Learning to Rank with Text-Music Embeddings

Noah Rush **Anthony Yang** **Tony Hsu** **Christopher Rybicki** **Grace Le** **Angus Lin**

Abstract

A lot of progress has been made in the field of generating annotations from images, but not much effort has been explored in generating annotations from sound, particularly music. While there are large interests in feature detection such as image and sound detection, music annotation has not been hugely explored, resulting in a lack of open-sourced higher representations of music. People tend to describe music in higher levels of representation, such as genre, or mood, more than describing solid details. As people review music in this way, they have generated a large corpus of descriptions/annotations of music in this less literal sense. In this paper, we propose an approach to learn a joint embedding space between audio features from musical performances and their reviews, using existing techniques that have been shown to work for words and images. We also provide an interface where we can upload music, compute the audio features, and return an analysis of the music based on close reviews to where it lands in the embedding space. Using the publicly available Multimodal Album Reviews Dataset (MARD), which consists of 65,566 albums and 263,525 customer reviews, we use the data to try and perform album retrieval from text reviews after embedding the audio features from albums and text reviews in a joint latent space. Using this approach we achieve a recall @ 1 of 19.95, recall @ 5 of 31.3, and recall @ 10 of 39.2. Our proposed method predicted worked better than Canonical Correlation Analysis, and was easier to scale up to new data. Unfortunately, our data was rather noisy, and the scale of audio features made it too large for our system to handle efficiently.

1 Introduction

A key area of interest in applying deep learning methods to image data is mapping the relationships between images and different kinds of text. This can take place in many forms, such as image captioning [3], poetry generation [12], or bi-directional image-sentence search [10], which frequently take advantage of multimodal datasets such as MSCoCo [1] and VizWiz-Captions [2]. One of the more curious applications of these multimodal datasets has been the use of image-text associations to produce embeddings in higher dimensional spaces that reflect the combined information associated between the two kinds of data.

In particular, we look at the papers from Liwei et. al. which investigates two-branch neural networks in creating latent representations that learn the semantic similarity of images and their corresponding textual data [14] [15]. Based on this, we seek to answer the question: how can we adapt these techniques to achieve similar results when using multimodal data consisting of audio and corresponding text descriptions?

By learning a joint embedding space between audio features and user-submitted reviews of the musical works, we are able to develop a system that allows users to submit musical recordings, and return a review predicted to be most appropriate based on similarly reviewed songs. The potential of using a joint embedding space is very open-ended, as it can be readily used for a variety of other tasks such as search and retrieval, or recommendation.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

Our open-sourced approach can be used for many applications. It can enable amateur and professional musicians to further analyze their audio recordings and receive the predicted reviews before their release. It also makes a contribution to the open-sourced music annotation and research community. Moreover, it can also allow music enthusiasts to further analyze their favorite playlists.

2 Background & Related Work

2.1 Embeddings

Word embeddings are a type of word representation that allows words with similar meanings to have a similar value within a metric space. In the world of deep learning, word embeddings are learned and trained. Word embedding belongs to a class of techniques where individual words are represented as real-valued vectors in a predefined vector space. Each word would be mapped to one specific vector, and the vector values are learned by neural network models. Each word is represented by a real-valued vector, typically with tens or hundreds of dimensions. In comparison to a classic natural language processing problem, we often use one-hot encoding, which would generate a sparse word representation, with thousands or millions of dimensions. Embeddings representation is learned from the usage of words, which allows similar words to have similar representations, while reducing the overall amount of space needed to represent it. In contrast, a bag of words model seems fragile because, unless explicitly managed, different words have different representations, regardless of how they are used.

2.2 Word Embedding Algorithms

Word embeddings can be learned either by using a neural network model with a joint task, such as document classification, or through an unsupervised process, i.e. using document statistics. An embedding layer, for lack of a better name, is a word embedding that is learned jointly with a neural network model on a specific natural language processing task, including document classification. The size of the vector space is specified as part of the model, such as 50, 100, or 300 dimensions. The vectors are initialized with small random numbers. The embedding layer is used on the front end of a neural network and is fit in a supervised way using the backpropagation algorithm.

2.3 Related Work

In Learning Deep Structure-Preserving Image-Text Embeddings [15], it was shown that you can learn an image-text embedding using a two-view neural network with two layers of nonlinearities on top of any representations of the image and text views. This paper built on work in [16] which introduced their bi-directional ranking loss to build joint embeddings. For [15], their datasets included the Flicker30K and MSCOCO image-sentence datasets. Their model for embedding is one we drew heavy inspiration from, having two branches, each with full connected layers. Using ReLu as the primary activation, they achieved best results by using batch normalization after the last linear layer, and L2 normalization before the intersection between the two branches.

3 Data

3.1 Dataset

For this project we use the Multimodal Album Reviews Dataset, or MARD, which have been enriched with music metadata from MusicBrainz, and audio descriptors from AcousticBrainz. One of the initial issues was the lack of data within in the downloaded MARD dataset. The MARD dataset only contains the audio features for one song on each album. As part of this project, we expanded the dataset by downloading additional audio features for each song within the album. We were able to expand the dataset from 28,403 files to include nearly 343,882 accousticbrainz audio feature files, with its associated albums and reviews. However, due to time and computation power limits, we were only able to utilize a subset of this dataset. For future work, we hope to expand the model to encompass the entire dataset.

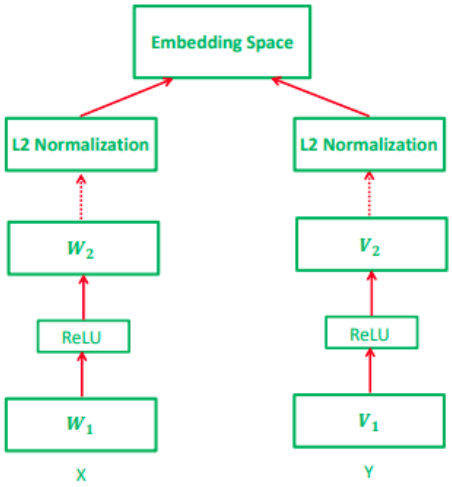

Figure 1: Proposed model structure based on [15]. In [15], X represented image features, but we will be using audio. W and V are fully connected layers separated by a non-linear activation and L2 normalization is performed before entry into the embedding space.

3.2 Multimodal Album Reviews Dataset (MARD)

The Multimodal Album Reviews Datset contains a number of text and accompanying metadata. However, most of this actually originates from a much larger Amazon dataset, which provides millions of reviews along with extraneous information such as date of publication, rating, and creator ID. Because each review is correlated to a product, every product has additional metadata associated with it, such as list of similar products, price, sell rank, genre categories and Amazon product ID. The Amazon taxonomy of genres consists of around 27 labels at the first level, namely alternative rock, reggae, classical...etc. MARD amounts to a total of 65,566 albums and 263,525 customer reviews. In MARD [6], they performed experiments on music genre classification, exploring a variety of feature types, including semantic, sentimental and acoustic features. These experiments demonstrated that modeling semantic information contributes to outperforming strong bag-of-words baselines. Their analysis implied a potential correlation between key cultural and geopolitical events and the language and evolving sentiments found in music reviews. Even though MARD has 65,666 albums, only 8,683 albums are initially mapped to accousticbrainz features. These albums have 84,747 reviews, around 9.7 reviews per album.

3.3 AcousticBrainz

AcousticBrainz is open-source community dataset that consists of a multitude of music features which have been extracted from over 4 million uniquely identified audio files (designated by MusicBrainz Identifiers or MBID). The platform is divided into three categories: data storage, feature extraction, and the implementation of musical semantic information. For this project, we are only interested in the data storage aspect, which is made readily available via an API. Using the MBID of an input audio file, we are able to obtain data about a particular file. Using the API has two types of data calls: high-level and low-level. The high-level data features include danceability, gender, acousticmood aggressive, timbre...etc. The low-level data features include dissonance, dynamic complexity, erbbands, melbands, mfcc, spectral entropy, and many others. Overall, AcousticBrainz describes the acoustic characteristics of music and includes low-level spectral information and high-level information for genres, moods, keys, scales.

4 Model

4.1 Network Architecture

We aim to follow the models and losses proposed in [14] and [15] and build a two branch neural network that takes in audio features and text embeddings and maps them to a joint latent space of common dimensionality. Wang et. al. in [14] and Microsoft in [5] also propose using joint-embedding spaces for more direct classification. The models use binary cross-entropy to try and match image to text in one, and audio and text in the other. However, since reviews can take many forms, and are more open-ended than semantic descriptors, we are not searching for a matching, but are more interested in relationships between texts and audio features.

Figure 1 shows the core embedding network, which is really quite simple. It takes X, features from audio, and Y, features from text, and runs them through a two layer fully-connected network. The layers are separated by a ReLu activation layer and L2 normalized before placed in the embedding space. During learning, the model computes the distance between true pairs of audio features and text reviews, and tries to bring them closer together, while trying to separate false pairs of embedded text and audio. In this way the embedded text and audio features live in the same embedding space. This architecture is merely the core, there are a lot of options to generate text and audio features, including some that will be trained downstream from the loss described below.

4.2 Training & Loss

We will try and learn similarities and groupings by using a ranking loss. A ranking loss aims not to categorize, but to rather rank the relative distances between inputs. To learn a ranking, we will sample triplets of the data and run them stochastic gradient descent. First, we consider the a bi-directional ranking loss. Consider an audio input x_i, then Y_i^+ and Y_i^- are respectively the sets of matching text reviews and non-matching. (In our dataset, it is possible for one audio feature to have many reviews). For $y_j \in Y_i^+$ and $y_k \in Y_i^-$ we want

$$d(x_i, y_j) + m < d(x_i, y_k) \tag{1}$$

where d(x,y) refers to the euclidean distance between the audio and text features in the embedding space. Vice-versa, for a text sample y_i We want

$$d(x_j, y_i) + m < d(x_k, y_i) \tag{2}$$

where $x_j \in X_i^+$ and $x_k \in X_i^-$. So we want matching pairs to be closer together to each other by some margin, m. These two constraints can be combined to form a margin-based loss:

$$L(X, Y) = \lambda_1 \sum_{i,j,k} [m + d(x_i, y_j) - d(x_i, y_k)]_+ + \lambda_2 \sum_{i',j',k'} [m + d(x_j, y_i') - d(x_k, y_i')]_+ \tag{3}$$

where $[t]_+ = max(0, t)$.

If the correct example is close enough within the margin of error, the max terms will take the loss to 0, λ_1 and λ_2 are hyper-parameters to balance the loss between audio and text. Another note, triplet sampling is a little weird, the literature suggests sampling within the mini-batch, taking a pair(x,y) and sampling until finding another y', such that (x,y') are not a match.

In their 2016 and 2018 papers, Wang et. al. also propose using neighborhood constraints to preserve the structure of their embedding space. In their project, they have a many-to-many correspondence in their image and text examples. After our album dataset expansion, we also have a many-to-many correspondence in that albums can have many reviews, and each review describes an album, which can have multiple songs, and thus groupings of audio features. The neighborhood constraints preserve structure by ensuring that reviews that describe the same album are close together, and that songs on the same album are close together. The loss is enforced much like the one defined above, but defined in terms of neighborhood. Let $N(y_i)$ be the neighborhood of some text y_i such that all other$y's$ in the neighborhood review the same album. Then our margin preserving condition is :

$$d(y_i, y_j) + m < d(y_i, y_k), y_j \in N(y_i) y_k \notin N(y_i) \tag{4}$$

For a song x_i, let x_j be on the same album and x_k be on a different album, then:

$$d(x_i, x_j) + m < d(x_i, x_k), x_j \in N(x_i) y_k \notin N(x_i) \tag{5}$$

We can convert this loss to a margin loss equation like above, and scale with hyper-parameters, λ_3 and λ_4. Our final loss is:

$$L(X,Y) = \lambda_1 \sum_{i,j,k} [m + d(x_i, y_j) - d(x_i, y_k)]_+ \tag{6}$$

$$+ \lambda_2 \sum_{i',j',k'} [m + d(x_{j'}, y_{i'}) - d(x_{k'}, y_{i'})]_+ \tag{7}$$

$$+ \lambda_3 \sum_{i,j,k} [m + d(x_i, x_j) - d(x_i, x_k)]_+ \tag{8}$$

$$+ \lambda_4 \sum_{i',j',k'} [m + d(x_{i'}, y_{j'}) - d(y_{i'}, y_{k'})]_+ \tag{9}$$

$$\tag{10}$$

4.3 Text Features

For the text features, we start by using a mean word-embedding. Given a sentence, S, with a sequence of tokens $(s_1, ..s_n)$,

$$y = \frac{\sum_{i=1}^{n} \phi^w(s_i)}{n} \tag{11}$$

where ϕ^w is a learned embedding function.

Mean embeddings lose all the sequence information contained in a sentence when generating text features. So from there, we move one to using a recurrent neural network, specifically an LSTM. Without going into details on the architecture of the LSTM [8], the main gist is that it keeps a hidden state and a cell state as it processes the sentence tokens from left to right. We use a bidirectional LSTM, which goes in both directions, and concatenate the final hidden states from each direction to create our text feature vector:

$$\overrightarrow{\mathbf{h}_{j+1}} = \overrightarrow{\text{LSTM}^E}\left(\phi^w(x'_{j+1}); \overrightarrow{\mathbf{h}_j}\right) \tag{12}$$

$$\overleftarrow{\mathbf{h}_{j-1}} = \overleftarrow{\text{LSTM}^E}\left(\phi^w(x'_{j-1}); \overleftarrow{\mathbf{h}_j}\right) \tag{13}$$

$$y = [\overleftarrow{h_1}; \overrightarrow{h_n}] \tag{14}$$

Finally, instead of learning text features on the fly, we try using feature extraction from a pre-trained BERT model. In the original BERT paper, [4], they show that feature extractions can lead to good performance in downstream NLP tasks.

4.4 Audio Features

We used audio features from Mel-frequency cepstrum. The mel-frequency cepstrum (MFC) represents the short-term power spectrum of a sound, computed from a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. Mel-frequency cepstral coefficients (MFCCs) are coefficients that together construct an MFC. The coefficients derive from a type of cepstral representation of the audio clip.

MFCCs are commonly derived as follows:

- Compute the Fourier transform of a signal.
- Map the powers of the spectrum obtained onto the mel scale, using triangular overlapping windows
- Compute the logs of the powers at each of the mel frequencies.
- Compute the discrete cosine transform of the list of mel log powers
- The MFCCs are the amplitudes of the resulting spectrum.

Another audio feature we used is spectral rolloff. Spectral rolloff is the frequency below which a specified percentage of the total spectral energy, e.g. 85%, lies. We chose this feature because the songs of different genre have much different frequency below their total spectral energy.

We also used the zero-crossing rate, the rate at which a signal changes from positive to zero to negative or from negative to zero to positive. We chose this feature because the value is widely used in both music information retrieval and speech recognition, as a mean to classify percussive sounds.

5 Evaluation

To provide baseline examples to test our set against, we used Canonical Correlation Analysis (CCA) between the audio and text features (as a baseline). We will also test against the linear form of learning joint-embedding spaces(WSABIE)[16].

Inspired by [13], we attempt album retrieval using review data. We withhold 10% of the album-review matches from the data into a development set. After we train the model, we embed all the albums and reviews into the embedding space, and check to see how close the correct album is. From this, we calculate recall@1, recall@5 and recall@10.

6 Implementation

6.1 Cross Correlation Analysis

Canonical Correlation Analysis is a statistical technique that has been used as a simple but surprisingly robust baseline for mapping related data sets to shared latent spaces [9]. It has been used in several other papers for learning embeddings between multi-modal data such as [6], [7], [11]. The core idea of the algorithm is to find projections of corresponding vectors that maximize the correlation between the projections.

For this model, we provided audio features directly as one-hot vectors, and we provided reviews as sentence embeddings generated by averaging the pre-trained GLoVe word embeddings, so that the final outputs are 300-dimensional vectors. Since this technique has a runtime complexity of $O(n^2)$, the model is less feasible for larger datasets, so we only tested this using the first 10,000 albums.

6.2 Two Branch Neural Network

We trained our models for 200 epochs with a batch size of 128 on a NVIDIA RTX 3060 ti GPU. We used Adam as our optimizer with a learning rate of 0.001. Our final embedding dimension was 128, ϕ^w our text embedder had dimension 256, and the fully connected dimension layer was also 256. For our LSTM's we used a hidden state dimension of 256. We used a vocab size of 60000, all other words beyond that were tagged with an unknown token. Our margin for the margin loss was 0.05, and we sampled 10 false pairs for every true pair.

Our sampling procedure tried to follow the one outlined in [14]. From a batch size of 128 true pairs, we mixed them up k times, each time matching an audio feature set with a non-matching review, and vice-versa. We also implemented Neighborhood sampling, which differs from the neighborhood constraint defined above. In neighborhood sampling, we make sure for each audio feature in a batch, that there is another audio feature from the same album, and for each review, there is another review reviewing the same album.

7 Experiments

We tested out several configurations of our model, including using a multi-layer two-branch network separated by non-linearity, or to use one linear layer, using an LSTM on to generate text features, using an LSTM to generate audio features, using a bi-directional margin loss, using neighborhood sampling, and finally using neighborhood constraints. Table 1 shows our results. Our results come from our dev set, and are reported in terms of recall@1, 5 and 10.

8 Results

Table 1 shows the results of various experiments we did. The first thing to notice is that BERT was rather unhelpful, and it benefited a lot to have trainable embedding parameters. A non-linear model

also improved over the linear variant, but a bigger improvement came from using the bi-directional loss instead of a uni-directional loss. To run the model with a uni-directional loss, we simply set $\lambda_1 = 1$, and λ_2 =0 from equation 3. LSTM features did not tank the model, but also did not provide any substantial performance increase. The technique that we found the most helpful was implementing neighborhood sampling. This resulted in our best model with a 50% performance increase.

With our baseline (CCA), we tested recall on the same dataset that our were used for generating the projection weights, and measured the review-to-audio recall with different latent dimensionalities. With latent vectors of size 50, we achieved top-10 recall of 8.87%, top-5 recall of 5.74%, and top-1 recall of 1.77%. With latent vectors of size 100, we achieved top-10 recall of 9.74%, top-5 recall of 6.90%, and top-1 recall of 2.54%.

Ablations					Review-to-Audio		
Bert Features(text)	LSTM(text)	non-linear	bi-directional	neighbor. sampling	R@1	R@5	R@10
-	-	✓	-	-	5.68	10.97	15.36
-	-	-	✓	-	8.46	15.37	22.30
✓	-	✓	✓	-	1.99	4.22	6.4
-	-	✓	✓	-	10	16.74	22.22
-	✓	✓	✓	-	9.6	17.34	22.39
-	-	✓	✓	✓	**19.95**	**31.3**	**39.2**

Table 1: Dev Performance for Model Ablations

9 Conclusion

This project attempted to adapt techniques for multi-modal learning between image and text, to a different domain, audio and text. Unfortunately, we were working with very noisy data, given that amazon annotations can frequently not be describing the audio, but rather the consumer experience, or many other things. Additionally, our audio features were not clearly labeled with regards to time and computation. However, this paper achieves two major things. One, we were able to expand the MARD dataset to contain more audio features from every song for each album, and secondly, we implemented an embedding network to attempt review to album retrieval. The embedding network works by learning a non-linear mapping for both audio features and text reviews into a joint latent space. Using this network. With this noisy data, we were able to achieve around 20% recall@1 with review text, and 40% recall@5. While these are not incredibly accurate numbers, we believe that with cleaner data, two-branch embedding networks with neighborhood sampling offers a worthwhile option for future multi-modal tasks, even with audio. We also think there is more room for improvement with audio feature manipulation. Our system dealt with a lot of data, and the run-times were very long, so we couldn't exploit all the available audio data.

10 Extension

To further perform user testing, we developed a local server to provide an interface for users to upload music and receive instant analysis of the music based on close reviews to where it lands in the embedding space. First, the user can upload any type of audio recording by clicking on 'Choose a file' button. Then, the server generated a .JSON file using the executable from AcousticBrainz, comprised of the selected audio features (see Audio Features section above). Then, it fed this .JSON file into the model and returned the analysis of the uploaded audio recording based on the closest reviews it can find in the embedding space. Figure 2 shows an example of this. The user uploaded audio recording of "In Undertow" by Alvvays and received the output of the model, which is the closest review of the song.

11 About the Team Members

Noah Rush is a CS major at Cornell Tech interested in language, music and images. His last project (available here: https://shrouded-meadow-94622.herokuapp.com/public/pdfs/ML2020.pdf), used GANs to try and transfer portrait art between genres. **Tony Hsu** is a CS major at Cornell Tech

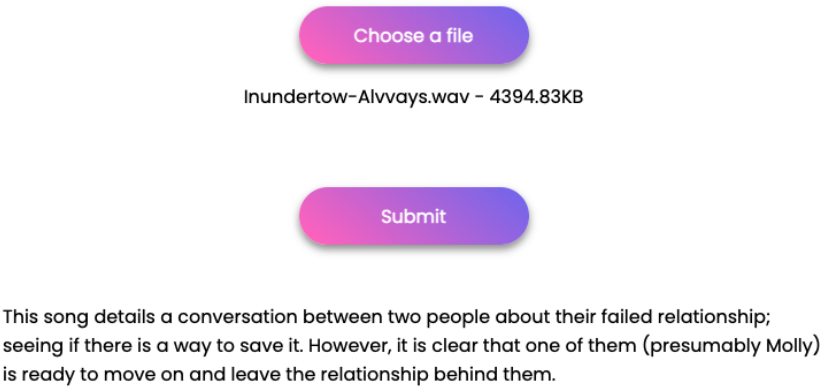

Figure 2: Interface for users to upload music, compute the audio features, and return the closest review it can find it the embedding space.

interested in using data science to empower social justice. **Angus Lin** is also a CS major at Cornell Tech, who has experiences in large-scale and distributed systems, parallel and cloud computing. His recent interest focuses on ML engineering and the scalability of ML system. **Grace Le** is a Health Tech student at Cornell Tech where her interests emerge with a curiosity about how we use mathematics and machine learning to facilitate better decision making in healthcare. **Christopher Rybicki** is a CS major at Cornell Tech interested in using algorithms and AI to reduce inequality and make open source infrastructure more sustainable.**Anthony Yang** is a CS major at Cornell tech interested in data science and algorithms. Anthony has previously worked with Noah on using GANs to transform art into different genres.

References

[1] Coco-common objects in context. `https://cocodataset.org/#home`. Accessed: 2021-05-22.

[2] Vixwiz. `https://vizwiz.org/`. Accessed: 2021-05-22.

[3] Jyoti Aneja, Aditya Deshpande, and Alexander G Schwing. Convolutional image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5561–5570, 2018.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[5] B. Elizalde, S. Zarar, and B. Raj. Cross modal audio search and retrieval with joint embeddings based on text and audio. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4095–4099, 2019.

[6] Yunchao Gong, Qifa Ke, Michael Isard, and Svetlana Lazebnik. A multi-view embedding space for modeling internet images, tags, and their semantics. *International journal of computer vision*, 106(2):210–233, 2014.

[7] Yunchao Gong, Liwei Wang, Micah Hodosh, Julia Hockenmaier, and Svetlana Lazebnik. Improving image-sentence embeddings using large weakly annotated photo collections. In *European conference on computer vision*, pages 529–545. Springer, 2014.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[9] Harold Hotelling. Relations between two sets of variates. In *Breakthroughs in statistics*, pages 162–190. Springer, 1992.

[10] Andrej Karpathy, Armand Joulin, and Li Fei-Fei. Deep fragment embeddings for bidirectional image sentence mapping. *arXiv preprint arXiv:1406.5679*, 2014.

[11] Benjamin Klein, Guy Lev, Gil Sadeh, and Lior Wolf. Fisher vectors derived from hybrid gaussian-laplacian mixture models for image annotation. *arXiv preprint arXiv:1411.7399*, 2014.

[12] Bei Liu, Jianlong Fu, Makoto P. Kato, and Masatoshi Yoshikawa. Beyond narrative description: Generating poetry from images by multi-adversarial training. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, page 783–791, New York, NY, USA, 2018. Association for Computing Machinery.

[13] Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. Evaluation methods for unsupervised word embeddings. 2015.

[14] Liwei Wang, Yin Li, Jing Huang, and Svetlana Lazebnik. Learning two-branch neural networks for image-text matching tasks, 2018.

[15] Liwei Wang, Yin Li, and Svetlana Lazebnik. Learning deep structure-preserving image-text embeddings. *CoRR*, abs/1511.06078, 2015.

[16] Jason Weston, Samy Bengio, and Nicolas Usunier. Large scale image annotation: Learning to rank with joint word-image embeddings. In *European Conference on Machine Learning*, 2010.