


RESEARCH

Open Access



# Task-agnostic representation learning of multimodal twitter data for downstream applications

Ryan Rivas<sup>\*</sup> , Sudipta Paul, Vagelis Hristidis, Evangelos E. Papalexakis and Amit K. Roy-Chowdhury

\*Correspondence:  
rriva002@ucr.edu  
Department of Computer  
Science and Engineering,  
University of California,  
Riverside, Riverside, CA, USA

## Abstract

Twitter is a frequent target for machine learning research and applications. Many problems, such as sentiment analysis, image tagging, and location prediction have been studied on Twitter data. Much of the prior work that addresses these problems within the context of Twitter focuses on a subset of the types of data available, e.g. only text, or text and image. However, a tweet can have several additional components, such as the location and the author, that can also provide useful information for machine learning tasks. In this work, we explore the problem of jointly modeling several tweet components in a common embedding space via task-agnostic representation learning, which can then be used to tackle various machine learning applications. To address this problem, we propose a deep neural network framework that combines text, image, and graph representations to learn joint embeddings for 5 tweet components: body, hashtags, images, user, and location. In our experiments, we use a large dataset of tweets to learn a joint embedding model and use it in multiple tasks to evaluate its performance vs. state-of-the-art baselines specific to each task. Our results show that our proposed generic method has similar or superior performance to specialized application-specific approaches, including accuracy of 52.43% vs. 48.88% for location prediction and recall of up to 15.93% vs. 12.12% for hashtag recommendation.

**Keywords:** Joint embedding, Machine learning, Twitter, Deep learning, Multimodal data

## Introduction

Twitter produces a wealth of information for analysis of trends, opinions, and interactions with 500 million tweets per day generated by its users [1]. As such, the microblogging service is a popular target for research involving machine learning. Several problem settings in the field of machine learning, or variants thereof, can focus on Twitter data. For example, sentiment analysis, spam detection, and location prediction, all well-established problem settings in their own right, are all applicable to tweets [2–4]. Much of the work in applying machine learning to Twitter data focuses on only one component, or a few components, of a tweet. A sentiment analysis model [5] might only use the text of a tweet, while a hashtag recommendation system [6] might use both text and image. A

tweet can contain additional components that may be informative to a machine learning model. Examples of these components include the author of the tweet, whose interactions with other users can be modeled by a graph [7], and the location, which can link the tweet to others at the same location [8]. Incorporating several of these tweet components into a machine learning framework can potentially create a model that is better informed than others at a given task. One approach to accomplishing this is by creating a joint embedding framework.

Joint embeddings [9] are used in several machine learning tasks that handle different modalities, e.g. text and images, in order to leverage the relationship between them. The intuition behind a joint embedding space is that inputs from different modalities mapped into the space should be close if they are semantically related. For example, in the problem of image-text retrieval [10], the image captions or tags closest to an image in an embedding space should be those that best describe the visual content of the image. However, these models are generally limited to 2 or 3 modalities (components), like the aforementioned text and image. Introducing additional modalities has two potential benefits. First, it can better inform existing applications by taking these additional modalities into account. In the image-text retrieval task, also considering the author and the location of a post from an image sharing or microblogging service might achieve better results. Second, introducing additional modalities can open up a joint embedding space to new applications. Recent work in hashtag recommendation uses the text and image of a social media post as input to a neural network model [11–13], but a joint embedding model that includes hashtags as well as text, images, and other modalities might also perform well at this task.

Our motivation for this paper is thus to address the problem of creating a joint embedding framework that incorporates several tweet components and using an embedding model trained with such a framework to address multiple machine learning problems involving Twitter data. Specifically, the first question we ask in this paper within the context of Twitter is *can additional modalities in a joint embedding space improve its performance in typical applications and/or enable it to perform well in new ones?* We show in our experiments that additional modalities can indeed allow a joint embedding model to perform better than a similar model with fewer modalities, and can also perform well at new tasks. The second question is *can we build a single, task-agnostic joint embedding model for tweets and use it several diverse applications?* Our results show that a single trained joint embedding model can be successfully applied to multiple different tasks.

### Overview of the proposed approach

Our approach builds on VSE++, the framework proposed by Faghri et al. [14]. We do so by extending the framework to incorporate 3 more tweet components in addition to text and images: hashtags, considered separately from tweet text; the author, as represented by a graph embedding [15] learned from a graph of Twitter user mentions; and location, to represent the context of a tweet in terms of what other Twitter users are discussing at the same place. Extending VSE++ is not trivial, as adding 3 additional modalities complicates training. The loss function involved in training the model must account for how one modality interacts with four others rather than just one. We accomplish this with a novel approach that learns representations for each component using triplet loss

[16] calculated using an embedding for one type of tweet component and the average embedding across all components of a tweet. Like VSE++, this loss incorporates hard negatives, which have been shown to be effective in several tasks [17–20].

Our proposed model is applicable to several tasks, including:

- Image retrieval/text retrieval: Given a tweet  $t$  without an image (or text), retrieve text (or an image) relevant to  $t$ .
- Hashtag recommendation: Given a tweet  $t$  without hashtags, predict one or more hashtags relevant to  $t$ .
- Bot detection: Given a user  $u$  and several tweets written by  $u$ , predict whether or not  $u$  is a bot, i.e. an automated Twitter account.
- Location prediction: Given a tweet  $t$  without geolocation data, predict where the author of  $t$  was when it was posted.

In our experiments, we show the performance of our proposed model compared to baselines from these domains using Twitter data.

Contributions: We make the following contributions.

- We introduce the problem of developing a task-agnostic representation learning framework for tweets that incorporates several tweet components. To our knowledge, this is the first work to address this problem.
- We develop a novel framework with pairwise ranking loss to learn a robust joint embedding with 5 tweet components.
- We showcase the usefulness of additional tweet components by applying the learned embeddings to different tasks and comparing their performance to task-specific baselines.

The remainder of the paper is organized as follows: “[Related work](#)” discusses prior work. “[Approach](#)” describes our proposed method. “[Experiments](#)” explains the experimental evaluation of our proposed method and presents the results of our experiments. We conclude in “[Conclusion](#)”.

## **Related work**

### **Joint embedding**

Joint embedding models have been proposed for image-text retrieval [14, 21–23], video-sentence retrieval [24–28], video-paragraph retrieval [29, 30], temporal localization of moments [31–33], and a variety of other tasks [34–37]. The general idea behind a joint embedding model is to place vector representations of different media, such as text and images, into the same embedding space such that the distance between semantically similar vectors (e.g. an image and its captions or tags) is minimized. For the image-text retrieval task, Faghri et al. [14] projected images and text in a visual-semantic embedding space learned with a loss function that utilizes hard negatives. In [21], Mithun et al. used images and noisy text from the Web to improve a joint embedding model. Lee et al. [23] captured the fine-grained interplay between objects present in an image and text to better align images and text in a joint embedding space. For the task of video-sentence

retrieval, Mithun et al. [24] employed multimodal cues such as image, motion, and audio for video encoding. In [25], Dong et al. used multi-level encodings for video and text to perform zero-example video retrieval. Wray et al. [26] enriched embedding learning by disentangling the parts-of-speech of captions. For the temporal localization task, moment-sentence pairs [31, 33] or clip-sentence pairs [32] are aligned in the joint embedding space.

### **Machine learning on Twitter data**

Several studies have created machine learning methods specifically for use with Twitter data [38–40]. However, these works typically focus on text instead of also including other parts of a tweet. Other work incorporates additional tweet data, e.g. images and tweet author metadata, to accomplish specific tasks such as hashtag recommendation, bot detection, and location prediction. We explore some of these studies below.

### **Hashtag recommendation**

Many studies have focused on hashtag recommendation for Twitter and other micro-blogging platforms. Rawat and Kankanhalli [11] proposed a deep neural framework to recommend descriptive tags for an image that combined image features from a convolutional neural network (CNN) [41] with features from a “ContextNet” neural network using the image’s associated location and time data as input to provide context for the image. Their proposed method outperformed a model that considered only image data. Zhang et al. [12] used an image’s features from a VGG<sup>1</sup> network [42] and text features from a long short-term memory (LSTM) network [43] representing the image’s caption as input to a co-attention mechanism [44] to recommend hashtags. They compared their proposed framework to state-of-the-art methods that used only the image caption and found that their method performed the best. The method proposed by Ma et al. [13] uses a framework similar to [12] to recommend hashtags, but it also incorporates images and text associated with previous uses of a candidate hashtag to achieve better performance than state-of-the-art methods.

### **Bot detection**

Bot detection on Twitter is another area of active research. Botometer (formerly Bot-OrNot), originally proposed by Davis et al. [45], extracts over 1000 features from a Twitter user’s metadata, interaction patterns, and content, then uses them as input to a random forest classifier [46] to predict the likelihood of that user being a bot. Botometer’s most recent version, v4 [47], is an ensemble classifier that combines Botometer v3 [48] with random forest classifiers that are each trained on a specific class of Twitter bot. This method outperformed baselines on several datasets. Another approach by Kudugunta and Ferrara [49] uses a deep learning method to determine whether an individual tweet was made by a bot, rather than the more conventional approach of determining whether or not a Twitter user is a bot. It uses word

---

<sup>1</sup> “VGG” refers to the Visual Geometry Group at the University of Oxford’s Department of Engineering Science, which created the VGG model.

embeddings from tweet text as input to an LSTM combined with tweet metadata to make its predictions and achieved high accuracy in testing compared to baselines.

### Location prediction

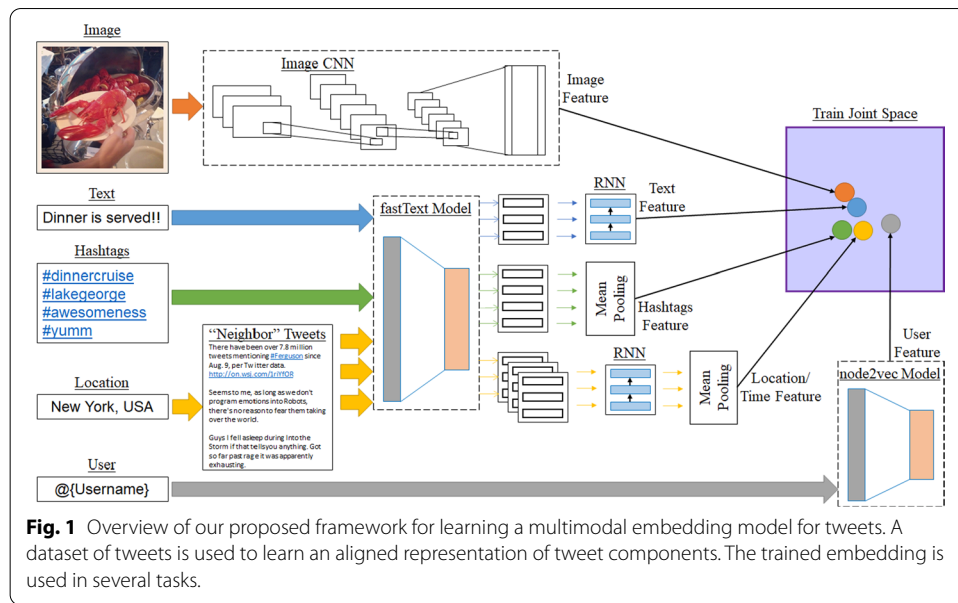
Previous work has examined location prediction for tweets. Matsuo et al. [50] combined a text-based location estimator and a CNN for image features to perform grid-based location prediction and showed that combining image and text outperformed using only one modality. Kumar and Nezhurina [51] proposed a method to predict the location of a Twitter user's *next* tweet based on the user's past tweets. Their method used tweet geo-coordinates, mentions of predefined location categories, and tweet personality traits as input to an ensemble of popular classification methods, which outperformed individual classifiers used in the ensemble. The method proposed by Lau et al., Deepgeo [52], takes a tweet's text and creation time as well as the author's UTC offset, time zone, location (i.e. the free text location listed in a user's profile), and account creation time as input to a deep learning framework to predict location as a city class, i.e. the city in which a tweet was written. The authors of [52] found that Deepgeo outperformed the state-of-the-art in this task.

### Big Data analysis on Twitter

As one of the largest social media platforms, Twitter presents a prime target for studies involving analysis of big data. Linell et al. [53] estimated the prevalence of sleep loss incurred by the beginning of Daylight Saving Time (DST) by analyzing a dataset of 13.1 million tweets. They found that the beginning of DST causes changes in sleep behavior as indicated by a change in the time of peak Twitter activity. Feizollah et al. [54] analyzed tweets related to halal tourism by identifying related topics and performing sentiment analysis. They found that the word "halal" was primarily associated with food and hotels on Twitter, many non-Muslim countries are popular halal tourist destinations, and the majority of tweets expressed positive sentiment. Piña-García and Ramírez-Ramírez [55] examined data from Twitter and other sources to predict the most frequent crimes in Mexico City. They showed that their methods can successfully estimate the occurrences of crime, they note that more work is necessary to develop an accurate model. Our proposed method can support future big data studies similar to these, as researchers can use it to analyze Twitter data they have collected according to their goals.

### Approach

In this section, we describe our proposed model to represent tweet components in an embedding space. We first provide an overview of the VSE++ framework on which our proposed method is based ("[Description of VSE++ framework](#)"). Next, we describe the structure of the neural network framework and how we represent tweet components ("[Network structure and input features](#)"). Then, we present our approach to training a joint embedding model with this framework using pairwise ranking loss ("[Training joint embedding](#)").



### Description of VSE++ framework

VSE++, proposed by Faghri et al. [14], is a joint embedding framework designed for image-text retrieval. Given a dataset of images and captions, a trained VSE++ model attempts to pair each image with its corresponding caption. It does so by projecting them in a joint embedding space; image embeddings are generated via a ResNet [56] or VGG [42] model, while the embeddings for image captions are generated by a gated recurrent unit (GRU)-based text encoder, which is a commonly-used method to represent sentences [21].

The most notable contribution of [14] is the incorporation of hard negatives during training. In the context of image-text retrieval, a hard negative is the closest non-matching caption to an image (or vice-versa). More generally, a hard negative is the closest negative (i.e. non-matching embedding) to a training query. A loss function incorporating hard negatives will assign higher loss to a query with a negative that is particularly close to the query compared to a more conventional “sum of hinges” loss function that would be more influenced by the average distance of the negatives. Faghri et al. explain that one advantage of this approach is avoiding local minima created by the sum of hinges loss, leading to a better performing model. Their loss function for an image  $i$  and an image caption  $c$  is thus

$$\mathcal{L}_{ic} = \max[0, \Delta - f(i, c) + f(i, \hat{c})] + \max[0, \Delta - f(i, c) + f(\hat{i}, c)], \tag{1}$$

where  $\hat{c}$  is the hardest negative caption with respect to  $i$ ,  $\hat{i}$  is the hardest negative image with respect to  $c$ , and  $\Delta$  is the margin parameter. Experimental evaluation with the Microsoft COCO [57] and Flickr30K [58] datasets showed that this approach was superior to other image-text retrieval methods.

VSE++ is written in Python and uses the PyTorch deep learning library [59]. Using this as our basis, we extend the concepts presented by Faghri et al. to account for additional modalities present in tweets.

## **Network structure and input features**

### ***Network architecture***

We learn a joint embedding model using a deep neural network framework. Our framework, shown in Fig. 1, has 5 branches for tweet text, an image, hashtags, the location of the tweet, and the user who wrote the tweet. Each of these branches uses a different network. The goal of this design is for the individual branch networks to focus on component-specific features while the fully connected layers convert these to embeddings in the joint space with a dimensionality of 1024.

### ***Text representation***

For encoding tweet text (i.e. the text of the tweet without hashtags), we use an embedding layer with weights initialized with word embeddings from a fastText [60] model trained on Twitter data. The dimensionality of this layer is 300. The word embeddings are then input to a GRU. The GRU maps the text features to the joint embedding space.

### ***Image representation***

To encode an image contained in a tweet, we use a 152-layer ResNet model [56] trained on the ImageNet dataset [61]. The dimensionality of the image embedding is 2048; this is mapped to the joint space via a fully connected layer. Similar frameworks have also evaluated a 19-layer VGG model [42] as an alternative, however ResNet has been shown to perform better, at least for the task of image-text retrieval [14, 21], so we limit our experiments to the ResNet model.

### ***Hashtag representation***

To represent hashtags, which are a form of metadata used to denote keywords or topics within tweet text [13], we first separate hashtags from the text of the tweet. We then average over the fastText word embeddings of the extracted hashtags and map this to the joint space with a fully connected layer.

### ***Location representation***

To emphasize the context of a tweet, we consider location in terms of tweets from the same place. This is a two-step process: first, we collect the text of “neighbor” tweets from the same place as an input tweet. For simplicity, we do this by grouping tweets from our collected Twitter data with the same Twitter-assigned place ID as the input tweet. The texts of these tweets are then encoded through a network branch identical to that of the input tweet’s text, but this is followed by averaging the encodings of the texts.

### ***User representation***

We represent the author of a tweet by using a trained graph embedding model. Specifically, we use the fastnode2vec [62] implementation of node2vec [63]. The weighted graph used to train this model was constructed with users as vertices and mentions

as edges, i.e. an edge  $(u, v, w)$  represents user  $u$  mentioning user  $v$  in  $w$  tweets. The dimensionality of the graph embedding is 300; this is mapped to the joint space via a fully connected layer.

**Training joint embedding**

For a pair of dissimilar tweets  $t_1$  and  $t_2$ ,  $t_1$  should have embedding vectors for its components that are similar to each other, but are not similar to the embedding vectors of the components of  $t_2$ . Conversely,  $t_2$  should have embedding vectors for its components that are similar to each other, but are not similar to the embedding vectors of the components of  $t_1$ . With that intuition in mind, our goal is to learn a joint embedding characterized by the weights of the fully connected layers, the text and location word embedding layers, and the GRUs.

We base our approach on previous work that uses hinge-based bi-directional ranking loss for visual-semantic embeddings [14, 21]. These approaches maximize the similarity between corresponding image and text embeddings and minimize similarity to non-matching embeddings. They also focus on hard negatives, i.e. given a pair  $(i, t)$  of image and text embedding vectors, the corresponding hard negatives are the image vector  $\hat{i} \neq i$  and the text vector  $\hat{t} \neq t$  closest to  $t$  and  $i$ , respectively.

Our approach must also account for hashtags, user, and location. To accomplish this, we first calculate the loss using each pair  $(c, a)$  in a minibatch, where  $c$  is the embedding for one component from a tweet (e.g. text or image) and  $a$  is the averaged tweet component embeddings from the same tweet. This can be written as follows:

$$\mathcal{L}_{ca} = \sum_{(c,a)} \{ \max[0, \Delta - f(c, a) + f(c, \hat{a})] + \max[0, \Delta - f(a, c) + f(a, \hat{c})] \}, \tag{2}$$

where  $\Delta$  is the margin value for the ranking loss,  $f(c, a) = f(a, c)$  is the similarity scoring function between a tweet component embedding  $c$  and averaged tweet component embeddings  $a$ , and  $\hat{a} = \arg \max_{a^-} f(c, a^-)$  and  $\hat{c} = \arg \max_{c^-} f(a, c^-)$  are the hardest negative samples. In our experiments we use cosine similarity for  $f(c, a)$ , but our approach does not depend specifically on this. With Equation 2 and the set of tweet component embeddings  $(t, i, h, l, u)$ , representing text, image, hashtags, location, and user of a tweet, respectively, our complete loss function is

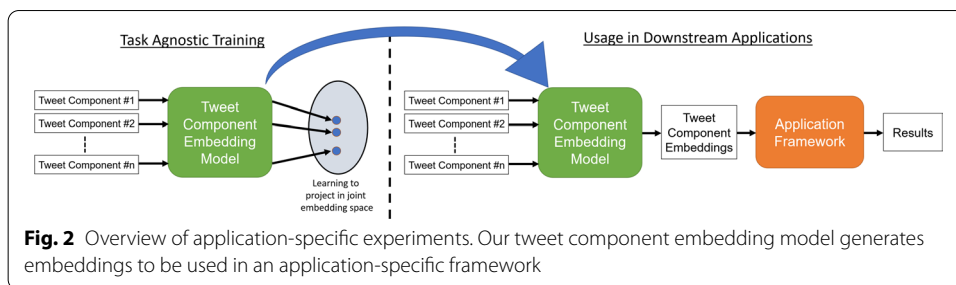
$$\mathcal{L} = \mathcal{L}_{ta} + \mathcal{L}_{ia} + \mathcal{L}_{ha} + \mathcal{L}_{la} + \mathcal{L}_{ua}. \tag{3}$$

i.e. the total loss for a minibatch is the sum of the component-specific minibatch losses (Eq. 2).

**Experiments**

Though the tweet component representations/embeddings generated by our framework are learned in a task-agnostic way, our experiments demonstrate the proposed model’s effectiveness on several machine learning applications involving Twitter data by comparing the results of experiments versus baselines designed specifically for those applications. In each of these experiments, we generate tweet component embeddings from our trained model and use them in an application-specific framework as shown in Fig. 2.





**Table 1** Applications and baselines evaluated

Section	Application	Baseline	Performance metrics
Image/text retrieval	Image/text retrieval	VSE++ [14]	Recall @ <i>k</i> , median rank
Hashtag recommendation	Hashtag recommendation	Co-Attention [12]	{Precision, recall, <i>F</i> <sub>1</sub> score} @ <i>k</i>
Bot detection	Bot detection	Botometer v4 [47]	<i>F</i> <sub>1</sub> score, AUC
Location prediction	Location prediction	Deepgeo [52]	Accuracy

Note that the tweet component embedding model is only trained once, then used in all of the applications evaluated in our experiments.

Table 1 summarizes the applications, baselines, and performance metrics in our experiments

**Dataset**

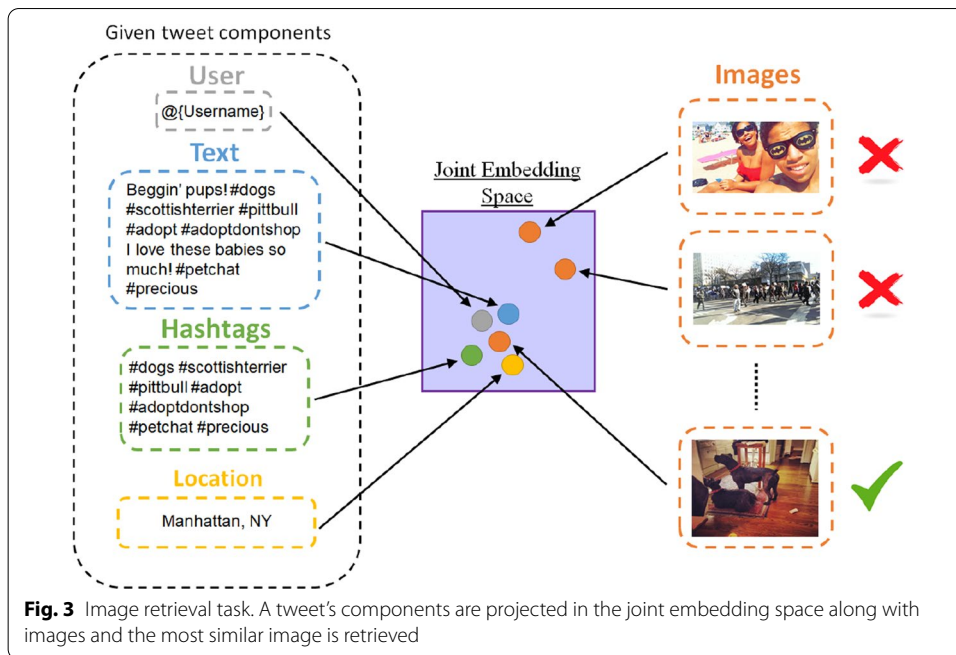
We trained our model on a dataset of tweets before applying the model to the machine learning tasks studied in our experiments. We created this dataset by collecting tweets from the Twitter Streaming API, which streams tweets in real-time.<sup>2</sup> To accomplish this, we used the Tweepy library for Python.<sup>3</sup> We collected all tweets from the API within the bounds established by the API’s rate limits. From these tweets, we filtered out tweets that do not contain all of the components necessary to generate embedding vectors with our proposed model (i.e. text, image, hashtags, geolocation data, and author ID). 100,000 tweets in the dataset from March 1–8, 2020 were used for training, while 5000 tweets from March 9, 2020 were used for validation. An additional 5000 tweets from March 10, 2020 were used for testing. The tweets used in our experiments were limited to those with a location that falls within a bounding box that encompasses most of North America.

**Training details**

The embedding networks in our model were trained with an Adam optimizer [64] over a total of 30 epochs. We set the initial learning rate of 0.0002 and decreased the learning rate by a factor of 10 after 15 epochs. We set the gradient L2 norm threshold for clipping gradients to 2, the margin  $\Delta$  to 0.2, and the mini-batch size to 128. The model was

<sup>2</sup> <https://developer.twitter.com/en/docs/tutorials/consuming-streaming-data>.

<sup>3</sup> <https://www.tweepy.org/>.



**Table 2** Image retrieval results

Method	Recall @ 1 (%)	Recall @ 5 (%)	Recall @ 10 (%)	Median rank
VSE++	0.02	0.1	0.18	2500
VSE++ with hashtags as text	0.02	0.16	0.34	2329
Proposed method	0.04	0.24	0.36	2213

evaluated on the validation set every 500 training iterations. The trained model used for evaluation on the test data was selected based on the sum of recalls (recall @ 1, 5, and 10) on the validation set to mitigate overfitting. The fastText and node2vec models used in our proposed model were trained on tweets from March 1–7, 2020, which were collected from the Twitter Streaming API as described in “Dataset”.

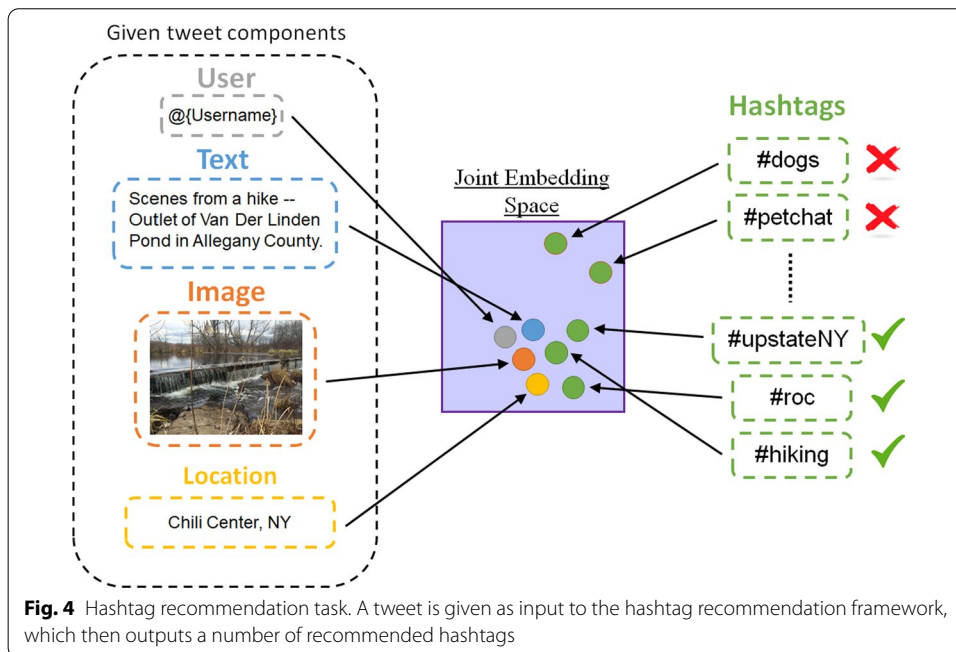
### Image/text retrieval

For image retrieval and text retrieval, we compare our proposed model to VSE++ [14], which was the basis for our method. We also include results from using hashtags as the text for VSE++ because many hashtags have significant descriptive information of their associated images [65], which the text of a tweet might not. In our image retrieval experiments, which are based on the experiments in [14], each model is given a test tweet minus the image and attempts to retrieve a relevant image from the test data. Similarly, our text retrieval experiments involve attempting to retrieve relevant tweet text given a test tweet minus its text.

While retrieval with VSE++ is simply a matter of finding the most similar image to the input text (or vice versa), this becomes somewhat more complex with the additional embeddings in our proposed model's joint embedding space. To determine which image

**Table 3** Text retrieval results

Method	Recall @ 1 (%)	Recall @ 5 (%)	Recall @ 10 (%)	Median rank
VSE++	0.02	0.1	0.18	2492
VSE++ with hashtags as text	0.02	0.08	0.16	2481
Proposed method	17.4	25.04	28.2	308



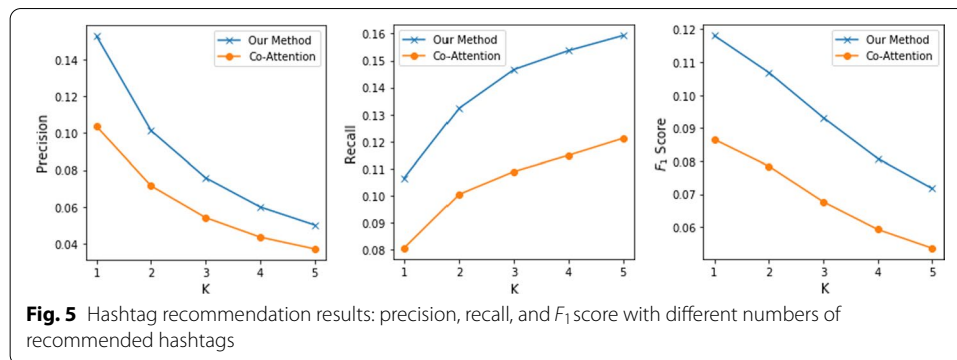
(or text) embedding to retrieve, we retrieve the embedding corresponding to the highest similarity score from any of the input tweet’s component embeddings. This is shown with our proposed model in Fig. 3, where the image corresponding to the image embedding closest to the input tweet’s component embeddings is shown.

Our results are shown in Table 2 (image retrieval) and Table 3 (text retrieval), which show that our method has higher recall (@ 1, 5, and 10) and median rank than VSE++ in both image retrieval and text retrieval of Twitter data.

**Hashtag recommendation**

Our experiments on hashtag recommendation are performed in the context of recommending hashtags for images and their associated text. Our baseline for this is the Co-Attention model proposed in [12], which uses both the image and text from a tweet or a post on a photo sharing service such as Instagram. We trained the baseline on the same dataset used to train our model described in “Dataset”.

To recommend hashtags for a test tweet  $t$  with our model, we use a nearest neighbor-style approach by calculating the embeddings of the components of  $t$  and scoring each training hashtag embedding  $h$  according to the maximum cosine similarity between  $h$  and the component embeddings of  $t$ . The recommendation of  $k$  hashtags for  $t$  is thus the



top  $k$  hashtags according to these scores. This approach is illustrated in Fig. 4 for  $k = 3$ , in which the 3 hashtags from the training data corresponding to the 3 hashtag embeddings closest to any component embedding from the input tweet are returned.

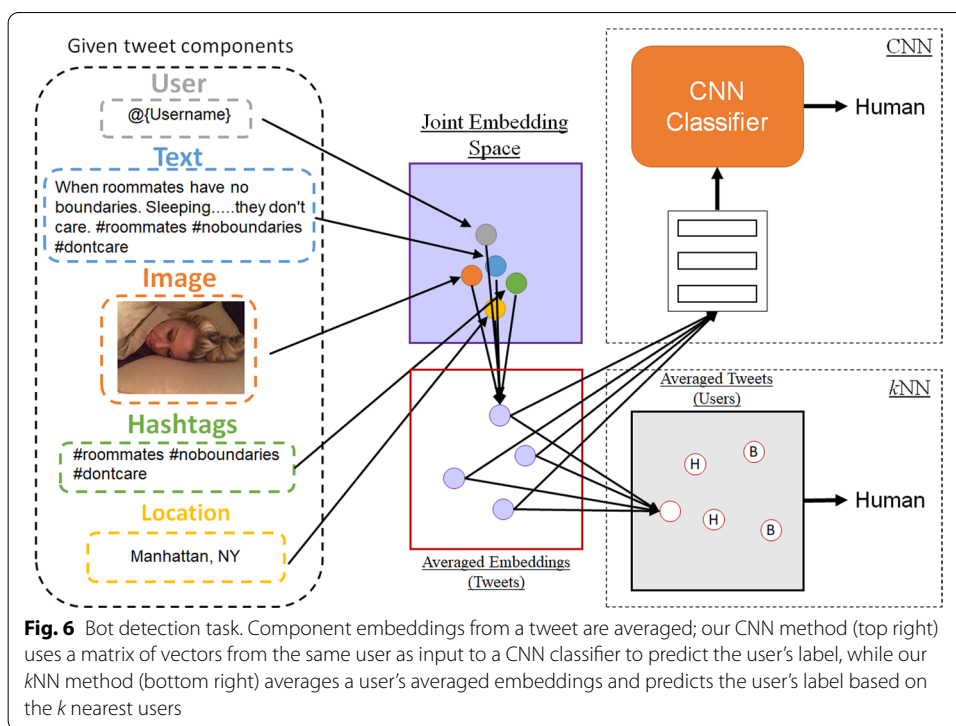
Figure 5 shows the average precision, recall, and  $F_1$  scores for  $k = 1, \dots, 5$  for both our method and the Co-Attention baseline. Our method performs better in all three measures for all values of  $k$  evaluated.

### Bot detection

As the state-of-the-art for bot detection on Twitter, Botometer v4 [47] serves as our baseline in these experiments. Because we use the same datasets in our experiments as those used in [47], we compare the results of our method to those presented there. Specifically, we compare to their cross-domain experiments, which combine several annotated bot detection datasets [47, 48, 66–73] for a training dataset of 43,576 bots and 32,849 humans and a test dataset of 9432 bots and 8862 humans. These datasets consist of a Twitter user ID combined with a binary class label indicating whether the user is a bot or a human.

We evaluated two approaches using our trained tweet component embedding model. However, the input for each approach is similar. For each user  $u$  in the bot detection datasets, we first retrieved up to 200 tweets written by  $u$ . We represent  $u$  by up to  $n$  of the most recent of these tweets, where  $n$  is a hyperparameter of our bot detection frameworks. In our experiments, we use  $n = 10$ . The component embeddings of each of these tweets are then computed by our model and averaged; each instance is thus an  $n_u \times m$  matrix composed of  $n_u \leq n$  averaged component embeddings of length  $m$  of tweets from user  $u$ .

The first approach uses tweet component embeddings in a CNN classifier. The network's structure is based on the text CNN proposed by Kim [74], except the input is matrices of averaged embedding vectors as described above rather than matrices of word embedding vectors. In addition to representing each user by up to  $n$  consecutive tweets, we also attempt to balance the training data by adding duplicates of users of the less frequent class (humans) with a different set of up to  $n$  tweets drawn from that user's retrieved set of tweets. Training the CNN is performed via 5-fold cross-validation of the training data. This approach is demonstrated in Fig. 6, which shows the conversion of a tweet's components to embedding vectors, which are then

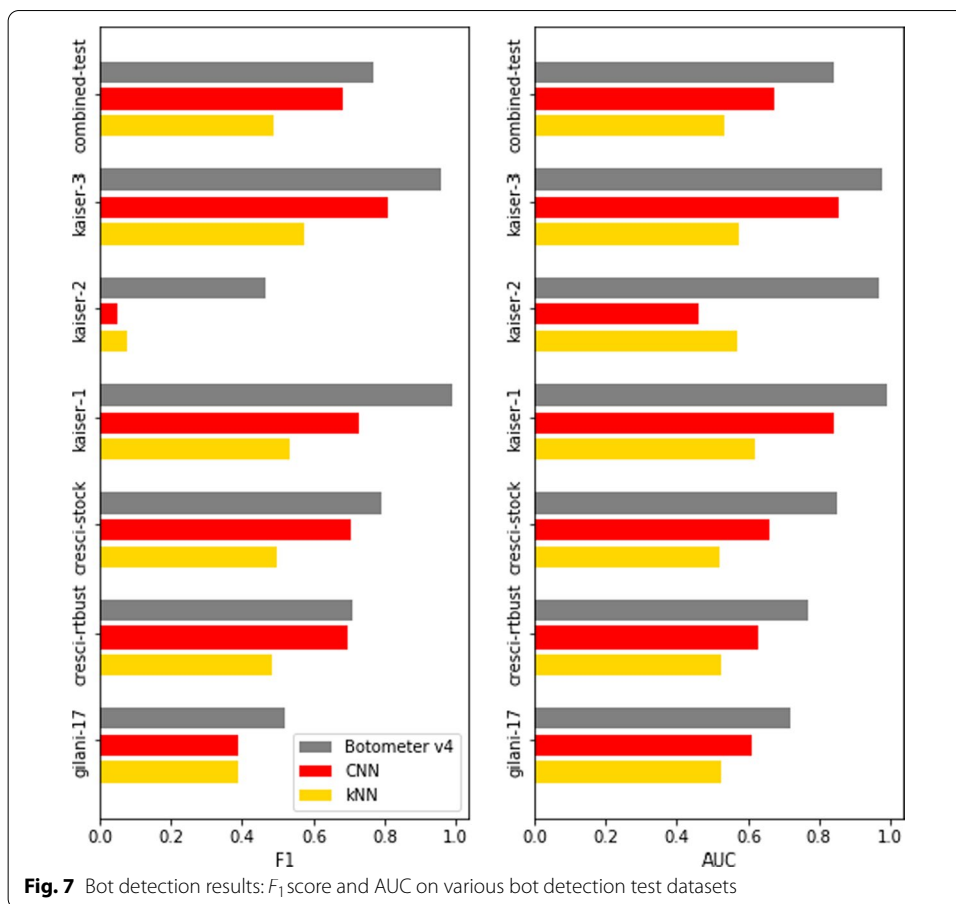


averaged. The user's averaged tweet embeddings are used as input to a CNN classifier, which predicts the user's class label.

Our second approach is a simpler  $k$ -nearest neighbors ( $k$ NN) method. For each user  $u$ , we average their embeddings into a single vector (i.e. the average of the averaged tweet component embeddings from  $u$ 's tweets) and calculate the cosine similarity between  $u$  and the users in the training data. We collect the class labels (bot or human) of the users with the  $k$  highest similarity scores; the predicted label for  $u$  is thus the most common label among the  $k$  labels collected by this method. This approach is similar to the movie recommendation system proposed by Singh et al. [75], except we are using vectors to represent Twitter users rather than movies. Figure 6 illustrates our  $k$ NN-based approach, in which tweet components are converted to embedding vectors, which are then averaged into a single vector for each of that user's tweets. Those vectors are again averaged into a single vector representing the user, whose class label is predicted according to other nearby user vectors.

We further refined our approaches by using a validation dataset taken from the training data to tune some of their hyperparameters using a small set of values for each one. Based on these results, we set the CNN's filter window sizes to 2, 3, and 4, and number of feature maps to 300. For the  $k$ NN approach, we set  $k = 1$ .

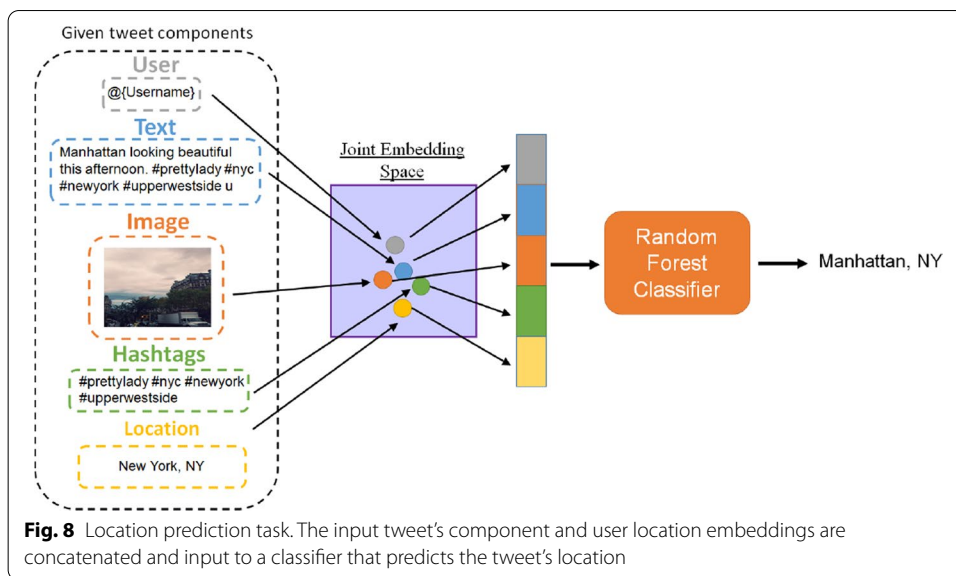
The results of our bot detection experiments are shown in Fig. 7, which shows the  $F_1$  and AUC scores for the baseline and our proposed approaches for the combined test dataset as well as individual bot detection datasets. The baseline's performance is better for all datasets, however our proposed methods come close in some cases.



**Location prediction**

We evaluate our proposed model on the task of location prediction in terms of cities, i.e. given a tweet without location data, predict the city in or near which that tweet was posted. As a baseline, we use Deepgeo [52], which combines information from a tweet as well as the author’s profile in a deep learning framework. Notably, Deepgeo is designed for the classification setting of predicting a city class rather than predicting a location’s latitude and longitude. To train and test this baseline, we used subsets of the datasets described in “Dataset” that include only tweets with a Twitter place ID corresponding to a city (i.e. we omitted tweets with a place ID corresponding to other place types such as administrative regions and points of interest) that is present in all datasets (training, validation, and test). This left a total of 41,849 tweets in the training data, 2841 tweets in the validation data, and 2594 tweets in the test data, with a set of 522 classes (cities) between them.

Our approach for using our proposed tweet embedding model uses concatenated embeddings, i.e. each tweet is represented by a single feature vector that contains each of the tweet’s non-location component embeddings. Noting that Lau et al. [52] found that a user’s location (i.e. the location listed in a user’s profile) contributed substantially to their tweet location prediction model’s accuracy, we also represent user location in the tweet’s feature vector. We do so by using the user location text of each



**Table 4** Location prediction results

Method	Accuracy (%)
Deepgeo	48.88
Proposed method	52.43

tweet’s author as input to the text branch of our trained tweet component embedding model and concatenating the resulting embedding with its corresponding tweet’s component embeddings. We then use these feature vectors and their corresponding class labels as input to a random forest classifier. We use the default hyperparameter values defined in the implementation in [76] with the exception of using “balanced subsample” class weights. This setting calculates class weights such that they are inversely proportional to class frequencies for every decision tree’s bootstrap sample. Balancing class weights in this manner accounts for any differences in class frequencies within the training data. Our tweet location prediction approach is summarized in Fig. 8, where the embeddings of a tweet’s non-location components and its author’s location text are computed and concatenated to a single feature vector, which is then passed to a random forest classifier to predict the tweet’s location.

The results of our location prediction experiments are shown in Table 4, which show that our method has higher accuracy than the Deepgeo baseline.

**Discussion**

**Image/text retrieval**

In our image and text retrieval tasks, we found that our method outperforms VSE++ in all metrics evaluated. This is particularly true for text retrieval, in which additional tweet components are more effective in retrieving text than image alone. However, we note that these findings do not extend to the more general problem of image-text retrieval, as a tweet’s text and other non-image components may not be as semantically similar to

its image as a caption written specifically for that image. This is supported by the image retrieval results, where we observe that both methods perform very poorly.

#### ***Hashtag recommendation***

We used our trained tweet component embedding model to recommend the  $k$  most similar hashtags to any of the component embeddings of an input tweet. For  $k = 1, \dots, 5$ , our method had better average precision, recall, and  $F_1$  than the Co-Attention baseline. As with our text retrieval experiments, this suggests that the addition of other tweet components along with images improves performance in this task. Further work might refine this approach, either by eliminating tweet components that do not improve performance or by adding new components not evaluated in this study.

#### ***Bot detection***

Our bot detection experiments evaluated two different methods: one that used a CNN classifier and another that used a  $k$ NN approach. In both cases, each tweet was represented by the average of all of its component embeddings. However, the Botometer v4 baseline outperformed our methods with all datasets evaluated. One limitation of our methods is the data used to train the tweet component embedding model. Our methods may perform better in the bot detection task if the model were trained with tweets from users in the bot detection training data. In the dataset described in “[Dataset](#)”, bots may be underrepresented compared to the bot detection data because bots are estimated to make up only 9–15% of active Twitter accounts [67]. Another possible reason for the poor performance of our method compared to the baseline is that the baseline benefits from some manual work in separating bots in the training datasets according to distinct bot classes; their model takes advantage of this additional information while our method only considers the binary classes of “bot” and “human.” One more possibility is that the poor performance of our proposed bot detection methods is caused by one or more of the tweet components, e.g. by overfitting. This may suggest that additional components do not necessarily contribute to model performance. If this is indeed the case, it could be overcome by validating models trained with different subsets of tweet components. Addressing these issues may improve the performance of our bot detection methods.

#### ***Location prediction***

For our location prediction task, we concatenated tweet component embeddings into a single vector for input to a random forest classifier to predict in which one of 522 cities an input tweet was written. Compared to the Deepgeo baseline, our method performed slightly better. A future iteration of our proposed method may yield higher accuracy by taking advantage of the other tweet and user information used by Deepgeo, e.g. tweet creation time, user timezone, or account creation time.

#### **Conclusions**

In this paper, we proposed a joint embedding framework for representing multimodal tweet data to generate embeddings for machine learning tasks on Twitter. The framework aligns tweet component embeddings in the joint space using a loss function that incorporates hard negatives. We tested a trained tweet component embedding model



on four Twitter machine learning applications and found that it can perform well on most applications evaluated. In the text retrieval task, our proposed method achieved recall@1 of 17.4% compared to 0.02% for the baselines. For hashtag recommendation, we achieved an  $F_1$  score of 0.1181 vs. the baseline's 0.0866 for  $K = 1$ . Our location prediction experiments showed accuracy of 52.43% for our method compared to 48.88% for the baseline. These results show that our proposed method can be applicable to a variety of tasks involving Twitter data. However, its performance in bot detection, where our method achieved an  $F_1$  score of only 69% compared to the baseline's 77% on the combined test dataset, shows that this is not the case for all tasks. Future work will investigate how our joint embedding framework can be improved, both in its design (e.g. incorporating different tweet components) and its application (e.g. how to best use generated tweet component embeddings for a given task), to perform well in additional tasks.

#### Abbreviations

CNN: Convolutional neural network; DST: Daylight Saving Time; GRU: Gated recurrent unit;  $k$ NN:  $k$ -nearest neighbors; LSTM: Long short-term memory; VGG: Visual Geometry Group.

#### Acknowledgements

Not applicable.

#### Author's contributions

RR conducted the experiments and wrote the manuscript. SP assisted with experiments and researched related work. VH conceived the study and provided coordination and guidance in the experiments and writing of the manuscript. EP and AR provided guidance in the experiments and the direction of the study. All authors read and approved the final manuscript.

#### Funding

Not applicable.

#### Availability of data and materials

The data used in our experiments are available from Twitter and [47, 48, 66–73].

#### Declarations

##### Ethics approval and consent to participate

Not applicable.

##### Consent for publication

Not applicable.

##### Competing interests

The authors declare that they have no competing interests.

Received: 2 October 2021 Accepted: 26 January 2022

Published online: 10 February 2022

#### References

1. Gruda D, Hasan S. Feeling anxious? Perceiving anxiety in tweets using machine learning. *Comput Human Behav.* 2019;98:245–55.
2. Giachanou A, Crestani F. Like it or not: a survey of twitter sentiment analysis methods. *ACM Comput Surv (CSUR).* 2016;49(2):1–41.
3. Wu T, Wen S, Xiang Y, Zhou W. Twitter spam detection: survey of new approaches and comparative study. *Comput Secur.* 2018;76:265–84.
4. Zheng X, Han J, Sun A. A survey of location prediction on twitter. *IEEE Trans Knowl Data Eng.* 2018;30(9):1652–71.
5. Pota M, Ventura M, Catelli R, Esposito M. An effective bert-based pipeline for twitter sentiment analysis: a case study in italian. *Sensors.* 2021;21(1):133.
6. Chen YC, Lai KT, Liu D, Chen MS. Tagnet: triplet-attention graph networks for hashtag recommendation. *IEEE Trans Circuits Syst Video Technol.* 2021.
7. Masood MA, Abbasi RA. Using graph embedding and machine learning to identify rebels on twitter. *J Informetr.* 2021;15(1):101121.

8. Liu Y, Luo X, Zhang M, Tao Z, Liu F. Who are there: discover twitter users and tweets for target area using mention relationship strength and local tweet ratio. *J Netw Comput Appl.* 2021;18:103302.
9. Baek D, Oh Y, Ham B. Exploiting a joint embedding space for generalized zero-shot semantic segmentation. In: *Proceedings of ICCV 21*, pp. 9536–9545; 2021.
10. Qu L, Liu M, Wu J, Gao Z, Nie L. Dynamic modality interaction modeling for image-text retrieval. In: *Proceedings of SIGIR'21*. 2021; pp. 1104–1113.
11. Rawat YS, Kankanhalli MS. ContagNet: Exploiting user context for image tag recommendation. In: *Proceedings of MM'16*. 2016; pp. 1102–1106.
12. Zhang Q, Wang J, Huang H, Huang X, Gong Y. Hashtag recommendation for multimodal microblog using co-attention network. In: *Proceedings of IJCAI'17*. 2017; pp. 3420–3426.
13. Ma R, Qiu X, Zhang Q, Hu X, Jiang YG, Huang X. Co-attention memory network for multimodal microblog's hashtag recommendation. *IEEE Trans Knowl Data Eng.* 2019.
14. Faghri F, Fleet DJ, Kiros JR, Fidler S. Vse++: improving visual-semantic embeddings with hard negatives. 2017; [arXiv:1707.05612](https://arxiv.org/abs/1707.05612).
15. Zheng W, Yin L, Chen X, Ma Z, Liu S, Yang B. Knowledge base graph embedding module design for visual question answering model. *Pattern Recogn.* 2021;120:108153.
16. Vygon R, Mikhaylovskiy N. Learning efficient representations for keyword spotting with triplet loss. 2021; [arXiv:2101.04792](https://arxiv.org/abs/2101.04792).
17. Schroff F, Kalenichenko D, Philbin J. Facenet: a unified embedding for face recognition and clustering. In: *Proceedings of CVPR'15*. 2015; pp. 815–823.
18. Wu CY, Manmatha R, Smola AJ, Krahenbuhl P. Sampling matters in deep embedding learning. In: *Proceedings of ICCV'17*. 2017; pp. 2840–2848.
19. Zheng Z, Zheng L, Garrett M, Yang Y, Xu M, Shen YD. Dual-path convolutional image-text embeddings with instance loss. *ACM Trans Multimed Comput Commun Appl (TOMM)*. 2020;30(2):1–23.
20. Zhang W, Stratos K. Understanding hard negatives in noise contrastive estimation. 2021; [arXiv:2104.06245](https://arxiv.org/abs/2104.06245).
21. Mithun NC, Panda R, Papalexakis EE, Roy-Chowdhury AK. Webly supervised joint embedding for cross-modal image-text retrieval. In: *Proceedings of MM'18*. 2018; pp. 1856–1864.
22. Wang Z, Liu X, Li H, Sheng L, Yan J, Wang X, Shao J. Camp: cross-modal adaptive message passing for text-image retrieval. In: *Proceedings of the 2019 IEEE International Conference on Computer Vision*. 2019; pp. 5764–5773.
23. Lee KH, Chen X, Hua G, Hu H, He X. Stacked cross attention for image-text matching. In: *Proceedings of ECCV'18*. 2018; pp. 201–216.
24. Mithun NC, Li J, Metzger F, Roy-Chowdhury AK. Learning joint embedding with multimodal cues for cross-modal video-text retrieval. In: *Proceedings of ICMR'18*. 2018; pp. 19–27.
25. Dong J, Li X, Xu C, Ji S, He Y, Yang G, Wang X. Dual encoding for zero-example video retrieval. In: *Proceedings of CVPR'19*. 2019; pp. 9346–9355.
26. Wray M, Larlus D, Csurka G, Damen D. Fine-grained action retrieval through multiple parts-of-speech embeddings. In: *Proceedings of ICCV'19*. 2019; pp. 450–459.
27. Liu Y, Albanie S, Nagrani A, Zisserman A. Use what you have: video retrieval using representations from collaborative experts. 2019; [arXiv:1907.13487](https://arxiv.org/abs/1907.13487).
28. Yu Y, Kim J, Kim G. A joint sequence fusion model for video question answering and retrieval. In: *Proceedings of ECCV'18*. 2018; pp. 471–487.
29. Zhang B, Hu H, Sha F. Cross-modal and hierarchical modeling of video and text. In: *Proceedings of ECCV'18*. 2018; pp. 374–390.
30. Shao D, Xiong Y, Zhao Y, Huang Q, Qiao Y, Lin D. Find and focus: retrieve and localize video events with natural language queries. In: *Proceedings of ECCV'18*. 2018; pp. 200–216.
31. Hendricks LA, Wang O, Shechtman E, Sivic J, Darrell T, Russell B. Localizing moments in video with natural language. In: *Proceedings of ICCV'17*. 2017; pp. 5803–5812.
32. Escorcía V, Soldan M, Sivic J, Ghanem B, Russell B. Temporal localization of moments in video collections with natural language. 2019; [arXiv:1907.12763](https://arxiv.org/abs/1907.12763).
33. Paul S, Mithun NC, Roy-Chowdhury AK. Text-based localization of moments in a video corpus. 2020; [arXiv:2008.08716](https://arxiv.org/abs/2008.08716).
34. Hahn M, Silva A, Rehg JM. Action2vec: a crossmodal embedding approach to action learning. 2019; [arXiv:1901.00484](https://arxiv.org/abs/1901.00484).
35. Zhu D, Ma Y, Liu Y. Deepad: a joint embedding approach for anomaly detection on attributed networks. In: *Proceedings of ICCS'20*. 2020; pp. 294–307.
36. Li C, Cao Y, Hou L, Shi J, Li J, Chua TS. Semi-supervised entity alignment via joint knowledge embedding model and cross-graph model. In: *Proceedings of EMNLP-IJCNLP'19*. 2019; pp. 2723–2732.
37. Xiong B, Bao P, Wu Y. Learning semantic and relationship joint embedding for author name disambiguation. *Neural Comput Appl.* 2020;33(6):1987–98.
38. Dhirga B, Zhou Z, Fitzpatrick D, Muehl M, Cohen WW. Tweet2vec: character-based distributed representations for social media. 2016; [arXiv:1605.03481](https://arxiv.org/abs/1605.03481).
39. Vosoughi S, Vijayaraghavan P, Roy D. Tweet2vec: learning tweet embeddings using character-level cnn- lstm encoder-decoder. In: *Proceedings of SIGIR'16*. 2016; pp. 1041–1044.
40. Müller M, Salathé M, Kummervold PE. Covid-twitter-bert: a natural language processing model to analyse COVID-19 content on twitter. 2020; [arXiv:2005.07503](https://arxiv.org/abs/2005.07503).
41. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. *Adv Neural Inform Process Syst.* 2012;25:1097–105.
42. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. 2014; [arXiv:1409.1556](https://arxiv.org/abs/1409.1556).
43. Behera RK, Jena M, Rath SK, Misra S. Co-lstm: convolutional lstm model for sentiment analysis in social big data. *Inform Process Manage.* 2021;58(1):102435.

44. Lu J, Yang J, Batra D, Parikh D. Hierarchical question-image co-attention for visual question answering. In: Proceedings of NIPS '16. 2016; pp. 289–297.
45. Davis CA, Varol O, Ferrara E, Flammini A, Menczer F. Botnot: a system to evaluate social bots. In: Proceedings of WWW '16 Companion. 2016; pp. 273–274.
46. Breiman L. Random forests. *Mach Learn*. 2001;45(1):5–32.
47. Sayyadiharikandeh M, Varol O, Yang KC, Flammini A, Menczer F. Detection of novel social bots by ensembles of specialized classifiers. In: Proceedings of CIKM '20. 2020; pp. 2725–2732.
48. Yang KC, Varol O, Davis CA, Ferrara E, Flammini A, Menczer F. Arming the public with artificial intelligence to counter social bots. *Human Behav Emerg Technol*. 2019;1(1):48–61.
49. Kudugunta S, Ferrara E. Deep neural networks for bot detection. *Inform Sci*. 2018;467:312–22.
50. Matsuo S, Shimoda W, Yanai K. Twitter photo geo-localization using both textual and visual features. In: Proceedings of BigMM '17. 2017; pp. 22–25.
51. Kumar S, Nezhurina MI. An ensemble classification approach for prediction of user's next location based on twitter data. *J Amb Intel Human Comput*. 2019;10(11):4503–13.
52. Lau JH, Chi L, Tran KN, Cohn T. End-to-end network for twitter geolocation prediction and hashing. In: Proceedings of IJCNLP '17. 2017; pp. 744–753.
53. Linnell K, Arnold M, Alshaabi T, McAndrew T, Lim J, Dodds PS, Danforth CM. The sleep loss insult of spring daylight savings in the us is observable in twitter activity. *J Big Data*. 2021;8:121.
54. Feizollah A, Mostafa MM, Sulaiman A, Zakaria Z, Firdaus A. Exploring halal tourism tweets on social media. *J Big Data*. 2021;8:72.
55. Piña-García CA, Ramírez-Ramírez L. Exploring crime patterns in Mexico city. *J Big Data*. 2019;6:65.
56. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of CVPR '16. 2016; pp. 770–778.
57. Lin T, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL. Microsoft coco: common objects in context. In: Proceedings of ECCV '14. 2014; pp. 740–755.
58. Young P, Lai A, Hodosh M, Hockenmaier J. From image descriptions to visual denotations: new similarity metrics for semantic inference over event descriptions. *Trans Assoc Comput Linguist*. 2014;2:67–78.
59. ...Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Köpf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S. Pytorch: an imperative style, high-performance deep learning library. *Adv Neural Inform Process Syst*. 2019;32:8026–37.
60. Bojanowski P, Grave E, Joulin A, Mikolov T. Enriching word vectors with subword information. *Trans Assoc Comput Linguist*. 2017;5:135–46.
61. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L. Imagenet: a large-scale hierarchical image database. In: Proceedings of CVPR '09. 2009; pp. 248–255.
62. Abraham L. fastnode2vec (2020). <https://doi.org/10.5281/zenodo.3902632> Accessed Accessed 26 Mar 2021.
63. Grover A, Leskovec J. Node2vec: scalable feature learning for networks. In: Proceedings of SIGKDD '16. 2016; pp. 855–864.
64. Kingma DP, Ba J. Adam: a method for stochastic optimization. 2014; [arXiv: 1412.6980](https://arxiv.org/abs/1412.6980).
65. Giannoulakis S, Tsapatsoulis N. Evaluating the descriptive power of instagram hashtags. *J Innov Digital Ecosyst*. 2016;3(2):114–29.
66. Lee K, Eoff B, Caverlee J. Seven months with the devils: a long-term study of content polluters on twitter. In: Proceedings of ICWSM '11 (2011).
67. Varol O, Ferrara E, Davis C, Menczer F, Flammini A. Online human–bot interactions: detection, estimation, and characterization. In: Proceedings of ICWSM '17. 2017.
68. Cresci S, Di Pietro, R., Petrocchi, M., Spognardi, A., Tesconi, M. The paradigm-shift of social spambots: evidence, theories, and tools for the arms race. In: Proceedings of WWW '17 Companion; 2017. pp. 963–972.
69. Gilani Z, Farahbakhsh R, Tyson G, Wang L, Crowcroft J. Of bots and humans (on twitter). In: Proceedings of ASONAM '17. 2017; pp. 349–354.
70. Mazza M, Cresci S, Avvenuti M, Quattrociocchi W, Tesconi M. Rtbust: exploiting temporal patterns for botnet detection on twitter. In: Proceedings of WEBSCI '19, 2019; pp. 183–192.
71. Cresci S, Lillo F, Regoli D, Tardelli S, Tesconi M. \\$.fake: evidence of spam and bot activity in stock microblogs on twitter. In: Proceedings of ICWSM '18. 2018.
72. Yang KC, Varol O, Hui PM, Menczer F. Scalable and generalizable social bot detection through data selection. In: Proceedings of AAAI '20, 2020; pp. 1096–1103.
73. Rauchfleisch A, Kaiser J. The false positive problem of automatic bot detection in social science research. *PLoS One*. 2020;15(10):0241045.
74. Kim Y. Convolutional neural networks for sentence classification. In: Proceedings of EMNLP '14. 2014; pp. 1746–1751.
75. Singh RH, Maurya S, Tripathi T, Narula T, Srivastav G. Movie recommendation system using cosine similarity and knn. *Int J Eng Adv Technol*. 2020;9:556–9.
76. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: machine learning in python. *J Mach Learn Res*. 2011;12:2825–30.

## Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.