be able to place those individual vectors in the same sub-cluster, especially the rare surface forms.

[3] We use the terms continuous, dense, distributed vectors interchangeably to refer to real-valued vectors.

We propose two *neural embedding* models in order to learn continuous concept vectors based on the skip-gram model [18]. Our first model is the *Concept Raw Context* model (CRX) which utilizes raw concept mentions in a large scale textual KB to jointly learn embeddings of both words and concepts. Our second model is the *concept–concept context* model (CCX) which learns the embeddings of concepts from their conceptual contexts (i.e., contexts containing surrounding concepts only).

After learning the concept vectors, we propose an *efficient BoC aggregation* method. We perform *weighted average* of the individual concept vectors to generate fully *continuous* BoC representations (CBoC). This aggregation method allows measuring the similarity between pairs of BoC in *linear time* which is more efficient than previous methods that require *quadratic* or at least *log-linear* time if optimized (see Eq. 2). Our embedding models produce more *representative* similarity scores for BoC containing *different but semantically similar* concepts as shown in Table 1 (columns 2–3).

We evaluate our embedding models on three tasks:

1. An intrinsic task of measuring entity semantic relatedness and ranking where we achieve 1.6% improvement in correlation scores.
2. Dataless concept categorization where we achieve state-of-the-art performance and reduce the categorization error rate by more than 5% compared to five prior word and entity embedding models.
3. An extrinsic task of dataless document classification which is a learning *protocol* used to perform text categorization without the need for labeled data to train a classifier [4]. Experimental results show that we can achieve better accuracy using our efficient BoC densification method compared to the original sparse BoC representation with much less concept dimensions.

The contributions of this paper are fourfold: First, we propose two *low-cost* concept embedding models which learn concept representations from concept mentions in free-text corpora. Our models require few hours rather than days to train. Second, we show through empirical results the efficacy of the learned concept embeddings in measuring entity semantic relatedness and concept categorization. Our models achieve *state-of-the-art* performance on two concept categorization datasets. Third, we propose simple and efficient vector aggregation method to obtain *fully dense BoC in linear time*. Fourth, we demonstrate through experiments on dataless document classification that we can obtain better accuracy using the dense BoC representation with much less dimensions (few in most cases), reducing the *computational cost* of generating the BoC vector significantly.

The rest of this paper is organized as follows: Sect. 2 reviews related work; Sect. 3 describes our proposed embedding models; Sect. 4 introduces the applications of the proposed models; Sect. 5 reports experiments and results; and finally, Sect. 6 provides discussion and concluding remarks.

2 Related work

Text conceptualization Humans understand languages through multi-step cognitive processes which involves building rich models of the world and making multi-level generalizations from the input text [31]. One way of automating such generalizations is through text conceptualization: either by extracting basic level concepts from the input text using concept KBs [13,30], or mapping the whole input into a concept space that captures its semantics as in ESA [6] and MSA [26].

One major line of conceptualization research utilizes semi-structured KBs such as Wikipedia in order to construct the concept space which is defined by all Wikipedia article titles. Such models have proven efficacy for semantic analysis of textual data especially short texts where contextual information is missing or insufficient (see Shalaby and Zadrozny [26] for examples).

Another research direction uses more structured concept KBs such as Probase[4] [35]. Probase is a probabilistic KB of millions concepts and their relationships (basically is-a). It was created by mining billions of Web pages and search logs of Microsoft's Bing[5] repository using syntactic patterns. The concept KB was then leveraged for text conceptualization to support various text understanding tasks such as clustering of Twitter messages and News titles [29,30,30], search query understanding [33], short text segmentation [10,32], and term similarity [13,15].

Despite its effectiveness, the dependency of Probase on syntactic patterns can be a limitation especially for languages other than English. In addition, we expect augmenting and maintaining these syntactic patterns to be costly and labor intensive. We argue that concept embeddings allow simpler and more efficient representations, simply because similarity scoring between concept vectors can be performed using vector arithmetic. While the Probase hierarchy allows only symbolic matching which still suffers data sparsity. On another hand, we spotted some cases where Probase probabilities were atypical[6]. This is due to learning concept categories from a limited set of syntactic patterns which does not cover all concept mention patterns. Concept embeddings relax this problem by leveraging all concept mentions in order to learn the embedding vector and therefore might be utilized to curate such atypical Probase assertions.

Concept/entity embeddings Neural embedding models have been proposed to learn distributed representations of concepts/entities[7]. Song and Roth [28] proposed using the popular Word2Vec model [17] to obtain the embeddings of each concept by averaging the embeddings of the concept's individual words. For example, the embeddings of *Microsoft Office* would be obtained by averaging the vector of *Microsoft* and the vector of *Office* obtained from the Word2Vec model. Clearly, this method disregards the fact that the semantics of multiword concepts whose composite meaning is different from the semantics of their individual words.

More robust concept and entity embeddings can be learned from the general knowledge about the concept in encyclopedic KB (e.g., its article) and/or from the structure of a hyperlinked KB (e.g., its link graph). Such concept embedding models were proposed by Hu et al. [9], Li et al. [16], and Yamada et al. [36] who all utilize the skip-gram learning technique [18], but differ in how they define the context of the target concept.

Li et al. [16] extended the embedding model proposed by Hu et al. [9] by jointly learning entity and category embeddings from contexts defined by all other entities in the target entity article as well as its category hierarchy in Wikipedia. This method has the advantage of learning embeddings of both entities and categories jointly. However, defining the entity contexts as pairs of the target entity and all other entities appearing in its corresponding article might introduce noisy contexts, especially for long articles. For example, the Wikipedia article for "*United States*" contains links to "*Kindergarten*," "*First grade*," and "*Secondary school*" under the "*Education*" section.

[4] https://concept.research.microsoft.com.

[5] https://www.bing.com/.

[6] $p(\text{Arabic coffee} \mid \text{beverage}) = 0$.

[7] In this paper, we use the terms "concept" and "entity" interchangeably.

Yamada et al. [36] proposed a method based on the skip-gram model to jointly learn embeddings of words and entities using contexts generated from surrounding words of the target entity or word. The authors also proposed incorporating *Wikipedia* link graph by generating contexts from all entities with outgoing link to the target entity to better model entity–entity relatedness.

Our models also learn word and concept embeddings jointly. Mapping both words and concepts into the same semantic space allows us to easily measure word–word, word–concept, and concept–concept semantic similarities. In addition, our CRX model (described in Sect. 3) extends the context of each word/concept by including nearby concept mentions and not only nearby words. Therefore, we better model local contextual information of concepts and words in *Wikipedia*, treated as a textual KB. During training, we generate word–word, word–concept, concept–word, and concept–concept contexts (cf. Eq. 5). In Yamada et al. [36] model, concept–concept contexts are generated from *Wikipedia* link graph not from their raw mentions in *Wikipedia* text. In the CCX model, we define concept contexts by all surrounding concepts within a window of fixed size.

Generating contexts from raw text mentions makes our models scalable and *not restricted to hyperlinked encyclopedic textual corpora* only. This facilitates exploiting other free-text corpora with annotated concept mentions (e.g., news stories, scientific publications, medical guidelines…etc). Moreover, our proposed models are *computationally less costly* than Hu et al. [9] and Yamada et al. [36] models as they require few hours rather than days to train on similar computing resources.

BoC densification Densification of the bag of concepts (BoC) is the process of converting the sparse BoC into a continuous BoC (CBoC) (aka dense BoC) in order to overcome the BoC sparsity problem. The process requires first mapping each concept into a continuous vector using representation learning. Song and Roth [28] proposed three different mechanisms for aligning the concepts at different indices given a sparse BoC pair ($\mathbf{u}$, $\mathbf{v}$) in order to increase their similarity score.

The *many-to-many* mechanism works by averaging all pairwise similarities. The *many-to-one* mechanism works by aligning each concept in $\mathbf{u}$ with the most similar concept in $\mathbf{v}$ (i.e., its best match). Clearly, the complexity of these two mechanisms is *quadratic*. The third mechanism is the *one-to-one*. It utilizes the Hungarian method in order to find an optimal alignment on a one-to-one basis [20]. This mechanism performed the best on the task of dataless document classification and was also utilized by Li et al. [16].

However, the Hungarian method is a combinatorial optimization algorithm whose complexity is *polynomial*. Our proposed densification mechanism is more efficient than these three mechanisms as its complexity is *linear* with respect to the number of *nonzero* elements in the BoC. Additionally, it is simpler as it does not require tuning a cutoff threshold for the minimum similarity between two aligned concepts as in previous work. Figure 1 shows a schematic diagram of our efficient densification mechanism applied to a BoC generated from a *Wikipedia* inverted index. We simply perform weighted average of the individual concept vectors in the obtained BoC where concept weights correspond to the TF-IDF scores from searching *Wikipedia*.

3 Learning concept embeddings

A main objective of learning concept embeddings is to overcome the inherent problem of *data sparsity* associated with the BoC representation. Here, we try to learn continuous concept

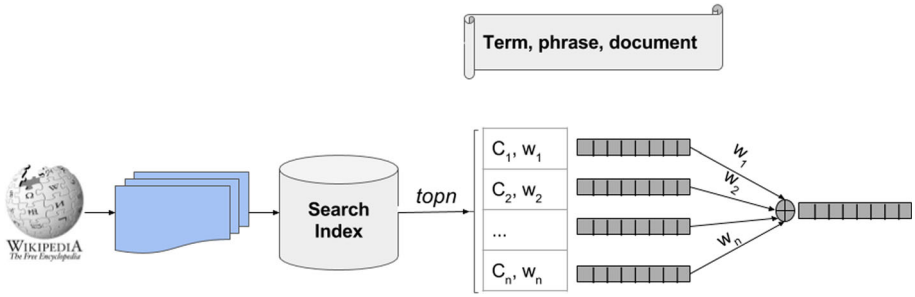

Fig. 1 Densification of the bag-of-concepts vector using weighted average of the learned concept embeddings. The concept space is defined by all Wikipedia article titles. The concept vector is created from the top n hits of searching a *Wikipedia* inverted index with the given text. The weights are the TF-IDF scores from searching *Wikipedia*

vectors by building upon the skip-gram embedding model [18]. In the conventional skip-gram model, a set of contexts are generated by sliding a context window of predefined size over sentences of a given text corpus. Vector representation of a target word is learned with the objective to maximize the ability of predicting surrounding words of that target word.

Formally, given a training corpus of V words $\omega_1, \omega_2, \ldots, \omega_V$. The skip-gram model aims to maximize the average log probability:

$$\frac{1}{V}\sum_{i=1}^{V}\sum_{-s\leq j\leq s, j\neq 0} \log\ p(\omega_{i+j}|\omega_i) \tag{3}$$

where s is the context window size, ω_i is the target word, and ω_{i+j} is a surrounding context word. The softmax function is used to estimate the probability $p(\omega_O|\omega_I)$ as follows:

$$p(\omega_O|\omega_I) = \frac{\exp(\mathbf{v}_{\omega_O}^{\mathsf{T}}\mathbf{u}_{\omega_I})}{\sum_{\omega=1}^{W}\exp(\mathbf{v}_{w}^{\mathsf{T}}\mathbf{u}_{\omega_I})} \tag{4}$$

where $\mathbf{u}_{\omega_I}$ and $\mathbf{v}_{\omega_O}$ are the input and output vectors, respectively, and W is the vocabulary size. Mikolov et al. [18] proposed hierarchical softmax and negative sampling as efficient alternatives to approximate the softmax function which becomes computationally intractable when W becomes huge.

Our approach genuinely learns distributed concept representations by generating concept contexts from *mentions* of those concepts in large encyclopedic KBs such as *Wikipedia*. Utilizing such annotated KBs eliminates the need to manually annotate concept mentions and thus comes at no cost.

3.1 Concept raw context model (CRX)

In this model, we jointly learn the embeddings of both words and concepts. First, all concept mentions are identified in the given corpus. Second, contexts are generated for both words and concepts from other surrounding words and other surrounding concepts as well. After generating all the contexts, we use the skip-gram model to jointly learn the embeddings of words and concepts. Formally, given a training corpus of V words $\omega_1, \omega_2, \ldots, \omega_V$, we iterate over the corpus identifying words and concept mentions and thus generating a sequence of

T tokens $t_1, t_2, \ldots t_T$ where $T < V$ (as multiword concepts will be counted as one token). Afterward, we train the a skip-gram model aiming to maximize:

$$\frac{1}{T}\sum_{i=1}^{T}\sum_{-s\leq j\leq s, j\neq 0} \log\ p(t_{i+j}|t_i) \tag{5}$$

whereas in the conventional skip-gram model, s is the context window size. In this model, t_i is the target token which would be either a word or a concept mention, and t_{i+j} is a surrounding context word or concept mention.

This model is different from Yamada et al. [36]'s anchor context model in three aspects: (1) while generating target concept contexts, we utilize not only surrounding words but also other surrounding concepts, (2) our model aims to maximize $p(t_{i+j}|t_i)$ where t could be a word or a concept, while Yamada et al. [36] model maximizes $p(\omega_{i+j}|e_i)$ where e_i is the target concept/entity (see Yamada et al. [36] Eq. 6), and (3) in case t_i is a concept, our model captures all the contexts in which it appeared, while Yamada et al. [36] model generates for each entity one context of s previous and s next words. We hypothesize that considering both concepts and individual words in the optimization function generates more robust embeddings.

3.2 Concept–concept context model (CCX)

Inspired by the distributional hypothesis [7], in this model, we hypothesize that: "*similar concepts tend to appear in similar conceptual contexts*." In order to test this hypothesis, we propose learning concept embeddings by training a skip-gram model on contexts generated solely from concept mentions. As in the CRX model, we start by identifying all concept mentions in the given corpus. Then, contexts of target concept are generated from surrounding concepts only. Formally, given a training corpus of V words $\omega_1, \omega_2, \ldots, \omega_V$, we iterate over the corpus identifying concept mentions and thus generating a sequence of C concept tokens $c_1, c_2, \ldots c_C$ where $C < V$. Afterwards, we train the skip-gram model aiming to maximize:

$$\frac{1}{C}\sum_{i=1}^{C}\sum_{-s\leq j\leq s, j\neq 0} \log\ p(c_{i+j}|c_i) \tag{6}$$

where s is the context window size, c_i is the target concept, and c_{i+j} is a surrounding concept mention within s mentions.

This model is different from Li et al. [16] and Hu et al. [9] as they define the context of a target concept by all the other concepts which have an outgoing link from the concept's corresponding article in *Wikipedia*.

Formally, given an article about concept c_t containing other concepts $(c_1, c_2, \ldots, c_n)$, Li et al. [16], Hu et al. [9] create context pairs in the form (c_t, c_i), $1 \leq i \leq n$. Thus, context size is limited to only 2 concepts.

Clearly, some of these concepts might be irrelevant, especially for very long articles which cite hundreds of other concepts. Our CCX model, alternatively, learns concept semantics from surrounding concepts and not only from those that are cited in its article. We also extend the context window beyond pairs of concepts (based on s in Eq. 6) allowing more influence to other nearby concepts.

Table 2 Example three sentences along with sample contexts generated from CRX and CCX. Contexts are generated with a context window of length 3

Sentence	CRX contexts	CCX contexts
Larry Page is the co-founder of ***Google*** which is headquartered in ***Menlo Park CA***	<*Larry Page*, co-founder> <co-founder, *Google*> <*Google*, headquartered> <headquartered, *Menlo Park CA*>	<*Larry Page*, *Google*> <*Larry Page*, *Menlo Park CA*> <*Google*, *Menlo Park CA*>
Bill Gates is the co-founder of ***Microsoft*** which is headquartered in ***Redmond WA***	<*Bill Gates*, co-founder> <co-founder, *Microsoft*> <*Microsoft*, headquartered> <headquartered, *Redmond WA*>	<*Bill Gates*, *Microsoft*> <*Bill Gates*, *Redmond WA*> <*Microsoft*, *Redmond WA*>
Google is headquartered in ***Menlo Park CA*** and was co-founded by ***Larry Page***	<*Google*, headquartered> <headquartered, *Menlo Park CA*> <*Menlo Park CA*, co-founded> <co-founded, *Larry Page*>	<*Google*, *Menlo Park CA*> <*Google*, *Larry Page*> <*Menlo Park CA*, *Larry Page*>

3.3 CRX versus CCX

One of the advantages of the CCX model over the CRX model is its computational efficiency during learning. On the other hand, the CCX model vocabulary is *limited to the corpus concepts* (all *Wikipedia* articles in our case), while the CRX model vocabulary is defined by all *unique concepts + words* in *Wikipedia*.

Another distinct property of the CCX model is its emphasis on concept–concept *relatedness rather than similarity* (as we will detail more in the experiments section). The CCX model by looking only at surrounding concept mentions while learning, is able to generate contexts containing more diverse but related concepts. On the other hand, the CRX model which jointly learns the embeddings of words and concepts puts more *emphasis on similarity* by leveraging the full contextual information of words and concepts while learning.

To better illustrate this difference, consider a sample of the contexts generated from CRX and CCX in Table 2 using a sliding window of length 3. As we can notice, the CRX contexts of "*Google*" and "*Microsoft*" are somewhat similar containing words like "*headquartered*" and "*co-founder*." This causes the model to learn similar vectors for these two concepts. On the other hand, the CCX contexts of "*Google*" and "*Microsoft*" do not share any similarities[8], rather we can see that "*Google*" has similar contexts to "*Larry Page*" as both has "*Menlo Park CA*" in their contexts, causing the model to learn similar embeddings for these two related concepts.

3.4 Training

We utilize a recent *Wikipedia* dump of August 2016[9], which has about 7 million articles. We extract articles plain text discarding images and tables. We also discard "References" and "External links" sections (if any). We pruned both articles not under the main namespace and

[8] This is an illustrative example and doesn't imply the two concepts will have totally dissimilar vectors.

[9] http://dumps.wikimedia.org/enwiki/.

pruned all redirect pages as well. Eventually, our corpus contained about 5 million articles in total.

We preprocess each article replacing all its references to other *Wikipedia* articles with the their corresponding page IDs. In case any of the references is a title of a redirect page, we use the page ID of the original page to ensure that all concept mentions are normalized.

Following Mikolov et al. [18], we utilize negative sampling to approximate the softmax function by replacing every $\log\ p(\omega_O|\omega_I)$ term in the softmax function (Eq. 3) with:

$$\log \sigma(\mathbf{v}_{\omega_O}^{\intercal}\mathbf{u}_{\omega_I}) + \sum_{s=1}^{k} \mathbb{E}_{\omega_s \sim P_n(w)}[\log \sigma(-\mathbf{v}_{\omega_s}^{\intercal}\mathbf{u}_{\omega_I})] \tag{7}$$

where k is the number of negative samples drawn for each word and $\sigma(x)$ is the sigmoid function ($\frac{1}{1+e^{-x}}$). In the case of the CRX model, ω_I and ω_O would be replaced with t_i and t_{i+j}, respectively. And in the case of the CCX model, ω_I and ω_O would be replaced with c_i and c_{i+j} respectively.

For both the CRX & CCX models with use a context window of size 9 and a vector of 500 dimensions. We train the skip-gram model for 10 iterations using 12-core machine with 64GB of RAM. The CRX model took $\sim$ 15 h to train for a total of $\sim$12.7 million tokens. The CCX model took $\sim$1.5 h to train for a total of $\sim$4.5 million concepts.

3.5 BoC densification

As we mentioned in the related work section, the current mechanisms for BoC densification are inefficient as their complexity is at least quadratic with respect to the number of nonzero elements in the BoC vector. Here, we propose simple and efficient vector aggregation method to obtain fully continuous BoC vectors (CBoC) in linear time. Our mechanism works by performing a weighted average of the individual concept vectors in a given BoC. This operation has two advantages. First, it *scales linearly* with the number of nonzero dimensions in the BoC vector. Secondly, it produces a fully dense BoC vector of *fixed size* representing the semantics of the original concepts and *considering their weights*. Formally, given a sparse BoC vector $\mathbf{s} = \{(c_1, w_1), \ldots, (c_{|\mathbf{s}|}, w_{|\mathbf{s}|})\}$, where w_i is weight of concept c_i[10], We can obtain the dense representation of $\mathbf{s}$ as in Eq. 8:

$$\mathbf{s}_{\text{dense}} = \frac{\sum_{i=1}^{|\mathbf{s}|} w_i . \mathbf{u}_{c_i}}{\sum_{i=1}^{|\mathbf{s}|} w_i} \tag{8}$$

where $\mathbf{u}_{c_i}$ is the vector of concept c_i. Once we have this dense BoC vector, we can apply the cosine measure to compute the similarity between a pair of dense BoC vectors.

As we can notice, this weighted average is done *once* for any given BoC vector. Other mechanisms that rely on concept alignment [28] require *realignment* every time a pair of BoC vectors are compared. Our approach improves the *efficiency*, especially in the context of dataless document classification with large number of classes. Using our densification mechanism, we apply the weighted average for the BoC of each category and for each instance document once.

Interestingly, our densification mechanism allows us to densify the sparse BoC vector using only the *top few dimensions*. As we will show in the experiments, we can get *near-best* results using these few dimensions compared to densifying with all the dimensions in the

[10] The weights are the TF-IDF scores from searching *Wikipedia*.

original sparse vector. This property reduces the cost of obtaining a BoC vector with a few hundred dimensions in the first place.

4 Text conceptualization applications

Concept-based representations have many applications in computational linguistics, information retrieval, and knowledge modeling. Such representations are able to capture the semantics of a given text by either identifying concept mentions in that text, transforming the text into a concept space, or both [31]. Thereafter, many cognitive tasks that require huge background and real-world knowledge are facilitated by leveraging the conceptual representations. We describe some of these tasks in this section and provide empirical evaluation of our concept embedding models on such tasks in the next section.

4.1 Concept/entity relatedness

Entity relatedness has been recently used to model *entity coherence* in many named entity linking and disambiguation systems [3,8,9,11,19,34,36]. In entity search, Hu et al. [9] utilized entity relatedness score to *rank* candidate entities based on their relatedness to the search query entities. Also, entity embeddings have proved more efficient and effective for measuring entity relatedness over traditional relatedness measures which use link analysis. Formally, given a entity pair (e_i, e_j), their relatedness score is evaluated as $\text{rel}(e_i, e_j) = \text{Sim}(\mathbf{u}_{e_i}, \mathbf{u}_{e_j})$, where Sim is a similarity function (e.g., *cosine*), and $\mathbf{u}_e$ is the embeddings of entity e.

4.2 Concept learning

Concept learning is a cognitive process which involves classifying a given concept/entity to one or more candidate categories (e.g., *milk* as *beverage, dairy product, liquid*…etc). This process is also known as *concept categorization*[11] [16]. Automated concept learning gains its importance in many knowledge modeling tasks such as knowledge base *construction* (discovering new concepts), *completion* (inferring new relationships between concepts), and *curation* (removing noisy or assessing weak relationships). Similar to Li et al. [16], we assign a given concept to a target category using Rocchio classification [24], where the centroid of each category is set to the category's corresponding embedding vector. Formally, given a set of n candidate concept categories $G = \{g_1, \ldots, g_n\}$, a sample concept c, an embedding function f, and a similarity function Sim, then c_i is assigned to category g_* such that $g_* = \arg\max_i \text{Sim}(f(g_i), f(c))$. Here, the embedding function f would always map the given concept to its vector.

4.3 Dataless classification

Chang et al. [4] proposed dataless document classification as a learning *protocol* to perform text categorization without the need for labeled data to train a classifier. Given only label names and few descriptive keywords of each label, classification is performed *on the fly* by mapping each label into a BoC representation using ESA [6]. Likewise, each data instance is mapped into the same BoC semantic space and assigned to the most similar

[11] In this paper, we use concept learning and concept categorization interchangeably.

Algorithm 1: Classification + Bootstrapping

Input: $\mathbf{U} = \{(l_1, \mathbf{u}_{l_1}), \ldots, (l_n, \mathbf{u}_{l_n})\}$: labels + embeddings
$\mathbf{D} = \{(d_1, \mathbf{v}_{d_1}), \ldots, (d_m, \mathbf{v}_{d_m})\}$: instances + embeddings
N: number of bootstrap instances
Result: $\mathbf{L} = \{\ldots, (d_i, l_j), \ldots\}$: label assignment for each instance

repeat
  candidates $\leftarrow \{l_1 : \phi, \ldots, l_n : \phi\}$
  foreach $(d, \mathbf{v}_d) \in \mathbf{D}$ **do**
    $d_{max\_sim} = 0$
    $d_{max\_label} = null$
    foreach $(l, \mathbf{u}_l) \in \mathbf{U}$ **do**
      $sim_l = Sim(\mathbf{v}_d, \mathbf{u}_l)$
      if $sim_l > d_{max\_sim}$ **then**
        $d_{max\_sim} = sim_l$
        $d_{max\_lebel} = l$
      end
    end
    add (d, d_{max_sim}) to **candidates**$[l]$
  end
  foreach $(l, \textbf{candidates}_l) \in \textbf{candidates}.items$ **do**
    repeat
      $score_{max} = 0$
      $d_{max} = null$
      foreach $(d, score_d) \in \textbf{candidates}_l$ **do**
        if $score_d > score_{max}$ **then**
          $score_{max} = score_d$
          $d_{max} = d$ ▷ most similar instance so far
        end
      end
      add (d_{max}, l) to $\mathbf{L}$ ▷ assign class label
      $\mathbf{u}_l \leftarrow \mathbf{u}_l + \mathbf{v}_d$ ▷ bootstrap label embedding
      remove d from **candidates**$_l$
      remove d from $\mathbf{D}$
    until *N highest scored instances added*
  end
until $\mathbf{D} = \phi$ ▷ *no more instances to classify*

label using a proper similarity measure such as *cosine*. Formally, given a set of n labels $L = \{l_1, \ldots, l_n\}$, a text document d, a BoC mapping model f, and a similarity function Sim. First, we conceptualize the each l_i and the document d by applying f on them, which will produce sparse BoC vectors s_{l_i} and s_d, respectively. Then, we densify the vectors as in Eq. 8 producing $s_{\text{dense}_{l_i}}$ and s_{dense_d}, respectively. Finally, d is assigned to label l_* such that $l_* = \arg\max_i \text{Sim}(s_{\text{dense}_{l_i}}, s_{\text{dense}_d})$.

In the context of dataless classification, Chang et al. [4] and Song and Roth [27] used bootstrapping in order to improve the classification performance without the need for labeled data. The basic idea is to start from target labels as the initial training samples, train a classifier, and *iteratively* add to the training data those samples which the classifier is *most confident* until no more samples to be classified. The results of dataless classification with bootstrapping were competitive to supervised classification with many training examples.

In this paper, we extend the use of bootstrapping to the concept learning task as well. In concept learning, we start with the vectors of target category concepts as a prototype view upon which categorization decisions are made (e.g., *vec(bird), vec(mammal)…etc*). We leverage bootstrapping by *iteratively updating* this prototype view with the vectors of concept instances we are most confident. For example, if "*deer*" is closer to "*mammal*" than any other instance in the dataset, then we update the definition of *"mammal"* by performing *vec(mammal) += vec(deer)*, and repeat the same operation for other categories as well. This way, we *adapt* the initial prototype view to better match the specifics of the given data.

Although bootstrapping is a time-consuming process, we argue that using dense vectors for representing concepts makes bootstrapping more appealing. This is because updating the category vector with an instance vector could be performed through optimized vector arithmetic which is available in most modern machines. Algorithm 1 presents the pseudocode for performing dataless classification and concept categorization with bootstrapping. In our implementation, we bootstrap the category vector with vectors of the most similar **N** instances at a time. Another implementation option might be defining a threshold and bootstrapping using vectors of **N** instances if their similarity score exceeds that threshold. In the experiments, we set $\mathbf{N} = 1$.

5 Experiments

5.1 Entity semantic relatedness

We evaluate the "goodness" of our concept embeddings on measuring entity semantic relatedness as an intrinsic evaluation.

5.1.1 Dataset

We use the KORE dataset created by Hoffart et al. [8]. It consists of 21 main entities from four domains: IT companies, Hollywood celebrities, video games, and television series. For each of these entities, 20 other candidate entities were selected and manually ranked based on their relatedness score based on human judgements. As in previous studies, we report the Spearman rank-order correlation (ρ) [37] which assesses how the automated ranking of candidate entities based on their relatedness score matches the ranking we obtain from human judgements.

5.1.2 Compared systems

We compare our models with four previous methods:

1. *KORE* [8] which measures entity relatedness by firstly representing entities as sets of weighted keyphrases and then computing relatedness using different measures such as keyphrase vector cosine similarity and keyphrase overlap relatedness.
2. *WLM* introduced by Witten and Milne [34] who proposed a Wikipedia Link-based Measure (WLM) as a simple mechanism for modeling the semantic relatedness between *Wikipedia* entities. The authors utilized *Wikipedia* link structure under the assumption that related entities would have similar incoming links.
3. *Exclusivity-based Relatedness (ExRel)* introduced by Hulpus et al. [12] who proposed this measure under the assumption that not all instances of a given relation type should be equally weighted. Specifically, the authors hypothesized that the relatedness score between two concepts should be higher if each of them is related through the same relation type to fewer other concepts in the employed KB link graph.
4. *Combined Information Content (CombIC)* introduced by Schuhmacher and Ponzetto [25] who compute the relatedness score using a graph edit distance measure on the *DBpedia KB*.

Table 3 Evaluation of concept embeddings for measuring entity semantic relatedness using Spearman rank-order correlation (ρ)

Method	IT companies	Celebrities	TV series	Video games	Chuck Norris	All
WLM	0.721	0.667	0.628	0.431	0.571	0.610
CombIC	0.644	0.690	0.643	0.532	0.558	0.624
ExRel	0.727	0.643	0.633	0.519	0.477	0.630
KORE	0.759	**0.715**	0.599	**0.760**	0.498	0.698
CRX	0.644	0.592	0.511	0.641	0.495	0.586
CCX	**0.788**	0.694	**0.696**	0.708	**0.573**	**0.714**

Overall, the CCX model gives the best results outperforming all other models. It comes 1st on 3 categories (bold), and 2nd on the other two (underlined)

5.1.3 Results

Table 3 shows the Spearman (ρ) correlation scores of the CRX and CCX model compared to previous models. As we can notice, the CCX model achieves the best overall performance on the five domains combined exceeding its successor KORE by 1.6%. The CRX model on the other hand came last on this task.

In order to better understand these results, we looked at rankings of individual entities from each domain to see how they compare to the ground truth. Table 4 shows the top-3-rated entities from each model on sample entities from the four domains. As we can notice, the ground truth assigns high rank to related rather than similar entities. For example, relatedness of "*Google*" to "*Larry Page*" is ranked 1st, while to "*Yahoo!*" is ranked 9th, and to "*Apple Inc.*" is ranked 12th. As the CCX model emphasizes semantic relatedness over similarity, it has high overlap in the top-3 entities with the ground truth (underlined entities). On the other hand, the CRX model predictions are actually meaningful when it comes to functional and topical similarity. As we can notice, it assigns high ranks of "*Google*" to other companies ("Yahoo!" and "Apple Inc."), of "Leonardo DiCaprio" to other celebrities ("*Tobey Maguire*"), and "*Mad Men*" to other TV series ("*The Sopranos*"), and of "*Guitar Hero*" to other video games ("*Frequency*," "*Rock Band*"). However, all these highly ranked entities by CRX have relatively low rankings in the ground truth (given in brackets). This caused the correlation score to be much lower than what we obtained from the CCX model.

The results indicate that the CCX model could be more appropriate in applications where relatedness and topical diversity are more desired than topical and functional coherence where the CRX model would be more appealing.

5.2 Concept categorization

This task can be viewed as both intrinsic and extrinsic. It is intrinsic because a *good* embedding model would generate clusters of concepts belonging to the same category and optimally place the category vector at the center of its instances vectors. On another hand, it is extrinsic as the embedding model could be used to generate a concept KB of is-a relationships with confidence scores, similar to *Probase* [35]. The model could even be used to curate and/or assert the facts in *Probase*.

Table 4 Top-3 rated entities from CRX & CCX models on sample entities from the 4 domains compared to the ground truth

Entity	CRX	CCX	Ground truth (GT)
Google	Yahoo! (9)	Larry Page (1)	Larry Page
	Apple Inc. (12)	Sergey Brin (2)	Sergey Brin
	Bing (search engine) (7)	Yahoo! (9)	Google Maps
Leonardo DiCaprio	Kate Winslet (4)	Tobey Maguire (7)	Inception (film)
	Steven Spielberg (9)	Kate Winslet (4)	Titanic (1997 film)
	Tobey Maguire (7)	Titanic (1997 film) (2)	Frank Abagnale
Mad Men	The Sopranos (15)	Matthew Weiner (1)	Matthew Weiner
	Matthew Weiner (1)	Jon Hamm (2)	Jon Hamm
	Jon Hamm (2)	Todd London (4)	Alan Taylor (director)
Guitar Hero (video game)	Frequency (video game) (10)	Harmonix Music Systems (1)	Harmonix Music Systems
	Rock Band (video game) (6)	WaveGroup Sound (3)	RedOctane
	Harmonix Music Systems (1)	RedOctane (1)	WaveGroup Sound

We can notice high agreement between CCX model ranks and the ground truth ranks (in brackets). The CRX model top-rated entities have lower ranks than ground truth ranks causing relatively low correlation scores

5.2.1 Datasets

As in Li et al. [16], we utilize two benchmark datasets: (1) Battig test [1], which contains 83 single-word concepts (e.g., *cat, tuna, spoon…etc*) belonging to 10 categories (e.g., *mammal, fish, kitchenware…etc*), and (2) DOTA which was created by Li et al. [16] from *Wikipedia* article titles (entities) and category names (categories). DOTA contains 300 single-word concepts (DOTA-single) (e.g., *coffee, football, semantics…etc*), and (150) multiword concepts (DOTA-mult) (e.g., *masala chai, table tennis, noun phrase…etc*). Both belong to 15 categories (e.g., *beverage, sport, linguistics…etc*). Performance is measured in terms of the ability of the system to assign concept instances to their correct categories.

5.2.2 Compared systems

We compare our models to various word, entity, and category embedding methods as described in Li et al. [16] including:

1. *Word embeddings* Collobert et al. [5] model (WE_{Senna}) trained on *Wikipedia*. Here, vectors of multiword concepts are obtained by averaging their individual word vectors.
2. *MWEs embeddings* Mikolov et al. [18] model ($WE_{Mikolov}$) trained on *Wikipedia*. This model jointly learns single- and multiword embeddings where MWEs are identified using corpus statistics.
3. *Entity-category embeddings* which include Bordes et al. [2] embedding model (TransE). This model utilizes relational data between entities in a KB as triplets in the form (entity,relation,entity) to generate representations of both entities and relationships. Li et al. [16] implemented three variants of this model ($TransE_1$, $TransE_2$, $TransE_3$) to generate representations for entities and categories jointly. Two other models introduced by Li et

Table 5 Accuracy of concept categorization

Dataset/instances method	Battig (83)	DOTA-single (300)	DOTA-mult (150)	DOTA-all (450)
WE_{Senna}	0.44	0.52	0.32	0.45
$WE_{Mikolov}$	0.74	0.72	0.67	0.72
$TransE_1$	0.66	0.72	0.69	0.71
$TransE_2$	0.75	0.80	0.77	0.79
$TransE_3$	0.46	0.55	0.52	0.54
CE	0.79	0.89	0.85	0.88
HCE	0.87	0.93	0.91	0.92
CCX	0.72	0.90	0.80	0.87
+ bootstrap	0.81	0.91	0.85	0.87
CRX	0.83	0.91	0.88	0.90
+ bootstrap	**0.89**	**0.98**	**0.95**	**0.97**

The CRX model with bootstrapping gives the best results outperforming all other models
Bold values indicate best performance

al. [16] are CE and HCE. CE generates embeddings for concepts and categories using category information of *Wikipedia* articles. HCE extends CE by incorporating *Wikipedia*'s category hierarchy while training the model to generate concept and category vectors.

5.2.3 Results

We report the accuracy scores of concept categorization[12] in Table 5. Accuracy is calculated by dividing the number of correctly classified concepts by the total number of concepts in the given dataset. Scores of all other methods are obtained from Li et al. [16]. As we can see in Table 5, the CRX model comes second after the HCE on all datasets, while the CCX model performance is much less than CRX. With bootstrapping, the CCX model performance improves on both datasets. CRX with bootstrapping outperforms all other models by significant percentages. These results show that learning concept embeddings from concept mentions is actually different from training the skip-gram model on phrases or multiword expressions. This is clear from the significant performance gains we get from the CRX and CCX models compared to $WE_{Mikolov}$ which was trained using skip-gram on phrases. Additionally, the results demonstrate the efficacy of our models which simply learn concept embeddings from concept mentions in free-text corpus compared to the more complex models which require category or relational information such as TransE, CE, and HCE.

5.3 Dataless classification

In this experiment, we evaluate the effectiveness of our concept embedding models on the dataless document classification task as an extrinsic evaluation. We demonstrate through empirical results the efficiency and effectiveness of our proposed BoC densification scheme

[12] From a multi-class classification perspective, the accuracy scores would be equivalent to the clustering purity score as reported in Li et al. [16].

Table 6 20NG category mappings

Top-level	Low-level
Sport	Hockey, Baseball, Autos, Motorcycles
Politics	Guns, Mideast, Misc
Religion	Christian, Atheism, Misc

which helps obtaining better classification results compared to the original sparse BoC representation.

5.3.1 Dataset

We use the 20-newsgroups dataset (20NG) [14] which is commonly used for benchmarking text classification algorithms. The dataset contains 20 categories; each has ∼ 1000 news posts. We obtained the BoC representations using ESA from Song and Roth [27] who utilized a Wikipedia index containing pages with 100 + words and 5 + outgoing links to create ESA mappings of 500 dimensions for both the categories and news posts of the 20NG. We designed two types of classification tasks: (1) fine-grained classification involving closely related classes such as *Hockey* versus *Baseball*, *Autos* versus *Motorcycles*, and *Guns* versus *Mideast* versus *Misc*, and (2) coarse-grained classification involving top-level categories such as *Sport* versus *Politics* and *Sport* versus *Religion*. The top-level categories are created by combining instances of the fine-grained categories which are shown in Table 6.

5.3.2 Compared systems

We compare our models to three previous methods:

1. *ESA* which computes the cosine similarity between target labels and instance documents using the sparse BoC vectors.
2. WE_{max} & WE_{hung} which were proposed by Song and Roth [28] for BoC densification using embeddings obtained from Word2Vec. As the authors reported, we fix the minimum similarity threshold to 0.85. WE_{max} finds the best match for each concept, while WE_{hung} utilizes the Hungarian algorithm to find the best concept–concept alignment on one-to-one basis. Both mechanisms have polynomial degree time complexity.

5.3.3 Results

Table 7 presents the results of fine-grained dataless classification measured in micro-averaged F1. As we can notice, ESA achieves its peak performance with a few hundred dimensions of the sparse BoC vector. Using our densification mechanism, both the CRX & CCX models achieve equal performance to ESA at many fewer dimensions. Densification using the CRX model embeddings gives the best F1 scores on the three tasks. Interestingly, the CRX model improves the F1 score by ∼ 7% using only 14 concepts on *Autos* versus *Motorcycles*, and by ∼ 3% using 70 concepts on *Guns* versus *Mideast* versus *Misc*. The CCX model still performs better than ESA on 2 out of the 3 tasks. Both WE_{max} and WE_{hung} improve the performance over ESA, but not as our CRX model.

When we applied bootstrapping, the performance of the CCX model improved slightly on *Hockey* versus *Baseball*, but significantly (∼ 5%) on the other two tasks achieving best performance on the third task with just 5 concepts. Bootstrapping with the CRX model has a similar

Table 7 Evaluation results of dataless document classification of fine-grained classes measured in micro-averaged F1 along with # of dimensions (concepts) in the BoC at which corresponding performance is achieved

Method	Hockey × Baseball		Autos × Motorcycles		Guns × Mideast × Misc	
ESA	94.60	@425	72.70	@325	70.00	@500
CCX (equal)	94.60	@20	–	–	70.33	@60
CRX (equal)	94.60	@60	73.10	@4	70.00	@7
WE_{max}	86.85	@65	76.15	@375	72.20	@300
WE_{hung}	95.20	@325	73.75	@300	71.70	@275
CCX (best)	95.10	@125	69.70	@7	72.47	@250
+ bootstrap	95.90	@450	74.25	@12	**77.43**	@5
CRX (best)	95.65	@425	**79.20**	@14	73.40	@70
+ bootstrap	**95.90**	@350	73.25	@12	77.03	@10

Bold values indicate best performance

effect to the CCX model except for *Autos* versus *Motorcycles* where performance degraded significantly. To better understand this behavior, we analyzed the results as bootstrapping progresses at 14 concepts like CRX (best). We noticed that, at the very early iterations of Algorithm 1, many instances belonging to *Autos* were closer to *Motorcycles* with similarity scores between 0.90–0.95. And when using those instances to bootstrap *Motorcycles*, they caused *topic drift* moving *Motorcycles*' centroid toward *Autos*, and eventually causing relatively lower accuracy scores.

In order to better illustrate the robustness of our densification mechanism when varying the number of BoC dimensions, we measured F1 scores of each task as a function of the number of BoC dimensions used for densification. As we see in Fig. 2, with *one* concept we can achieve high F1 scores compared to ESA which achieves *zero* or very low score. Moreover, *near-peak performance* is achievable with the top 50 or less dimensions. We can also notice that, as we increase the number of dimensions, both WE_{max} and WE_{hung} densification methods have the same undesired monotonic pattern like ESA. Actually, the imposed threshold by these methods does not allow for full dense representation of the BoC vector and therefore at low dimensions we still see low overall F1 score. Our proposed densification mechanism besides its low cost produces fully densified representations allowing good similarities at low dimensions.

Results of coarse-grained classification are presented in Table 8. Classification at the top level is easier than the fine-grained level. Nevertheless, as with fine-grained classification, ESA still peaks with a few hundred dimensions of the sparse BoC vector. Both the CRX & CCX models achieve equal performance to ESA at very few dimensions ($\leq$ 6). Densification using the CRX model embeddings still performs the best on both tasks. Interestingly, the CCX model gives very close F1 scores to the CRX model at less dimensions (@4 with *Sport* versus *Politics*, and @60 with *Sport* versus *Religion*) indicating its competitive advantage when training computational cost is a decisive criteria. The CCX model still performs better than ESA, WE_{max}, and WE_{hung} on both tasks.

Bootstrapping did not improve the results on this task significantly (if any). As we can notice in Table 8, the accuracy without bootstrapping is already high indicating that the initial prototype vector (centroid) of each class is representative enough of the instances to be classified.

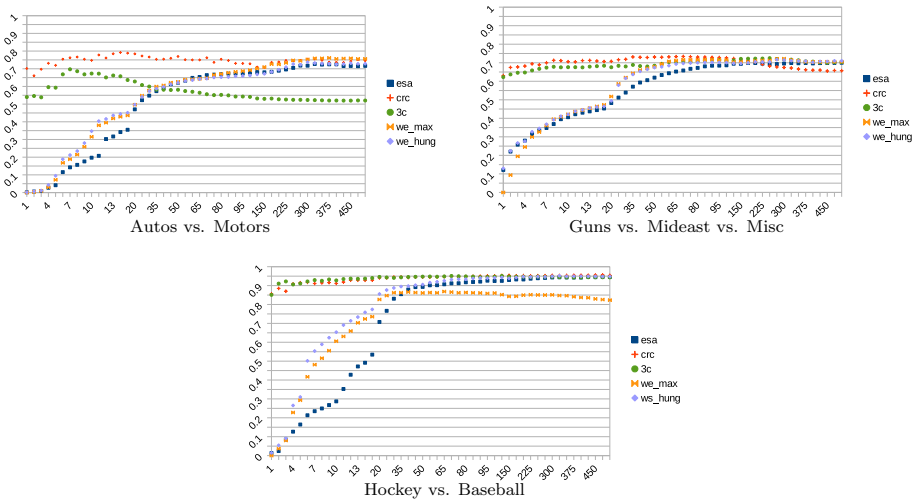

Fig. 2 Micro-averaged F1 scores of fine-grained classes when varying the # of BoC dimensions (color figure online)

Table 8 Evaluation results of dataless document classification of coarse-grained classes measured in micro-averaged F1 along with # of dimensions (concepts) at which corresponding performance is achieved

Method	Sport × Politics		Sport × Religion	
ESA	90.63	@425	94.39	@450
CCX (equal)	92.04	@2	95.11	@6
CRX (equal)	90.99	@2	94.81	@5
WE_{max}	91.89	@425	93.99	@425
WE_{hung}	90.89	@275	94.16	@450
CCX (best)	92.89	@4	95.86	@60
+ bootstrap	**93.20**	@10	95.13	@225
CRX (best)	93.12	@13	**95.91**	@95
+ bootstrap	92.96	@13	95.53	@70

Bold values indicate best performance

Figure 3 shows F1 scores of coarse-grained classification when varying the # of BoC dimensions used for densification. The same pattern of achieving *near-peak performance* at very few dimensions recur with the CRX & CCX models. ESA using the sparse BoC vectors achieves low F1 up until few hundred dimensions are considered. Even with the costly WE_{max} and WE_{hung} densifications, performance sometimes decreases.

5.3.4 Dataless versus supervised classification

We performed a pilot experiment to demonstrate the value of the dataless classification scheme in the absence or difficulty of obtaining labeled data for training a supervised classifier. For this purpose, we used a Support Vector Machine (SVM) classifier with a linear kernel, leveraging the scikit-learn machine learning library [21] to perform classification of fine-grained classes (cf. Table 6). We trained the SVM classifier with labeled samples ranging between 10 and

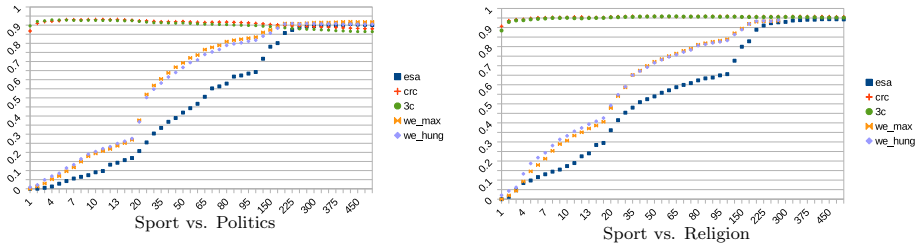

Fig. 3 Micro-averaged F1 scores of coarse-grained classes when varying the # of concepts (dimensions) in the BoC from 1 to 500 (color figure online)

Table 9 Evaluation results of dataless document classification of coarse-grained classes versus supervised classification with SVM measured in micro-averaged F1 along with # of dimensions (concepts) for CRX & CCX or % of labeled samples for SVM at which corresponding performance is achieved

Method	Hockey × Baseball		Autos × Motorcycles		Guns × Mideast × Misc	
SVM	91.61	@85%	79.25	@20%	77.56	@25%
CCX (best)	95.90	@450	74.25	@12	77.43	@5
CRX (best)	95.90	@350	79.20	@14	77.03	@10

90% of the total number of samples for each task and evaluate the performance on the rest. The results in Table 9 show the % of labeled sampled needed for training SVM in order to achieve equal performance to dataless classification with CRX and CCX. As we can notice, with *Hockey* versus *Baseball*, the SVM classifier cannot reach the same performance as either models and peaks when trained on 85% (∼1700 samples) of the data. With the *Autos* vs *Motorcycles* and *Guns* versus *Mideast* versus *Misc*, the SVM classifier achieves equal performance when trained on 20% (∼400 samples) and 25% (∼750 samples) of the data, respectively. These results demonstrate the competitiveness of our models to supervised classification even when training data are available, and its superiority when training data are scarce.

6 Discussion and conclusion

In this paper, we proposed two models for learning neural embeddings of explicit concepts based on the skip-gram model. Explicit concepts are lexical expressions (single or multi-words) that denote an idea, event, or an object and typically have a set of properties associated with it. In the models presented here, our concept space is the set of all *Wikipedia* article titles. We proposed learning concept representations from concept mentions/references in *Wikipedia* making our models applicable to other open domain and domain specific free-text corpora by firstly wikifying[13] the text and then learning from concept mentions.

It is clear from the presented results that the CRX model outperforms the CCX model on tasks that require topical coherence among the concepts vectors (e.g. concept categorization), while the CCX model is advantageous in tasks that require topical relatedness (e.g., measuring entity relatedness). To better show this difference qualitatively, we present a qualitative

[13] Wikification is the process of identifying mentions of concepts and entities in a given free-text and linking them to *Wikipedia*.

Table 10 Top-5 related concepts from CRX & CCX models for sample target concepts

Concept	Concept raw context (CRX)	Concept–concept context (CCX)
YouTube	Vevo	Viral video
	Facebook	Vimeo
	SoundCloud	Vevo
	Vimeo	Video blog
	Viral video	Dailymotion
Harvard University	Yale University	Harvard Kennedy School
	Princeton University	Cambridge, Massachusetts
	Brown University	Harvard College
	Columbia University	Radcliffe College
	Boston University	Harvard Society of Fellows
Black hole	Neutron star	Event horizon
	Accretion disk	Neutron star
	Primordial black hole	Gravitational singularity
	Supermassive black hole	Wormhole
	Event horizon	Hawking radiation
X-Men: Days of Future Past	X-Men: Apocalypse	X-Men: Apocalypse
	X-Men: First Class	The Wolverine (film)
	Deadpool (film)	X-Men: First Class
	Avengers: Age of Ultron	John Paesano
	X-Men: The Last Stand	William Stryker

analysis of both models in Table 10 (target concepts are similar to those reported by Hu et al. [9]).

As we can notice, the CRX model tends to emphasize concept *topical and categorical similarity*, while the CCX model tends to more emphasize *concept relatedness*. For example, the top-5 concepts closest to "*Harvard University*" using CRX are all universities, while the CCX model top-5 concepts include, besides educational institutions, location ("*Cambridge, Massachusetts*") and an affiliated group ("*Harvard Society of Fellows*"). The same pattern can be noticed with the "*X-Men*" movie where we get similar genre movies with CRX. While we get related characters such as "*William Stryker*"[14] with CCX.

Based on these observations, we claim that the CCX model would be beneficial in situations where *diversity* is more desired than *topical coherence*. This claim is also supported by the results we obtained on the concept categorization and dataless densification tasks. On concept categorization, the performance gap between CRX and CCX was large with almost all datasets. On dataless classification, the performance gap was large with documents belonging topics with nuance differences (i.e., *Autos* versus *Motorcycles*), but with other classes which have clear distinctions, the CCX performance was very competitive to CRX (e.g., *Hockey* versus *Baseball*).

In this paper, we also proposed an *efficient* and *effective* mechanism for BoC densification which outperformed the previously proposed densification schemes on dataless document classification. Unlike these previous densification mechanisms, our method *scales linearly*

[14] https://en.wikipedia.org/wiki/William_Stryker.

with the number of the BoC dimensions. In addition, we demonstrated through the results how this efficient mechanism allows generating high-quality dense BoC from few concepts alleviating the need of obtaining hundreds of concepts when generating the BoC in the first place.

Our learning method does not require training on a hierarchical concept category graph and is not tightly coupled to linked knowledge base. Rather, we learn concept representations using mentions in free-text corpora with annotated concept mentions which even if not available could be obtained through state-of-the-art entity linking systems.

Finally, the work presented in this paper serves two of our objectives: (1) it demonstrates utilizing textual knowledge bases to learn robust concept embeddings and hence increasing the *effectiveness* of the BoC representation to better capture semantic similarities between textual structures, and (2) it demonstrates utilizing the learned distributed concept vectors to increase the *efficiency* of the semantic representations in terms of space and computational complexities.

Acknowledgements This work was supported by the National Science Foundation (Grant No. 1624035). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. Baroni M, Lenci A (2010) Distributional memory: a general framework for corpus-based semantics. Comput Linguist 36(4):673–721
2. Bordes A, Usunier N, Garcia-Duran A, Weston J, Yakhnenko O (2013) Translating embeddings for modeling multi-relational data. In: Advances in neural information processing systems, pp 2787–2795
3. Ceccarelli D, Lucchese C, Orlando S, Perego R, Trani S (2013) Learning relatedness measures for entity linking. In: Proceedings of the 22nd ACM international conference on information & knowledge management. ACM, pp 139–148
4. Chang M-W, Ratinov L-A, Roth D, Srikumar V (2008) Importance of semantic representation: dataless classification. AAAI 2:830–835
5. Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P (2011) Natural language processing (almost) from scratch. J Mach Learn Res 12(Aug):2493–2537
6. Gabrilovich E, Markovitch S (2007) Computing semantic relatedness using wikipedia-based explicit semantic analysis. IJCAI 7:1606–1611
7. Harris ZS (1954) Distributional structure. Word 10(2–3):146–162
8. Hoffart J, Seufert S, Nguyen DB, Theobald M, Weikum G (2012) Kore: keyphrase overlap relatedness for entity disambiguation. In: Proceedings of the 21st ACM international conference on Information and knowledge management. ACM, pp 545–554
9. Hu Z, Huang P, Deng Y, Gao Y, Xing EP (2015) Entity hierarchy embedding. In: Proceedings of the 53rd annual meeting of the association for computational linguistics
10. Hua W, Wang Z, Wang H, Zheng K, Zhou X (2015) Short text understanding through lexical-semantic analysis. In: 2015 IEEE 31st international conference on data engineering (ICDE). IEEE, pp 495–506
11. Huang H, Heck L, Ji H (2015) Leveraging deep neural networks and knowledge graphs for entity disambiguation. arXiv:1504.07678
12. Hulpus I, Prangnawarat N, Hayes C (2015) Path-based semantic relatedness on linked data and its use to word and entity disambiguation. In: International semantic web conference. Springer, pp 442–457
13. Kim D, Wang H, Oh AH (2013) Context-dependent conceptualization. In: IJCAI, pp 2654–2661
14. Lang K (1995) Newsweeder: learning to filter netnews. In: Proceedings of the 12th international conference on machine learning, pp 331–339
15. Li P, Wang H, Zhu KQ, Wang Z, Wu X (2013) Computing term similarity by large probabilistic isa knowledge. In: Proceedings of the 22nd ACM international conference on Conference on information & knowledge management. ACM, pp 1401–1410
16. Li Y, Zheng R, Tian T, Hu Z, Iyer R, Sycara K (2016) Joint embedding of hierarchical categories and entities for concept categorization and dataless classification. arXiv:1607.07956

17. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. arXiv:1301.3781
18. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp 3111–3119
19. Milne D, Witten IH (2008) Learning to link with wikipedia. In: Proceedings of the 17th ACM conference on information and knowledge management. ACM, pp 509–518
20. Papadimitriou CH, Steiglitz K (1982) Combinatorial optimization: algorithms and complexity. Courier Corporation, North Chelmsford
21. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V et al (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12(Oct):2825–2830
22. Peng H, Song Y, Roth D (2016) Event detection and co-reference with minimal supervision. In: Proceedings of the 2016 conference on empirical methods in natural language processing, pp 392–402
23. Pennington J, Socher R, Manning CD (2014) Glove: global vectors for word representation. In: Proceedings of the empirical methods in natural language processing (EMNLP 2014), vol 12, pp 1532–1543
24. Rocchio JJ (1971) Relevance feedback in information retrieval. In: Salton G (ed) SMART retrieval system-experiments in automatic document processing. Prentice Hall, Englewood Cliffs, NJ, pp 313–323
25. Schuhmacher M, Ponzetto SP (2014) Knowledge-based graph document modeling. In: Proceedings of the 7th ACM international conference on web search and data mining. ACM, pp 543–552
26. Shalaby W, Zadrozny W (2015) Measuring semantic relatedness using mined semantic analysis. arXiv:1512.03465
27. Song Y, Roth D (2014) On dataless hierarchical text classification. In: AAAI, pp 1579–1585
28. Song Y, Roth D (2015) Unsupervised sparse vector densification for short text similarity. In: Proceedings of NAACL
29. Song Y, Wang H, Wang Z, Li H, Chen W (2011) Short text conceptualization using a probabilistic knowledgebase. In: Proceedings of the twenty-second international joint conference on artificial intelligence-volume volume three. AAAI Press, pp 2330–2336
30. Song Y, Wang S, Wang H (2015) Open domain short text conceptualization: a generative + descriptive modeling approach. In: IJCAI, pp 3820–3826
31. Wang Z, Wang H (2016) Understanding short texts. In: The association for computational linguistics (ACL) (Tutorial). https://www.microsoft.com/en-us/research/publication/understanding-short-texts/
32. Wang Z, Wang H, Hu Z (2014) Head, modifier, and constraint detection in short texts. In: 2014 IEEE 30th international conference on data engineering (ICDE). IEEE, pp 280–291
33. Wang Z, Zhao K, Wang H, Meng X, Wen J-R (2015) Query understanding through knowledge-based conceptualization. In: IJCAI, pp 3264–3270
34. Witten I, Milne D (2008) An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In: Proceeding of AAAI workshop on wikipedia and artificial intelligence: an evolving synergy, AAAI Press, Chicago, USA, pp 25–30
35. Wu W, Li H, Wang H, Zhu KQ (2012) Probase: A probabilistic taxonomy for text understanding. In: Proceedings of the 2012 ACM SIGMOD international conference on management of data. ACM, pp 481–492
36. Yamada I, Shindo H, Takeda H, Takefuji Y (2016) Joint learning of the embedding of words and entities for named entity disambiguation. arXiv:1601.01343
37. Kokoska S, Zwillinger D (1999) CRC standard probability and statistics tables and formulae. CRC Press

Walid Shalaby is currently a Research Scientist at the Industrial AI Lab, Hitachi America R&D. He received his PhD in Computer Science from the University of North Carolina at Charlotte in 2018. Dr. Walid's research interests include data and text mining, knowledge extraction, information retrieval, and machine learning. His PhD research involved developing methods for concept-based semantic representations and understanding of natural languages addressing tasks such as semantic search, entity type recognition, neural-based representations of concepts and entities. His research also involved enhancing the performance of deep learning architectures through unsupervised pre-training and entity awareness. Dr. Walid developed Mined Semantic Analysis (MSA), a patented technique for concept-based text representation. Dr. Walid has several years of research and industry experience; through three internships at CareerBuilder and Samsung Research America, he worked on designing algorithms, developing prototypes, and modeling of data driven solutions for systems such as personal digital assistants, job recommendation engines, and large scale job search engines. Dr. Walid received his Bachelor's and Master's degrees in Computer Science from the Faculty of Computers and Information, Cairo University.

Wlodek Zadrozny joined the faculty of the University of North Carolina in Charlotte in 2013, after a 27 year career at IBM T.J. Watson Research Center. Dr. Zadrozny's research focuses on natural language understanding and its applications. From 2008 to 2013, he was part of the IBM Watson project–the Jeopardy! playing machine. As a scientist at IBM Research, he led and contributed to a wide range of projects. They included semantic search applications, natural language dialogue systems, and a value net analysis of intangible assets. Dr. Zadrozny published over seventy refereed papers on various aspects of text processing; he is an author of over 50 patents granted and several patents pending. Wlodek Zadrozny received a PhD in Mathematics (with distinction) from Polish Academy of Science in 1980.