PinText: A Multitask Text Embedding System in Pinterest

Jinfeng Zhuang
jzhuang@pinterest.com
Pinterest
Seattle, WA

Yu Liu
yliu@pinterest.com
Pinterest
San Francisco, CA

ABSTRACT

Text embedding is a fundamental component for extracting text features in production-level data mining and machine learning systems given textual information is the most ubiqutious signals. However, practitioners often face the tradeoff between effectiveness of underlying embedding algorithms and cost of training and maintaining various embedding results in large-scale applications. In this paper, we propose a multitask text embedding solution called PinText for three major vertical surfaces including homefeed, related pins, and search in Pinterest, which consolidates existing text embedding algorithms into a single solution and produces state-of-the-art performance. Specifically, we learn word level semantic vectors by enforcing that the similarity between positive engagement pairs is larger than the similarity between a randomly sampled background pairs. Based on the learned semantic vectors, we derive embedding vector of a user, a pin, or a search query by simply averaging its word level vectors. In this common compact vector space, we are able to do unified nearest neighbor search with hashing by Hadoop jobs or dockerized images on Kubernetes cluster. Both offline evaluation and online experiments show effectiveness of this PinText system and save storage cost of multiple open-sourced embeddings significantly.

CCS CONCEPTS

• **Computing methodologies → Learning latent representations**; **Multi-task learning**;

KEYWORDS

Text Embedding, Multitask Learning, Nearest Neighbor Search;

ACM Reference Format:
Jinfeng Zhuang and Yu Liu. 2019. PinText: A Multitask Text Embedding System in Pinterest. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330671

1 INTRODUCTION

With the mission *to bring everyone the inspiration to create a life they love*, Pinterest has grown into one of the largest intelligent recommendation engines, with one of the fastest growth speed ever in industry. There are more than 250 million users using Pinterest to generate, discover, and share pins based on their personal tastes every month. The core concept of Pinterest app is a *pin*, which consists of a dominant image accompanied with rich textual metadata including title, description, comments, etc. Pinterest presents a pin to users through three surfaces: homefeed, related pin, and search. The user interface is illustrated in figure 1.

Although visual content plays an important role in the backend intelligent engine, textual information is still of upmost importance, given the fact that it can 1) represent visual content in human readable format and is almost always present together with a pin, 2) be easily extracted, derived, and stored, and 3) be put into serving system with mature inverted index solutions to support retrieval. Therefore, it is a very basic and necessary production requirement to build a text extraction pipeline and a natural language processing (NLP) component on top of it.

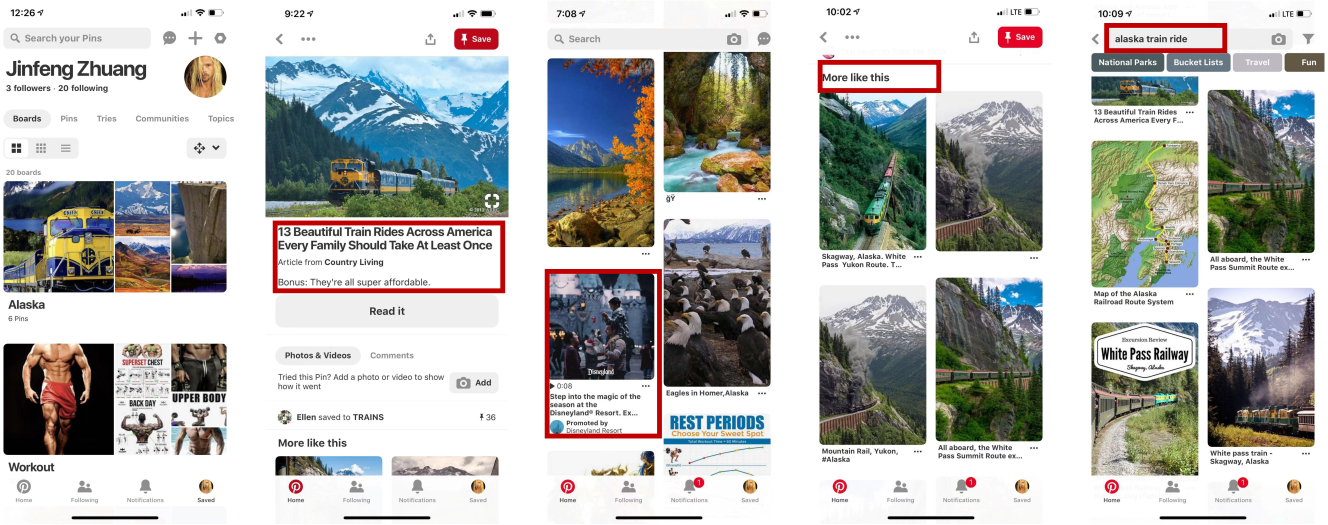
At Pinterest, we have dedicated knowledge teams that extract pins' and users' text into a set of canonical *annotation terms* from pins' title, description, board names, URLs, and visual content. It takes significant effort to generate candidate text sources and to rank them properly. In this paper, assuming proper textual information is extracted and ready to be consumed, we focus on the NLP part of the problem, in particular, the text embedding module, which abstracts input text into a real vector space, with the design goal of encoding semantics of text quantitatively, such that similar texts are close to each other by distance metrics in vector space. As a direct result, we are able to represent a pin, a user, and a search query in the same space by averaging their word level embedding vectors. This compact representation enables us to do classification and retrieval between different type of objects via nearest neighbor search in a unified way. Moreover, distance or similarity score can be used as a discriminative feature in retrieval relevance model or click prediction model.

Word embeddings have been actively studied and developed by researchers and practitioners in the machine learning community since the neural network language model was proposed [3, 4, 19, 22, 28, 33]. Researchers not only proposed principled algorithms and open sourced code, but also released pretrained embedding models with public data corpus like Wikipedia, Twitter, or Google news. It is known that high-quality word representation usually contributes most to text classification or other general tasks [29]. Moreover, our textual data is more about concrete annotation terms and short phrases than long sentences and paragraphs, which makes word embedding a more fundamental component than complex neural network architecture for specific tasks.

Although pretrained word embeddings provide a good baseline in Pinterest, we do observe the existence of a clear gap between industrial application and academic research, which motivates us

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330671


12:26
Search your Pins
Jinfeng Zhuang
3 followers · 20 following
Boards Pins Tries Communities Topics
20 boards
Alaska
6 Pins
Workout
9:22
Save
13 Beautiful Train Rides Across America Every Family Should Take At Least Once
Article from Country Living
Bonus: They're all super affordable.
Read it
Photos & Videos Comments
Tried this Pin? Add a photo or video to show how it went
Add
Ellen saved to TRAINS
More like this
7:08
Search
Step into the magic of the season at the Disneyland® Resort. Ex...
Promoted by Disneyland Resort
Eagles in Homer,Alaska
REST PERIODS
10:02
More like this
Skagway, Alaska. White Pass Yukon Route. T...
Mountain Rail, Yukon, #Alaska
All aboard, the White Pass Summit Route ex...
10:09
alaska train ride
National Parks Bucket Lists Travel Fun
13 Beautiful Train Rides Across America Every F...
Map of the Alaska Railroad Route System
All aboard, the White Pass Summit Route ex...
White Pass Railway
White pass train - Skagway, Alaska
Home Following Notifications Saved

(a) USER profile page. We can derive its interest by boards and search queries etc.

(b) A PIN about "Alaska train ride". Highlighted part is textual description of the pin.

(c) HOMEFEED page of a particular user. Highlighted part is an ad a.k.a. promoted pin.

(d) RELATED PINs of a subject pin about "Alaska train ride". They are semantically similar to it.

(e) SEARCH page presenting matched pins given a particular input query "Alaska train ride".

Figure 1: Illustration of Pinterest app around two core concepts: user (a) and pin (b). Figure (c,d,e) presents an example of homefeed, related pin, and search, respectively, regarding this particular user and the idea "alaska train ride". When user performs "repin" or "click" actions, we received a positive vote in logging system. The learning task is to mine the semantic text embeddings behind such operations. We know voting results but we don’t know who voted throughout this work.

to text embedding at practical system level. We call out the key points of design philosophy as following:

- *storage cost*: a single model easily takes up to tens of terabytes to store billions of pins with vector representation of cardinality 300. It is an even bigger cost of space to maintain multiple versions of embedding, plus the realtime computation cost like cosine similarity increases linearly to number of embeddings. So we hope to have an all-in-one solution to cover typical scenarios to save storage;
- *memory cost*: we need to compute embeddings on-the-fly in realtime application like query embedding, because there are always unseen queries that churn in everyday. This means we need to load models into memory. However, the pretrained fastText model for top 10 languages takes more than 50 gigabytes. We have to tailor models to Pinterest data to be of reasonable size at word level instead of character level;
- *supervised information*: most pretrained models are learnt in an unsupervised way by essentially predicting neighboring words in a context window. Supervised data could guide model learning more efficiently. It takes huge amount of supervised data to derive meaningful embedding vectors given the number of variables is #word × #dimension. Fortunately, we have Pinterest user engagement as a natural source of labeled data;
- *throughput and latency*: we have to iterate fast when new data comes and new experiements results are observed. At inference stage, we need infrastructure support for distributed offline computation. And we have to keep embedding models simple for latency-critical realtime computation.

Based on the motivations above, we believe it is necessary to build an in-house text embedding system to model the taste behind of each pin by mining its textual information. The storage and memory cost issue is resolved by a unified word level embedding focusing on Pinterest only data. We sample user repin and click engagement as positive training data, which is essentially supervised data labeled by all Pinners. The throughput and latency issue is resolved by a distributed cluster and proper caching servers. We have launched several projects in production depending on PinText. For the remaining part of this paper, section 2 reviews related works and highlights necessity of PinText, section 3 introduces system architecture, section 4 elaborates algorithm details, section 5 presents empirical results and applications, and section 6 concludes this work and lists future directions.

2 RELATED WORK

From the machine learning perspective, our work is closely related to three areas: text embedding in natural language processing, multitask learning, and transfer learning.

2.1 Text Embedding in NLP

Since the neural network language model was proposed [3], word embedding techniques have been actively stutied in machine learning community, with some representative examples including but not limited to word2vec [19], GloVe [22], tagSpace [32], fastText [4], starSpace [33], conceptNet [28], and more works focusing on context dependent embedding and efficient distributed learning [12, 17, 20, 30]. Most of them have pretrained models available to download and are ready to use. Those excellent works enable us to compare them directly within Pinterest applications. It has been reported previously that the word embedding itself contributes most to the success NLP models [1, 10, 29, 33]. It also provides the flexibility of building both supervised models (for classification, ranking, click prediction) and unsuperivised models (clustering, retrieval) on top of embedding results. This is the major reason why we put embeddings at the core of NLP system.

Recent developments on sequential text data lifted the state of the art of NLP significantly [5, 11, 14, 31]. However, our textual data is often about concrete annotation terms rather than long sentences and paragraphs, which means not easy to make use of context or word order information. The power of complex neural network architectures for feature extraction like convolutional neural networks [16], recurrent neural networks [11], or transformers [9, 31] would be limited in our scenario.

Although pretrained models are very helpful for prototyping, it is not optional to build in-house word embeddings tailored to Pinterest data, because 1) internal data distribution is very different from public corpus; 2) our training objective function is different. Continious bag of word (CBOW) or skip-gram models are essentially unsupervised in the sense that they predict word co-occurance in a context window. Our user engagement data is a kind of supervised information in nature. In this sense, we are strongly motivated to use supervised embedding training in the style of starSpace [33].

2.2 Multitask Learning

Our goal is to learn an all-in-one text embedding model capturing the inherent semantic information of textual data in Pinterest. We have to make the resulting model suitable for all three tasks: homefeed (HF), related pins (RP), and search (SR), with illustration in figure 1 and formal definition in section 4.1. This fits in the regime of multitask learning (MTL) [6, 26].

MTL aims to improve the learning by using knowledge in all or some of the given tasks. Successful MTL can produce a better model than single task learning, probably due to the fact that it augments training data compared to single task learning. Also by optimizing multiple models jointly, it potentially digs out information that generalizes across taks, for example, common representation or important features for all tasks, which is essentially a type of regularization that helps generalization.

The MTL motivation above exactly applies to our scenario here. We care about the maintainance cost of the embedding system in production and we hope to unify the embedding from three tasks. A user repin or click operation is a vote for the relationship between a query entity and a candidate entity. Mixing different entities together in learning will help us mine the underlying semantics that align to user engagement. The reason that two entities are presented together to a user in surface is because they have common words indexed in the backend serving system. So at the bottom of the retrieval logic chain, different entities are represented in a shared word universe. This observation leads us in the direction of shared word embedding [6]. Although MTL is not uncommon for classfication, there are not many works focusing on MTL word embedding itself in application [18]. We will discuss algorithm details in section 4.

2.3 Transfer Learning

The major motivation of PinText project is to have off-the-shelf text features independent on specific tasks such that downstream engineers can easily build task dependent models on top of it, for example, query to interest classification, search relevance model, etc. This means we train a model based on heuristics capturing semantic information of text and apply the learning to other tasks. In this sense, our work is closely related to transfer learning [21].

Since deep neural networks (DNN) refresh the computer vision benchmark by [15], initializing DNN with pretrained models has been very successful and has become pretty common practice. The NLP community typically initializes the beginning layer of DNN with pretrained word vectors like word2vec [19]. A typical issue with pretrained word embeddings is that word can have different semantic meanings in different context. Until recently, the NLP community has made some breakthrough with the idea of context dependent embeddings like ELMo, OpenAI GPT, BERT, etc [7, 9, 23–25, 27]. In particular the BERT model [9] refreshes many important NLP benchmarks. Unfortunately, those models cannot apply to our scenario easily for two major reasons: 1) our data is often not sequential like sentences or paragraphs; 2) inference efficiency is very important for Pinterest's scale where GPU or TPU is not always available.

Moreover, the first demand of all the three tasks we have in Pinterest is retrieval. This means we need realtime matching between query and candidates. The similarity defined in the embedding vector space, based on techiniques like latent semantic hashing, instead of supervised layer like softmax, is the most important learning objective here. The next sentence prediction task in BERT pretrained model is not relevant here. Therefore, we focus on the starSpace [33] style which can naturally handle retrieval task and does not require sequential text inputs. We are also able to do MTL optimization easily.

3 SYSTEM DESIGN

We introduce embedding based retrieval and ranking system architecture for the three HF, RP, and SR tasks.

3.1 System Overview

We can use textual terms to build an inverted index as in traditional information retrieval [2], in fact, this is how the retrieval system works at Pinterest. However, textual representation on complex entities has some clear limitations:

- *completeness*: Some terms are semantically close to each other while they have totally different spellings. For long tail queries, it is often difficult to find the terms in candidates;
- *compactness*: A user or a pin may have up to hundreds of annotation terms. It is hard to summarize the theme of such a complex entity by using concrete text terms. Lengthy textual representation results in ambiguity;
- *continuity*: When a partial match happens, we need a quantitative continuous way to define whether we should return candidates for a particular query.

Pinterest specializes in usual search that shows many possible inspirational ideas, rather than providing a concrete answer to a factual question. This open-ended nature makes the issues above

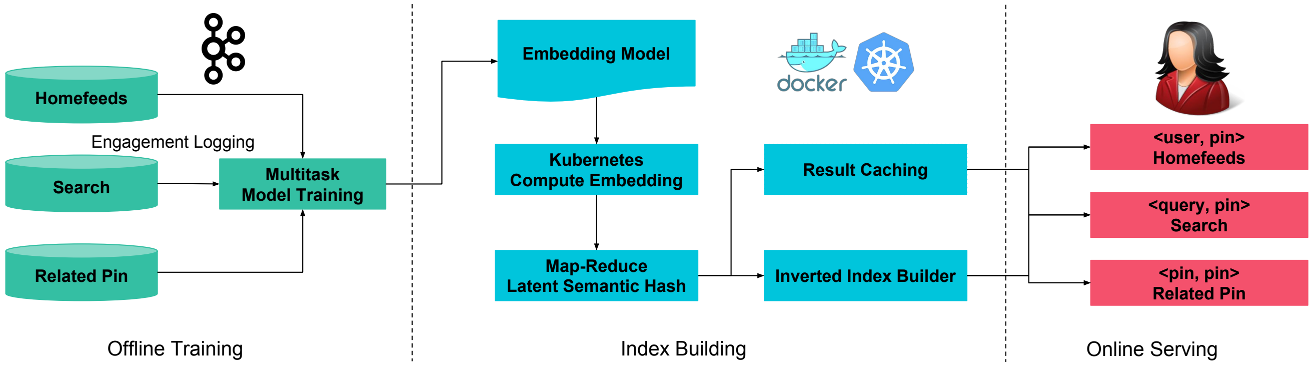


Figure 2: Simplified PinText system architecture consists of offline training, index building, and online serving. Left: We use Kafka to collect users' engagement data to construct training data. Middle: We use locality-sensitive hashing to compute embedding tokens and build inverted index for each pin. Embedding vectors and knn results can be cached. Right: use LSH token of embedding vectors to retrieve pins and use embedding similarity in ranking model.

even more obvious. Using a text embedding model, each entity can be compressed into a fixed length real vector. This provides compact semantic representations in a unified universe. Therefore, we can match a query to candidates by similarity measure in this common space, instead of relying on exact term match. Thus compactness and completeness issues are solved to a large extent. The similarity score can be used as a continious measure to filter candidates in a natural way, or as a discriminative feature in supervised models.

The motivation of the PinText embedding goes beyond information retrieval context. We actually hope to have a bootstrap system that generates feature vectors for any input text in downstream tasks. This can help engineers to start to develop their machine learning models involving text data quickly.

Therefore, we designed PinText as a backbone NLP system in Pinterest, with its major modules presented in figure 2. At a high level, it can be separated into three modules: offline model training, index building and result caching, and online serving. We use Kafka[1] to log user engagement data which is in turn converted to our training data. Considering the scalability issue, for example, millions of unique search queries come in per day, we rely on a Kubernetes [2] cluster to derive embedding vectors in a distributed way. After we derived the embedding of candidate entities including user, pin, and query, we employ locality-sensitive hashing (LSH) [8] to pre-compute tokens of each entity and build an inverted index based on them. At the online serving stage, we compute LSH tokens on-the-fly and use them to query against the index to implement k nearest neighbor (kNN) search. We also pre-compute kNN results for previously seen data and cache them to reduce CPU cost at online stage.

3.2 Embedding Model Training

The core of PinText is a word embedding model together with an efficient nearest neighbor search mechanism. To train a good model, high quality training data sampling is critical. It has to rely on a logging system composed of collection, transportation, and persistence that captures all user and pin interactions. Pinterest has one of the largest Kafka deployments in the cloud for this purpose and more. We use Kafka to transport critical events including impressions, clicks, hides, and repins to our data warehouse. With proper cleanup, we convert these events to a standard predefined data format such that hadoop jobs can query against it directly to sample training data. The embedding algorithm formulation is described in section 4. At this stage, we use a single server with Intel Xeon 8124M CPU, which has 18 cores working at 3.0 MHz. Training usually can finish in 48 hours for hundreds of millions of records. We are actively studying how to use multiple GPU machines for distributed training for next version.

3.3 Distributed Offline Computation

After we learn a word embedding dictionary, we derive the embedding of an entity by averaging its word-level embeddings. This is a hard-to-beat baseline method as shown in previous study [1, 10] and is easy to calculate and debug. Then we do locality sensitive hashing (LSH) [8] based kNN search for retrieval purpose.

Given our data scale, it is not a surprise that this learnt dictionary contains millions of tokens, where each token has a 300-dimension real vector. The total dictionary size goes up to several gigabytes easily. This model size makes it difficult to use Hadoop jobs directly, either by flattening entities' tokens followed by joining the embedding dictionary, or by broadcasting embedding dictionary to each map-reduce node. Moreover, we hope to make the PinText system flexible enough to include open sourced models, like fastText, for comparison purposes or cold start scenarios. There is no easy way to encapsulate unknown third party binaries in map-reduce framework. We use a Kubernetes (K8S) cluster to compute entity embedding vectors, followed by a hadoop job computing LSH tokens and kNN search.

Kubernetes Cluster for Embedding. Kubernetes is a system for automating deployment, scaling, and management of containerized applications open sourced by Google. In our scenario, a key difference between K8S and map-reduce system is that we have full computation power of K8S node such that we can use all scientific libraries for fast computation (e.g., numpy, scipy). We use docker [3] to wrap all the embedding computation logic into an image, then

[1] https://kafka.apache.org/
[2] https://kubernetes.io/
[3] https://www.docker.com/

schedule this image on a K8S cluster to compute text embedding of large-scale inputs.

Map-Reduce Job for LSH Token. After each entity is mapped to a real vector on K8S cluster, we are able to do kNN search between a query set and a candidate set. However, when the candidate set is large, the search becomes prohibitively slow. Although there is efficient local kNN solution like faiss [13], it requires to load whole candidate set to memory. We end up using LSH [8] for large sacle and using faiss for medium or small scale fast kNN search. It can retrieve billions of records in hours. We also send LSH tokens to search backend servers to build an inverted index, where the key is an LSH token and the value is a list of entities having this token.

Based on the tech stack above, we are able to scale up horizontally as our business keeps going up.

3.4 Online Serving

In the online stage, we need a mechanism to retrieve candidate pins in realtime given a query entity, which could be a query for SR, a user for HF, and a pin for RP. We have both caching and inverted index lookup working together.

Precomputed Key-value Map. As in last section 3.3, we are able to perform kNN search offline very efficiently. We cache the results as ⟨query, list of pins⟩ pairs with an in-house serving system called Terrapin [4]. Logically, it is equivalent to a dedicated cache system like Memcached or Redis, supporting realtime queries together with automatic data deployment management. In this way, we delegate most of the heavy online search to offline batch jobs, which are much easier to maintain and much cheaper to scale up.

Realtime Embedding and Retrieval. For some unseen text inputs, offline precomputation is unable to cover them. We deploy the learned word embedding dictionary to an online service and compute vectors and LSH tokens on-the-fly. Because we have built an inverted index of candidate entities' LSH tokens, the retrieval logic based on embeddings works exactly the same way as raw textual term based retrieval, which implies no further overhead development cost incurred.

To summarize, the PinText system includes a text embedding algorithm wrapped into a docker image which is scheduled on K8S cluster for deriving embeddings of large-scale inputs, and a map-reduce workflow to compute LSH tokens and perform kNN search, followed by inverted index building and persistent result caching. Because of the encapsulation of the embedding algorithm, PinText is flexible enough to bring pretrained models into Pinterest ecosystem. This merit is not negligible in the case of cold start where not enough training data is available to learn a model from scratch.

4 MULTITASK TEXT EMBEDDING

We present the multitask embedding algorithm in the PinText system inspiried by the starSpace work [33].

4.1 Task Definition

As illustrated in figure 1, we have three surfaces in Pinterest app. Each surface involes a pair of entities, i.e., ⟨user, pin⟩ in HF, ⟨query, pin⟩ in SR, and ⟨pin, pin⟩ in RP. Pinners are essentially voting for the relevance of two entities in a pair when they save or click a pin matching their interests. User engagement is equivalent to high quality data labeling (with proper filtering) at internet scale. This gives us a unique opportunity to derive a text embedding model that captures word semantics better than the open sourced models trained from Wikipedia or Google news. Let us start from positive pair definition here.

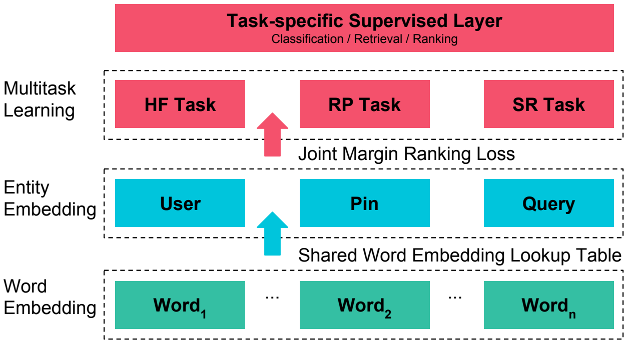

Task-specific Supervised Layer
Classification / Retrieval / Ranking
Multitask Learning
HF Task
RP Task
SR Task
Joint Margin Ranking Loss
Entity Embedding
User
Pin
Query
Shared Word Embedding Lookup Table
Word Embedding
Word1
...
Word2
...
Wordn

Figure 3: Multitask word embedding model architecture.

Definition 4.1. Given a query entity q and a candidate pin p, the engagement pair $\langle q, p\rangle$ is a **positive pair** if and only either of the two conditions is satisfied:

- user saves pin to her board;
- user clicks pin and stays the page for 30 seconds or longer.

The two conditions above correspond to *repin* and *long click* operations, respectively. We take them as voting for strong relevance between two entities. For the definition of *board*, please refer to figure (1a) which contains two boards "Alaska" and "Workout". A board is essentially a collection of pins. In this way, we are able to collect positive training data $\mathcal{E}^+$ for three learning tasks:

Definition 4.2. Depending on the choice of query entity q, we have three types of learning tasks:

- **homefeed** (HF): when q is a user, we learn the matching between a user's interests and a pin's topics;
- **search** (SR): when q is a search query, we learn the relevance between the query and pin's content;
- **related pin** (RP): when q is a subject pin, we learn the similarity between a subject pin and a candidate pin;

At this stage, if we instantiate q and p with textual representation, we can proceed to optimization formulation. In the search task, q usually exists as plain text. However, textual representation of a user or a pin is still unclear. We formally define them as following:

Definition 4.3. A **pin** p is a set of words $\{w\}$ where each word w_i appears in the union of pin's text metadata:

$$p := \left\{w | w \in \mathcal{T}, \mathcal{T} \in \{\text{title, descritpion, boardname, URL}\}\right\}$$

Definition 4.4. A **user** u is a set of words $\{w\}$ where each word w_i appears in the union of user's interests:

$$u := \left\{w | w \in \mathcal{T}, \mathcal{T} \in \{\text{interest}\}\right\},$$

where interests are provided by users at signup or derived from users' historical engagement.

[4] https://github.com/pinterest/terrapin

Definitions 4.3 and 4.4 are a logical abstraction, not an implementation. We have a dedicated Knowledge team and Personalization team focusing on developing machine learning based solutions to generate annotation terms of users and pins. It is beyond the scope of this paper to discuss their model details. Given the words of an entity, we simply use the mean of word embeddings as the entity embedding for downstream applications. Now we have every piece in place to move forward to formulation.

4.2 Multitask Formulation

We focus on learning word embeddings instead of learning user / pin / query embedding directly. The churn rate of whole collection of words is much smaller than the churn rate of pins and users. In this way, we avoid the problem of generalization to unseen entities at training phase. Also we avoid the unacceptable memory cost by not learning character n-gram level or sentence level embeddings.

Let $\mathbf{D} \in \mathbb{R}^{n \times d}$ denote the learnt dictionary, where n is the number of words, and d is the dimension of embedding vectors. Given a word w_i, we derive its embedding function $F(w_i) = \mathbf{w}_i$ by exactly taking $\mathbf{w}_i \in \mathbb{R}^d$ as the i-th row of $\mathbf{D}$. We compute the embedding of a entity q by averaging its word embeddings. Through this section, we use bold lowercase characters to denote embedding row vectors, e.g., $\mathbf{q} := F(q)$. In order to train a single task embedding model, we define the objective function $J(\mathcal{E})$ by enforcing the similarity on a positive pair greater than the similarity on a couple of randomly sampled background pairs.

Taking the search task as an example, where q is a search query:

$$J(\mathcal{E}_{SR}) = \sum_{(q,p^+)\in\mathcal{E}^+} \sum_{(q,p_i^-)\in\mathcal{E}_q^-} L\Big(S(\mathbf{q},\mathbf{p}^+), S(\mathbf{q},\mathbf{p}_i^-)\Big)$$

subject to

$$\begin{aligned} L(x,y;\mu) &= \max\big(0, \mu - (x-y)\big) \\ S(\mathbf{q},\mathbf{p}) &= \mathbf{q}\mathbf{p}^\top / (||\mathbf{q}||_2 ||\mathbf{p}||_2) \\ \mathbf{q} &= \sum_{w_i \in q} \mathbf{w}_i / |q| \\ ||\mathbf{w}_i||_\infty &\leqslant \gamma, 1 \leqslant \forall i \leqslant n \\ \mathcal{E}_{SR} &= \mathcal{E}^+ \cup_q \mathcal{E}_q^- \\ \left|\mathcal{E}_q^-\right| &= \rho,\ \forall \langle q,p \rangle \in \mathcal{E}_q^- \text{ is a random non-positive pair} \end{aligned} \tag{1}$$

where ρ for negative sampling ratio, μ for rank margin loss, and γ for radius of embedding vectors, are hyperparameters that need to be tuned through experiments. We fix L and S to be hinge loss and cosine similarity, respectively. In this way, for a particular entity q, we enforce that its similarity x with a positive entity is greater than its similarity y with a random negative entity by a margin μ. Otherwise it introduces a loss $\mu - (x - y)$. The heuristics here is that a good semantic embedding should capture users’ engagement.

The multitask learning objective function would be

$$J(\mathcal{E}_{MTL}) = J(\mathcal{E}_{HF}) + J(\mathcal{E}_{SR}) + J(\mathcal{E}_{RP}) \tag{2}$$

It is a simple aggregation of 3 learning tasks, where all tasks share the same word embedding lookup table, as visualized in figure 3. With this MTL objective function, we can do gradient decent with respect to each entry in embedding dictionary $\mathbf{D}$ to learn the word embeddings directly.

4.3 Implementation

We need to pay attention to practical details to make the result working in real systems.

Importance of each task. In equation 2, we did not put hyperparameters for each single task to control the tradeoff between their importance, although doing so may help benchmark performance. This aligns to our goal at the very beginning to build a fundamental text embedding system that can work reasonably well for all downstream applications. Not tuning the tradeoff between each task implies we hope the objective function captures the natural engaged traffic of Pinterest. We fine tune the training data sampling within each task. We only have to learn and maintain a single embedding dictionary.

Tradeoff between coverage and precision. For high precision optimization, we would enforce strong sample filtering logic on each task such that a positive training pair actually implies close semantic relationship between query entity q and candidate entity p. A smaller training set means shorter training time, which is important for fast model iteration. However, this also means we are losing coverage of words. To solve this dilemma, we first train a base model with high coverage, which is corresponding to loser training data sampling, then continue training it on a fine sampled dataset. We use the high coverage model to initialize embedding dictionary for all future model iterations. In this way, the embedding training still covers low frequency words but puts more efforts on more important words.

Parallel optimization. The objective equation 2 is additive in nature. This means we can calculate the loss on different examples independently. Accordingly, the gradient decent with respect to words can be done in parallel. The overhead of thread scheduling is small compared to computation cost. This gives the opportunity to fully use all threads. Due to the fact that computation is shallow, we did not use GPUs at this stage. We will revisit this in future.

5 APPLICATIONS AND EXPERIMENTS

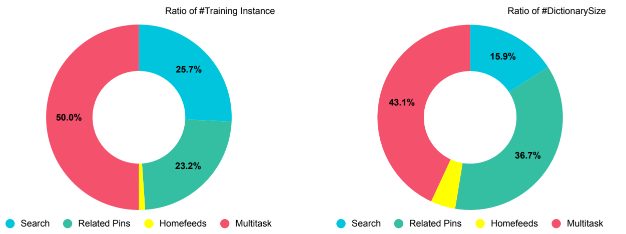
We present some empirical results of the PinText project and discuss its outcome in this section.

5.1 Interest Classification and MTL Config

We evaluate two classification tasks based on cosine similarity on top of the learnt word & entity embeddings. The first is a single label Query2Interest (Q2I) task, where the input is a search query and the class is 24 high level interests (e.g., *fashion, food and drinks, home decor*, etc). The second one is multilable Pin2Interest (P2I) task, where the input is a pin and the class is all possible interests (e.g., *chicken recipe* is a low level interest with high level parent interest *food and drink*). Each pin has 22 positive interest labels on average. The interests are manually labeled by human beings. We use them as ground truth set to evaluate classification precision. The interest taxonomy is built from Pinterest data and accessible to all advertising customers.

The baselines we evaluated in Q2I task including:

- fastText: pretrained EN model with wikipedia [4];
- GloVe: 6B version pretrained by Standford university [22];
- word2vec: model pretrained with Google news data [19];
- conceptNet: pretrained model by [28];



(a) The ratio of #training instances in the mixed multitask training.

(b) The ratio of #learnt dictionary size in the mixed multitask training.

Figure 4: Contribution of single task in the mixed multitask training.

- PinText-HF: single homefeed task PinText model;
- PinText-SR: single search task PinText model;
- PinText-RP: single related pin task PinText model;
- PinText-MTL: multitask PinText model trained by mixing HF, SR, and RP data together;

Table 1 shows query classification results. Our own PinText-MTL model produces significantly better precision than pretrained models. We draw some important conjecture and conclusion:

Multitask learning is indeed better than single task learning. Although Q2I classification is more related to SR task than RP / HF task, combining them together still helps to produce a better embedding model. This means word semantic information is carried by RP / HF, and that can be complementary to SR. This conclusion aligns to our MTL intuition. You may want to re-check the illustration in figure 1 to understand why semantics exist in all 3 tasks.

We conject other two possible reasons why our PinText model is better. First is supervised training data could be better than unsupervised data. Our in-house Pinterest engagement data is from user "voting" between a potential query and a candidate. The pretrained models are usually trained in an unsupervised way by CBOW or skip-gram mechanism which essentially predicts adjacent words in a context window. Second, in-house data could be better than outside data, even if both are supervised or unsupervised. The reason is people use Pinterest for clearly different purpose than using Wikipedia or news sites. For example, a typical case we deep dived is "allbirds", which is more related to birds and animals in pretrained models. However, in Pinterest it is strongly related to a shoe brand.

We further probed the merits of multktask versus single task training on multilabel P2I classification, as shown in table 2. Similar conclusion holds: MTL is consistently better than single task. However, RP has better performance on P2I task, while SR is better on Q2I task. This is strong evidence that single task is not enough to learn a complete embedding, due to either missing words or missing part of words’ semantics. Using MTL solves this severe problem to a large extent.

We did further breakdown analysis on the contribution of each task in figure 4. When we manually check semantic affinity between query and candidate, we found HR data is essentially noisy. We have to set high bar for HF pair to be selected as positive. When user comes to homepage, she is mostly browsing without a clear target in her mind. Additionally, users map to many interests while SR / RP scenarios are more focused. SR and RP have about 1:1 sampling ratio, and contribute about 98% of MTL training data together. If we compare the learnt dictionary size, the RP to SR ratio is about 2.3 with 1:1 instance sampling ratio. This aligns to the fact that pins have more words than search queries. It also shows clearly that MTL has better coverage than single task training due to augmented training data.

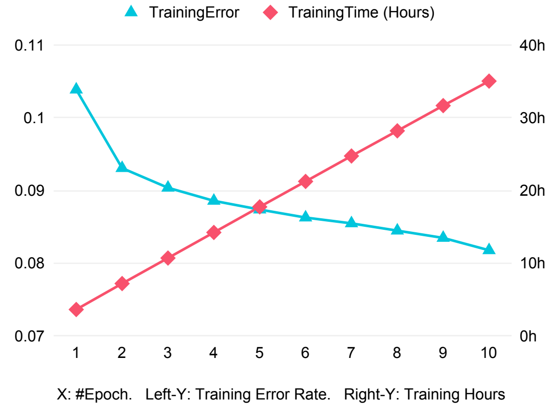


Figure 5: The training error and training time curve with 10 epoch.

We tracked the training error change and time cost increase with 10 epoches in figure 5. The classification accuracy is similar after 10 epoches. Once nice property of this MTL formulation is that training time is linear. This is very helpful for fast model iteration.

5.2 Search Retrieval and Query Broadmatch

In this section, we present a pin retrieval application based on the proposed PinText embedding system with the high-level architecture described in figure 2. We have multiple term based retrieval candidate source in serving backend as in a general information retrieval system [2]. The embedding based retrieval would be an additional candidate source to the existing sources, and has to pass all relevance based trimmers. Because the majority of results are computed offline and cached, the online infrastructure cost is negligible. Moreover, aftering converting embedding vectors to LSH tokens, token based retrieval is essentially the same as textual term based retrieval. This is important for agile online experimentation.

We first evaluated the embedding based retrieval performance offline with a labeled dataset for search task. The labeled candidates are the top retrieval candidates based on the current ranking system. This might be biased toward the existing system in terms of recall, and might contain more positive examples than negative examples. But it is fair to all PinText tasks and is valid to verify if MTL schema is good. We label each pin into {relevant, neutral, not relevant} by its relevance to the query and employ multiple person to label each pair. Then a pin is categorized into {positive, negative} by whether its average labeling score is above or below 0.70. We compute average normalized discounted cumulative gain (NDCG@K) on this dataset. Table 3 shows the results of PinText models with $K \in \{3, 4, 5\}$, which means offline NDCG results align to Q2I and P2I classification. This gives us high confidence to move forward with online applications. In an online A/B experment, we use current production as controlled group A and embedding based retrieval as an additional candidate source as enabled group B. We observe clear gains in enabled group in terms of revenue, click rate, and repin rate. We also aboserve clear drop in hide rate of pins. This means

Table 1: Query2Interest classification by 1-nearest-neighbor embedding cosine similarity.

Model	fastText	GloVe	word2vec	conceptNet	PinText-HF	PinText-RP	PinText-SR	PinText-MTL
Precision(%)	46.36%	49.36%	56.40%	65.84%	51.88%	72.72%	79.76%	**81.68%**

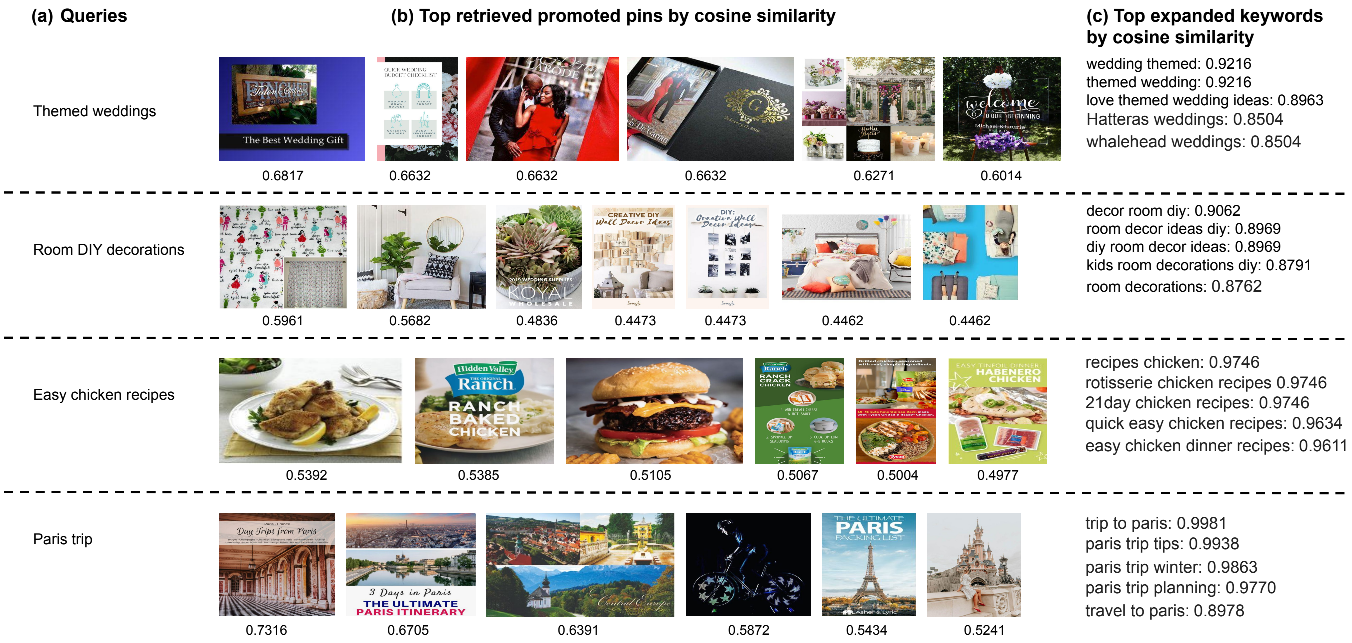

(a) Queries
(b) Top retrieved promoted pins by cosine similarity
(c) Top expanded keywords by cosine similarity
Themed weddings
The Best Wedding Gift
welcome to our beginning
0.6817
0.6632
0.6632
0.6632
0.6271
0.6014
wedding themed: 0.9216
themed wedding: 0.9216
love themed wedding ideas: 0.8963
Hatteras weddings: 0.8504
whalehead weddings: 0.8504
Room DIY decorations
Creative DIY Wall Decor Ideas
DIY: Creative Wall Decor Ideas
0.5961
0.5682
0.4836
0.4473
0.4473
0.4462
0.4462
decor room diy: 0.9062
room decor ideas diy: 0.8969
diy room decor ideas: 0.8969
kids room decorations diy: 0.8791
room decorations: 0.8762
Easy chicken recipes
Hidden Valley Ranch
RANCH BAKED CHICKEN
HABENERO CHICKEN
0.5392
0.5385
0.5105
0.5067
0.5004
0.4977
recipes chicken: 0.9746
rotisserie chicken recipes 0.9746
21day chicken recipes: 0.9746
quick easy chicken recipes: 0.9634
easy chicken dinner recipes: 0.9611
Paris trip
Day Trips from Paris
3 Days in Paris
THE ULTIMATE PARIS ITINERARY
Central Europe
THE ULTIMATE PARIS PACKING LIST
0.7316
0.6705
0.6391
0.5872
0.5434
0.5241
trip to paris: 0.9981
paris trip tips: 0.9938
paris trip winter: 0.9863
paris trip planning: 0.9770
travel to paris: 0.8978

Figure 6: Examples of PinText based retrieval and search keyword broadmatch in Pinterest. These top 4 queries exhibit diversity advantage of embedding based retrieval when query is not specific or exact term match is not good. The cosine similarity between queries and pins can also serve as a ranking feature.

Table 2: Multilabel Pin2Interest classification precision@Top-K ranked by embedding cosine similarity.

K	PinText-HF	PinText-SR	PinText-RP	PinText-MTL
K=5	91.55%	97.29%	97.85%	**97.93%**
K=10	91.35%	96.75%	97.28%	**97.39%**
K=15	90.86%	94.96%	**95.49%**	**95.61%**
K=20	**89.90%**	89.21%	**89.74%**	**89.89%**

Table 3: Offline Top-K NDCG of PinText Embedding based Retrieval.

K	PinText-HF	PinText-SR	PinText-RP	PinText-MTL
K=3	0.8426	0.8539	0.8527	**0.8587**
K=4	0.8479	**0.8616**	0.8584	**0.8618**
K=5	0.8508	**0.8642**	0.8612	**0.8644**

embedding based retrieval can be complementary to other term based retrieval sources with very good (if not better) quality. The column (b) in Figure 6 on some representative top queries shows the cosine similarity scores between queries and pins.

We introduce another production application *query broadmatch* of the PinText system. Among our 250 million monthly active users, a large portion of them are outside of United States. One challenge we are facing in new markets is the cold start problem, either because of less engagement or because we have no existing models built. This gives unique opportunity for a semantic embedding based service. Thanks to the merit of separating models from infrastructure in the system design in section 3, we are able to plug in any pretrained models with prior domain knowledge when we face the cold start problem. This allows us to scale up business logics in US to international markets quickly before we finish developing in-house models, which easily takes months to years to build bacause we either need long time to accumulate training data or the international market itself is too small to train a model from scratch. This saving in time can be very important for business expansion. Specifically, when a user inputs a query, we retrieve the expanded query first by PinText system then use it to query against inverted ads index. In this way, some of the previously unmatched queries would have more semantically meaningful results. By this broadmatch logic, we observe gains in click rate and gains in repin rate in an online A/B experiment.

6 CONCLUSIONS AND FUTURE DIRECTIONS

In this paper we presented a multitask embedding system used in Pinterest, which has shown promising results on both offline scientific metrics and online business metrics. This system is generic enough to plug in any word embedding models and flexible enough to work with real production pipelines.

We will continue investing in this direction. For the next milestone of PinText, we are actively working on three projects. The first one is distributed training infrastructure on multiple machines.

As our growth continues, more and more engagement data is being generated everyday. To fully mine the power of our new data, we must be able to train an embedding model reasonably quickly with it. The second project is embeddings as a service. We mainly use our Kubernetes cluster and map-reduce workflow for large-scale input. This works very well offline. However, when it comes to real-time embeddings, we see the difficulty of holding a big embedding dictionary in memory on task-dependent servers. It is necessary to have a standalone embedding service so all downstream applications can depend on it. The third one is a unified *master embedding* solution for users, pins and text, such that different objects can be compared in the same vector space directly. This will bring the scope of possible application scenarios to next level.

To summarize, PinText is proven system in production at Pinterest. We will deploy and launch more ads and organic applications of PinText in the future.

ACKNOWLEDGMENTS

We thank teammates from Pinterest Infra team and Ads team for seamless collaboration. We thank Stephanie deWet, Mukund Narasimhan, Jiafan Ou, and Nick Liu for the fruitful discussion.

REFERENCES

[1] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations, April 24 - 26, 2017, Palais des Congrès Neptune, Toulon, France.*

[2] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. 2011. *Modern Information Retrieval - the concepts and technology behind search, Second edition.* Pearson Education Ltd., Harlow, England.

[3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.* 3 (March 2003), 1137–1155.

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *TACL* 5 (2017), 135–146.

[5] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014.*

[6] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008.* 160–167.

[7] Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised Sequence Learning. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada.* 3079–3087.

[8] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 20th ACM Symposium on Computational Geometry, Brooklyn, New York, USA, June 8-11, 2004.* 253–262.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805

[10] Ricardo Henao, Chunyuan Li, Lawrence Carin, Qinliang Su, Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, and Yizhe Zhang. 2018. Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers.* 440–450.

[11] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[12] Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers.* 873–882.

[13] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734* (2017).

[14] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing.* 1746–1751.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.* 1106–1114.

[16] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. 1999. Object Recognition with Gradient-Based Learning. In *Shape, Contour and Grouping in Computer Vision.* 319. https://doi.org/10.1007/3-540-46805-6_19

[17] Jiwei Li and Dan Jurafsky. 2015. Do Multi-Sense Embeddings Improve Natural Language Understanding?. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015.* 1722–1732.

[18] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation Learning Using Multi-Task Deep Neural Networks for Semantic Classification and Information Retrieval. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015.* 912–921.

[19] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013). arXiv:1301.3781

[20] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* abs/1301.3781 (2013). arXiv:1301.3781 http://arxiv.org/abs/1301.3781

[21] Sinno Jialin Pan and Qiang Yang. 2010. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* 22, 10 (2010), 1345–1359.

[22] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar.* 1532–1543.

[23] Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers.* 1756–1765.

[24] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers).* 2227–2237.

[25] Matthew E. Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018. Dissecting Contextual Word Embeddings: Architecture and Representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018.* 1499–1509.

[26] Sebastian Ruder. 2017. An Overview of Multi-Task Learning in Deep Neural Networks. *CoRR* abs/1706.05098 (2017). arXiv:1706.05098

[27] Sebastian Ruder and Jeremy Howard. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers.* 328–339.

[28] Robert Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.* 4444–4451.

[29] Joseph P. Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word Representations: A Simple and General Method for Semi-Supervised Learning.. In *ACL*, Jan Hajic, Sandra Carberry, and Stephen Clark (Eds.). The Association for Computer Linguistics, 384–394.

[30] Shyam Upadhyay, Kai-Wei Chang, Matt Taddy, Adam Kalai, and James Y. Zou. 2017. Beyond Bilingual: Multi-sense Word Embeddings using Multilingual Context. In *Proceedings of the 2nd Workshop on Representation Learning for NLP, Rep4NLP@ACL 2017, Vancouver, Canada, August 3, 2017.* 101–110.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA.* 6000–6010.

[32] Jason Weston, Sumit Chopra, and Keith Adams. 2014. #TagSpace: Semantic Embeddings from Hashtags. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL.* 1822–1827.

[33] Ledell Yu Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, and Jason Weston. 2018. StarSpace: Embed All The Things!. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018.* 5569–5577.