Query2Interest Classification at Pinterest

Jinfeng Zhuang
jzhuang@pinterest.com
Pinterest
Seattle, WA

Jinyu Xie
jxie@pinterest.com
Pinterest
San Francisco, CA

Yunsong Guo
yunsong@pinterest.com
Pinterest
San Francisco, CA

Heath Vinicombe
heathvinicombe@pinterest.com
Pinterest
San Francisco, CA

Rui Li
rli@pinterest.com
Pinterest
San Francisco, CA

Balaji Krishnapuram
bkrishnapuram@pinterest.com
Pinterest
San Francisco, CA

Roelof van Zwol
rvanzwol@pinterest.com
Pinterest
San Francisco, CA

ABSTRACT

We solve a short text classification problem named by query to interest (Query2Interest, or Q2I) classification based on Pinterest data. Q2I predicts a list of interests as a user searches on Pinterest app. This module is important to applications including query understanding, search relevance, and ads targeting. However, it is unclear if pre-trained deep learning models like BERT based on large training corpus perform well for queries with less than 5 words. To attack the short text classification challenge, we conduct a thorough study of deep learning models performance on Q2I and propose a novel pin annotation-based query augmentation algorithm which lifts all model performance significantly. Our contributions in this paper include: 1) implementing common DNN models and comparing their performance in the Q2I setting; 2) adapting and fine-tuning pre-trained NLP models including Transformer, BERT, GPT 2, XLNet and examining if transfer learning can achieve state-of-the-art as in NLP benchmarks; 3) proposing a multimodality data augmentation approach that not only boosts classification precision significantly but also allows simpler models like fastText continue working almost equally well as complex DNN models.

CCS CONCEPTS

• **Information systems** → **Query intent**; *Query representation*; • **Computing methodologies** → **Supervised learning by classification**; *Neural networks*;

KEYWORDS

query understanding, text classification, NLP models

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

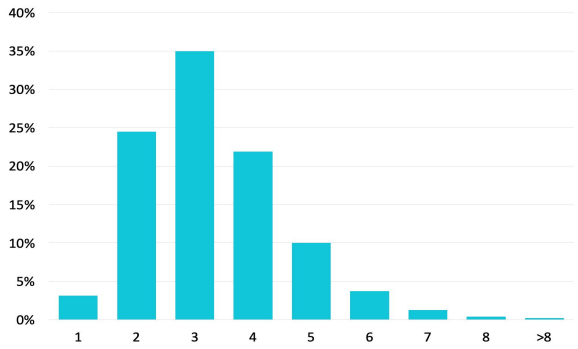


Figure 1: Query length distribution on a crawled Pinterest data set. On average, each query has 3.3 words. The queries with length <= 5 takes 94.5% of all queries. This makes Query2Interest classification essentially challenging.

ACM Reference Format:
Jinfeng Zhuang, Jinyu Xie, Yunsong Guo, Heath Vinicombe, Rui Li, Balaji Krishnapuram, and Roelof van Zwol. 2020. Query2Interest Classification at Pinterest. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Pinterest has become one of the major visual discovery and sharing platforms on the web. Today, there are more than 300 million people around the world who use Pinterest every month looking for motivations and inspirations[1]. Apparently, search is an essential surface to bring users to their interests. We conjecture that there might be difference between Pinterest and Google search. When users search in Google, they usually know what they want in mind and issue specific search queries describing that intent. On the other hand, Pinterest's search surface is strongly related to inspiration, aligning to the company's mission "to bring everyone the inspiration to create a life they love". Users have a higher level interests behind the specific search query terms. Before that interest is satisfied, they keep browsing and searching to instantiate their ideas. This means

[1]https://newsroom.pinterest.com/en/300million

digging out this interest is very important in order to help users discover and satisfy their true intents.

In this paper, we focus on the query to interest classification (Q2I) problem. With the predicted interests, search backend can retrieve broader candidate pins and potentially more relevant pins, if the input query is not specific enough or not accurate enough. In the case of ads targeting, using interest as targeting type can improve recall significantly and simplify advertisers' data operations. For example, when a user searches "chicken", there are very different styles of chicken recipes, including BBQ chicken, chicken salad, fried chicken wings, etc. If we can map the query to a predefined interest "chicken recipe", we are able to retrieve all pins matched to this interest. It naturally provides better diversity around user's intent here. Moreover, by comparing this query's interest to pins' interests, it can also be used as search relevance features for better ranking purpose.

Query2Interest is a classical text classification problem under the setting above, with the characteristics that input text is very short. There are a very broad set of mature machine learning algorithms that can be applied to this problem directly. It is very interesting to study how classical algorithms perform on this real application scenario. In particular, deep learning models are very successful on classification tasks, for example, dynamic memory networks [15], fastText [1], convolutional neural networks (CNN) [13, 40], recurrent neural networks (RNN) [17, 24], hierarchical attention networks (HAN) [38], etc. We are also aware of recent breakthroughs in natural language processing community, where pre-trained models based on large corpus high-quality data can be used in downstream tasks in a transfer learning setting, either by using their embedding output as input features (feature based transfer learning) or by fine tuning their internal network parameters (fine tune based transfer learning). Some representative works in this category include but are not limited to Transformer [33], ELMo [25], GPT 2 [26, 27], BERT [7], MT-DNN [21], XLNet [37] etc. The transformer [33] architecture as a feature extractor plays a central role for the success of these advanced NLP models. However, it is unclear whether transformer still has the power when input is very short, because there is no long enough text context to allow the self-attention mechanism in transformer to find out the internal structures of input.

In this paper, we set up a fair benchmark for different deep neural network (DNN) models, including pre-trained NLP models on large corpus of text data, to better understand how different models perform on an important real problem. We propose to expand an input query with text information derived from its top pins in search results, with the motivation of increasing query length to fit DNN models' complexity better. Data augmentation has been proved to be a successful approach for short text classification [8, 16, 31, 35, 36]. It is intuitive that adding more meaningful words will help the model to decide "class boundary" more easily. We explore an annotation technique that mines both the visual and textual information of the top pin in search results to expand input query, assuming the top pin is closely related to query's interests and a pin usually has richer information than query, for example, title, description, etc. With a robust Query2Interest classifier, it is possible to do interest based search ads targeting, interest based search relevance model and guided search term recommendation, etc. To summarize, the contribution of this paper includes:

- A fair model architecture comparison of common deep learning models for short text classification, which may give hints to other ML researchers and practitioners about which models to use;
- A thorough study of modern pre-trained NLP models on query classification with limited context. We adapted the most representative STOA models to Q2I and presented a fair comparison to show that they may not necessarily work best in this setting;
- An effective data augmentation approach for Q2I. Specifically, we derive the top pin's annotation terms and use them to expand the given query. This approach lifts top 1 classification precision by 10+%. We also provide some possible important applications with it.

We conclude that fastText with query augmentation can rival more complex deep learning models. Pre-trained NLP models take significantly longer training and inference time and may not meet latency / throughput requirements in real applications.

2 RELATED WORK

Q2I is a subproblem of query understanding. It is common practice to do tokenization, spelling correction, stemming and lemmatization, and proper rewriting and expansion on raw input queries before sending them to backend document index in a typical search engine. However, to understand the intent of a search query, it is important to map them into higher level interest concepts. This problem has been actively studied in web search and data mining community, for example, classifying query into informational or transactional categories [12], intents mined from logs [8, 36], Wikipedia keywords [16], etc.

We study this problem with Pinterest data, where the interest list in their ads targeting interface can be used as clear intent definition. Q2I plays an important role for interest based search ads targeting and personalized homefeed ads targeting. Pinterest has its uniqueness of data distribution. To our knowledge, there are no off-the-shelf solutions to be plugged into Q2I yet. One common sense about the essential challenge of query understanding is that the input query is so short that very limited features can be derived from them for the ML models. Thus data augmentation becomes a natural strategy of solving this challenge, for example, retrieval on Wikipedia [16], pre-defined knowledge base [35, 36], search engine based features [8], etc. Apparently, Pinterest is mainly a visual discovery platform for the inspiration of people's daily life, which means there could be a big gap between Pinterest's query distribution and previously built knowledge base. In this paper, we propose a novel approach to use the top pin of search result to help derive augmentation terms.

After completing the query expansion phase, we need to train a text classification model. Given this is a very classical problem, there are many successful algorithms can serve here. Support vector machine (SVM) [5] with proper feature engineering like tf-idf weighting can achieve good empirical results. More recently, it has been found that deep learning models where text representation and classification layers are learnt together in an end-to-end style

beats traditional SVM base solutions [10, 38]. It is very interesting to study the performance of those DNN architectures on Q2I.

Similar to the success of transfer learning in computer vision community, researchers have developed pre-trained natural language processing models based on large corpus of open sourced text, with the most representative examples like BERT [7], GPT-2 [26, 27], XLNet [37], and RoBERTa [22]. Those models refresh all of the NLP benchmarks including classification tasks. Another interesting set of work on knowledge distill [18, 29] is to make these powerful models lighter for fast inference and training. We only focus on the classification accuracy perspective in this paper. The transformer architecture [6, 33] plays a central role in their success. It is unclear whether transformer still works well when input text is so short that no syntax structure can be mined. In this paper, we fine tune the pre-trained state-of-the-art NLP models on this Q2I problem. To our knowledge, this paper is the first report of NLP models' performance on very short text input.

3 QUERY TO INTEREST CLASSIFICATION

In this section, we present the problem formulation and the proposed solutions.

3.1 Problem Definition

Q2I is a typical multi-class and multi-label text classification problem. It specializes in that we focus on Pinterest's query data only. Formally, given a set of labeled data $\{q, y\}$, where q is a raw search query and y is a pre-defined interest, the *Query2Interest* task is to train a classification model such that it can predict a list of interests for any input query as accurately as possible. Here we use the top tier 24 interests in Pinterest's interests taxonomy as labels. A typical example of this Q2I labeled data looks like: {*wood grain cabinets kitchen*, *__label__home_decor*, *__label__diy_and_crafts*}.

To understand the context of the problem, we must explain the definition of *pin* first, which plays a central role of the Pinterest app. In a nutshell, a pin is an image with metadata including title, description, and the page containing this pin. A *pinner* is a Pinterest user who saves a pin to a *board*, which is essentially a collection of pins. The board information like board title is very useful for the theme of the pins in it, because it is created and added by pinners. We refer to Figure 1 in [41] for an illustration of pin, board, and pinners. One of the motivations of Q2I is to help return related pins in the search surface.

As illustrated in figure 1, each query contains only 3.3 words on average. This makes Q2I essentially challenging because there are very limited features can be derived. Let's consider the extreme case: each query has only one unique word (no two queries share any common word), then no matter how good the classifier is, there is no way to generalize on unseen queries. Moreover, the class is essentially vague even for human beings if query is very short. The prediction precision would be improved if we can make queries more specific. It is a natural strategy to augment data to make input queries longer and carry more context information.

3.2 Interest Taxonomy as the Label Space

A couple of years ago, Pinterest decided to enable an "interest" based ads targeting interface, which named the most popular keywords as interests, and organized them in a tree structure (a taxonomy) for advertisers to pick nodes from for their campaigns. For example, if Home Depot creates a campaign for the interest "Living Room", the users interested in "Living Room" in general, as well as users interested in "Sofas", "TV Stands", "French Living Room Style", "Living Room Decor", etc., will all be exposed to the ads of this campaign. Having realized the potential that such a structured knowledge representation provides, Pinterest decided to investigate ways to improve the quality of the taxonomy; and augment the scope of the taxonomy as needed to provide coverage for all Pinterest content. The interest taxonomy has now expanded its usages beyond ads targeting to search, shopping, and home feed recommendation. The details of how this taxonomy is built is available at [9].

This interest taxonomy has taken the central role for characterizing the interest behind users, pins, and search queries. By predicting their interests and matching interests for retrieval and ranking, we observe huge lift in both ads and organic scenarios. This interest taxonomy is exactly the label space of Q2I.

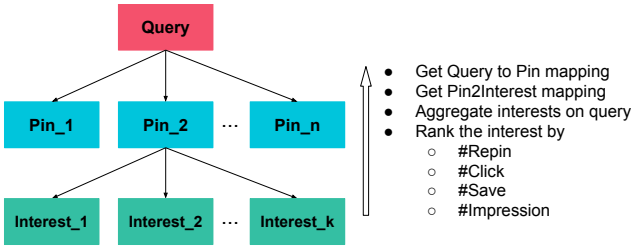


Figure 2: Label sampling from search engagement. For each $\{p, y\}$ pair, we have different types of engagements. We learn the weights of each type of engagement such that the combined label set overlaps the most with a human judged golden set.

3.3 Derivation of Labels

There are two ways to derive the labels, one is by human judgement, the other is by automatic derivation from user engagement. The techniques are far from trivial, and the labeling quality dominates the model quality. Let's discuss the details and conclusions in this section.

3.3.1 Human Judgement of Labeling. For the first version of Q2I model, we employ vendors to do label judgement. We present $\{q, y\}$ pairs to vendors and let them label three categories: positive, negative, and unknown. Then we use majority voting to decide the ground truth labels. The real challenge is that the number of interests are tens of thousands. It is totally prohibitive to label all of them, because the labeling cost is simply linear to the number of pairs. We use PinText system to select top semantically similar interests as candidates, which reduces the scale from several thousands to several dozens. That is exactly how we get the label for Q2I V1, and it performed reasonably well. However, we must improve this because it cannot scale up. Model quality is very sensitive to the size of training corpus.

3.3.2 Engagement based Interest Derivation. We have a backend logging system tracking the query to the engaged pin list mapping $\{q, [p_1, ..., p_k]\}$, where engagement can be click, save, repin, long

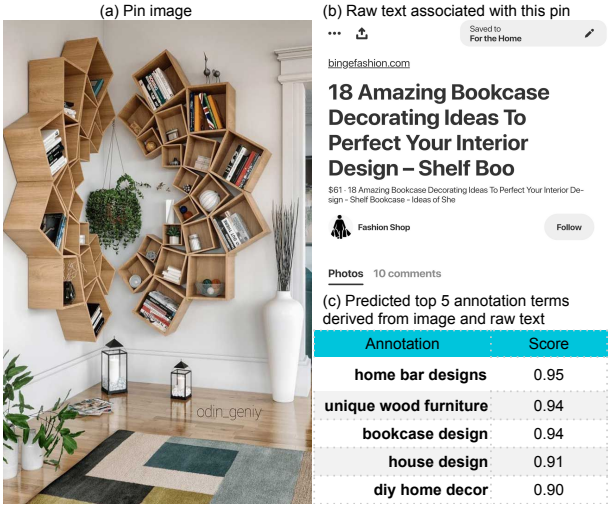


Figure 3: Example of top derived annotation terms from a pin.

click, etc. Given that we have a reliable Pin2Interest (P2I) classification system [19] generating interests at pin level $\{p, y, s\}$, where s is a score measuring the confidence that y is a correct interest of p, we are able to aggregate the interests at pin level to query level to construct the labels of query. It is important to note that this does not imply Q2I's performance is upper bounded by P2I. For a reasonably popular query, it usually has hundreds of engaged pins in a two year window. We have a relatively big pool of interests to select from to derive query's interests.

The sampling logic is shown in figure 2. Instead of simply choosing one of the different types of user engagement, we learn a linear combination of them such that the ranked interest list is mostly similar to the interest list ranked by human judgement. In this way, we are confident that the derived labeled data quality rivals human labeled data quality. We have a group of engineers review this together and it turns out that the former are often slightly better than the later. This is not surprising because they are essentially millions of pinners' majority voting results.

3.4 Augmentation by Pin Annotation

As aforementioned in section 3.1, the essential challenge of Q2I is that the input text is too short. We focus on a query augmentation approach in this section. The key assumption we make is that the top engaged pins in search results encodes interests closely related to the input query. This is the same motivation as in label derivation from engagements. However, deriving text information from a pin is not always trivial [34]. We are facing two challenges here:

- *Empty Text*: Some pins have a title and a short text description of the content. However, there are also pins with only images and no meaningful texts;
- *Noisy Text*: To solve Q2I task, we hope to expand query with accurate pin's content, which means precision is more important than recall here.

To solve the first one, we extract text from a variety of candidate sources, for example, we employ a pre-trained image classifier [39] to predict the top classes of visual content. In this way, even if there is not much meaningful text available, we are still able to find

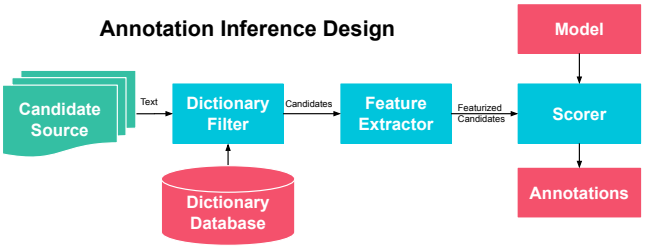


Figure 4: Annotation inference workflow. We use the top annotations of the top engaged pins to augment input query.

out the true concept if the pin, which in turn helps to derive the interests of the query. For the second one, we first tokenize the word-level n-grams with n from 2 to 6. Then we use a predefined dictionary service to filter out rare or unknown words and phrases. After collecting the candidate terms in these two steps, we extract features per [pin, annotation candidate] pair then train a GBDT classifier [3] to rank annotation terms. We use such per annotation point-wise model instead of pairwise or list-wise models [20], to make this relatively light-weight for auxiliary purpose, considering the facts that: 1) we do not have to guarantee recall because we can use multiple pin's annotation terms to compensate missing interests; 2) we only care about the precision of top few predictions instead of the overall precision. It probably does not hurt if we miss some concepts of the pin, but it is bad to introduce noisy non-related text to the input query. Algorithm 1 gives a clear description of this annotation generation process.

Algorithm 1: Annotation Generation per Pin

input : A Pin and its Metadata
output: A List of Annotations for Input Pin
1. Extract text from a variety of sources:
   - pin's title and description
   - titles of boards it's saved to
   - summary text of the landing page it links to
   - optical character recognition
   - detected objects in image
   - URL of pin and landing page
2. Tokenize text into ngrams and match them against a predefined dictionary
3. Extract features from annotation candidates (not full list):
   - the frequency of candidate in pins
   - inverse document frequency of candidate
   - candidate source (title is more important than body text)
   - embedding similarity between the candidate and the pin
   - visual embedding projection to text space
   - language encoding of terms
4. Run GBDT model to score each [pin, annotation] pair
5. Select top annotations with score greater than a threshold

Figure 3 presents a good example of annotations for a bookshelf pin. By appending the top annotation terms to input query, the average length is significantly improved as shown in figure 5.

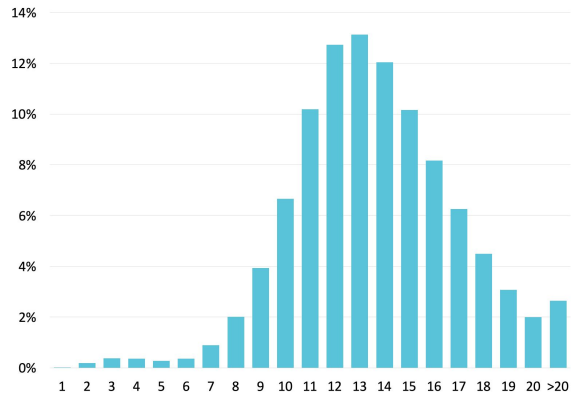


Figure 5: Query length distribution after annotation augmentation. On average, the length of each query is increased to 13.6 words from original 3.3 words. This makes short query to typic sentence length.

3.5 Deep Neural Network Architectures

After we augment query with the derived pin's annotation terms, we need to choose a proper text classification model to train. This is not a trivial problem given there are dozens of existing algorithms can work for this purpose. We only focus on deep neural networks solutions here given the fact that they usually produce the best performance. The typical DNN models learn both token representations and supervised layers together. Given the short length of the query, it is unclear which models perform better than the others. Therefore, we study the most common DNN architectures as below. Without stated otherwise, we always use a fully connected layer (dense layer) with softmax as activation function for prediction layer.

fastText. It uses a dense layer on top of the average of word embeddings of input text [10]. It is known that fastText can produce similar performance to DNN models with much faster training speed.

Convolutional Neural Networks. Similar to CNN in computer vision community, we use multiple convolutional filters with different filter size on word embedding sequence to extract features. CNN is good at capturing local features but hard to capture long distance dependency. In the Q2I case, input text is usually short, so the long dependency issues does not exist. We expect it is able to generate a good model. Specifically, we implement the algorithm described in [13] with convolution filter size in $\{1, 2, 3\}$.

Recurrent Neural Networks. RNN is particularly good at sequential data. Given natural language is sequential in nature, RNN algorithms like long-short term memory networks [11] dominates NLP before the BERT model comes out. We use bi-directional gated recurrent units (GRU) [4] to extract features and concatenate the embeddings of last token as query representation. Usually RNN has the drawback that it cannot be trained on parallel because current unit depends on the hidden state of the previous unit. Again due to our input has usually less than 20 words, it is not a problem at all.

CNN + RNN Vertical. To fully explore the modeling ability of CNN and RNN, we try to combine them together to see if it fits the data better than CNN or RNN alone. We first use CNN to get the hidden embedding sequence then put GRU on top of it to model this sequence.

CNN + RNN Horizontal. The other way to combine CNN and RNN is to use them separately and then concatenate the hidden units together. Assuming they compensate each other, this might generate better features.

Hierarchical Attention Networks. One limitation of RNN is that it only takes the last hidden unit as input, which means there is no attention of which prior units are most important for current unit. HAN [38] is a model uses attention mechanism on word level to get a better sentence embedding vector. Then another attention layer on sentence level vectors to derive the whole document embedding. It is quite intuitive HAN does a better job than simply averaging word embeddings like fastText. In the query augmentation case, we treat the original query and each annotation phrase as the sentence.

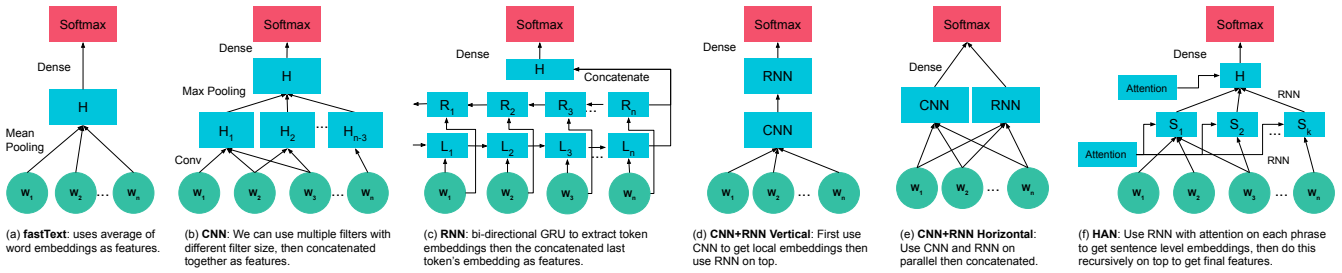
We plotted the high-level model architectures in figure 6. Comparing DNN models with fastText, they have more parameters to explore word relations inside the input text like local / sequential or attention based features, with the cost of longer training and inference time. We did not try very deep models [30] for this Q2I task due to the same reason that input length may not long enough for the potential complexity benefit of high depth. There are also many classical classifiers other than DNN models, like naive bayesian, support vector machine, random forest, etc. Given fastText and DNN models have been proven effective on text classification, we would not evaluate these models in this paper. Instead, we strict the scope to conclude whether DNN models can work better than fastText on this very short input scenario. This means we should study modern neural networks based NLP models, as in the next section.

3.6 Pre-trained NLP Models

Transfer learning has been very successful in computer vision community. After Word2Vec [23], it is known to be useful to initialize the token embeddings with a Word2Vec model pre-trained using Wikipedia data. A recent breakthrough in NLP is the BERT model [7]. Downstream applications can just fine tune it to achieve state-of-the-art performance on most of NLP tasks, instead of training a complex model from scratch. These models are trained on natural language sentences. But our input here are concrete phrases, which may violets the assumption of mining sentence structure with transformer [33]. Therefore, it is a very interesting problem to study if we can transfer the knowledge in these advanced models in Q2I prediction.

BERT. The BERT model refreshes all NLP benchmarks in 2018. It is essentially a stacked encoder from transformer to model natural sentence structure, with predicting masked words and next sentence as training task. This architecture solves the limitation of extracting long distance dependency of CNN and the limitation of one-direction sequential information of RNN, plus it can be trained on parallel. BERT model works very well for classification task too based on various ways of fine tuning the pre-trained model [32].

GPT/GPT 2. Because BERT uses bi-directional information, it is not so natural for generation task, where only previous tokens of current token can be used at inference. OpenAI has upgraded their GPT model [26] to GPT 2 model [27] with better and bigger training data. We would like to fine tune GPT 2 and check how its performance on short queries.



(a) **fastText**: uses average of word embeddings as features.
(b) **CNN**: We can use multiple filters with different filter size, then concatenated together as features.
(c) **RNN**: bi-directional GRU to extract token embeddings then the concatenated last token's embedding as features.
(d) **CNN+RNN Vertical**: First use CNN to get local embeddings then use RNN on top.
(e) **CNN+RNN Horizontal**: Use CNN and RNN on parallel then concatenated.
(f) **HAN**: Use RNN with attention on each phrase to get sentence level embeddings, then do this recursively on top to get final features.

Figure 6: The common deep neural networks architectures we studied for Query2Interest classification. All DNN models share two common layers: a word embedding layer at the bottom and a dense layer with softmax as activation function on top.

XLNet. XLNet [37] tries to solve some limitations of BERT including the discrepancy between training and test data and the input length limitation [6]. It reports better performance on all the NLP benchmarks.

There are other excellent transfer learning models before and after BERT, some representative examples include ULMFiT [28], ELMo [25], MT-DNN [21], RoBERTa [22], etc. Given BERT usually beats the models before it and the models after BERT on classification task are not necessarily bigger than BERT, we would only focus on the three models above. Enumerating them would be beyond the scope of this paper.

4 EXPERIMENT AND APPLICATION

We evaluate the solutions proposed in the last section and propose some applications based on Q2I.

4.1 Experiment Setup

we crawled 1M queries from Pinterest as training set, 10K queries for tuning set to help try different hyper-parameters, and 10K queries for test set. There are 24 top level interests as labels. We use Keras to implement all the DNN models from scratch for fair comparison (in case different implementation details result in big performance gap). The common settings of all these DNN models include:

- the first layer is always a word embedding layer as in figure 6;
- the last layer is always a dense layer with softmax as activation function;
- we always use Adam optimizer [14] to learn the network parameters.

For the transfer learning setting, we used cased BERT large model, 345M version of GPT 2 model, and large XLNet model. We use top 1 prediction precision as evaluation metric. All the hyper-parameters can be found in the released code. We do not list them here to save space.

In addition to the tune-tuning based transfer, we also try a feature based one: we use the pre-trained transformer [2] in Tensorflow hub to compute query embeddings. We do not fine tune the internal weights of transformer at all. The goal is to verify if pre-trained word embeddings has similar performance to models trained or fine tuned in an end-to-end manner.

We evaluate both original queries and the augmented queries. Some examples of query augmentation:

- I am not I $\mapsto$ *bible verse wallpaper iphone, two word quotes, background, faith quotes, gospel quotes*
- pirate food ideas $\mapsto$ *pirate theme party, pirate themed birthday, pirate party, pirates, pirate birthday party*
- raised shower area $\mapsto$ *bathroom windows, shower renovation, walk in tub shower, minimal bathroom, window in shower*
- night markets taiwan $\mapsto$ *places to travel, taipei travel, taipei taiwan, taiwan night market, taiwan travel*

In the first example, it would be hard to know the intent of the query without augmentation.

4.2 Query2Interest Precision

We present the evaluation results in table 1. Some key observations we can draw:

First, data augmentation significantly improves Q2I precision across all the algorithms we tried. Both DNN and pre-trained NLP models benefit a lot from it: the least improvement is on XLNet by 11.6%, and the most improvement is 14.2% on fastText. The fact that the relative gain of XLNet is smaller probably means, the simpler of the model structure, the more helpful the side information is. For pre-trained models XLNet, it already carries a lot of context information of each token. Thus it produces better P@1 than fastText before augmentation. On average, query augmentation improves all models by 12.8% (the feature based transformed not counted). The precision is improved to 90+% from below 80%. This result aligns to prior works on data augmentation on very short text classification problems. Compared to textual information based augmentation, our annotation term expansion is essentially a multimodality solution which combines both textual and visual information.

Second, fastText might still be good enough as the "go to solution" for Q2I classification problem, depending on how thirsty the applications are about prediction precision and the serving time constraints (like latency). Before data augmentation, DNN models are usually 2$\tilde{3}$% more accurate than fastText. This is intuitive because DNN can model fined information like word orders better than fastText. For example, the convolution filter with size n in CNN followed by a max pooling mines word-level n-grams information, which may be lost in the averaging of fastText. RNN structure takes input sequentially such that it preserves word order information. However, after data augmentation, the best DNN model bi-directional GRU is only 1.2% more accurate than fastText. Considering the introduced complexity at both training and inference, probably fastText is still the best algorithm for production. In

Table 1: P@1 of Query2Interest classification with annotation.

Model	RawQuery	+Annotation	Gain
FastText	76.78%	91.04%	14.26%
TextCNN	78.23%	91.37%	13.14%
TextRNN	**79.31**%	**92.24**%	12.93%
CRNN-Ver	77.96%	91.10%	13.14%
CRNN-Hon	78.44%	91.21%	12.77%
HAN	N/A	**92.11**%	N/A
Transformer	64.24%	88.54%	24.30%
BERT	**79.88**%	**92.13**%	12.25%
GPT-2	77.97%	90.85%	12.88%
XLNet	**79.24**%	90.86%	11.62%

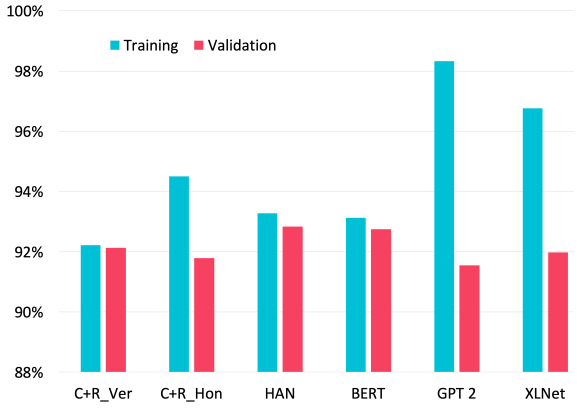


Figure 7: P@1 of training versus validation set of most complex models.

particular, Facebook has the C++ implementation of fastText, which usually takes a few minutes on 1 million queries. Data augmentation not only improves all models' performance, but also provides the possibility of using simple models over complex models in real production.

Among all the DNN models and pre-trained NLP models, RNN solution gives the best performance consistently before and after query augmentation. This implies bi-directional word order plays a role for encoding whole sequence's semantics. In general, fastText and CNN are not so good at capturing order information. Simple combination of CNN and RNN is not helpful, potentially caused by the overfitting. Another possible explanation is that the problem is not complex enough to match their model architecture complexity.

Third, transfer learning still works even on this very short input problem. This is sort of surprising given the basic building block is transformer structure. The self-attention mechanism of transformer is good at capturing the internal sentence structure. In the case of search query, it only has 3.3 words on average before augmentation. We assume there is no need of attention. So it is unclear the good performance of transfer learning models is from the model structure itself or pre-trained data. One interesting finding is that feature based transfer learning with transformer only generates the worst performance, with most significant relative gains of augmentation. The learning is that fine tuning on model internal parameters are important, especially when data distribution changes severely.

To better understand the behavior of complex DNN models, we plot the training / validation set performance in figure 7. The basic

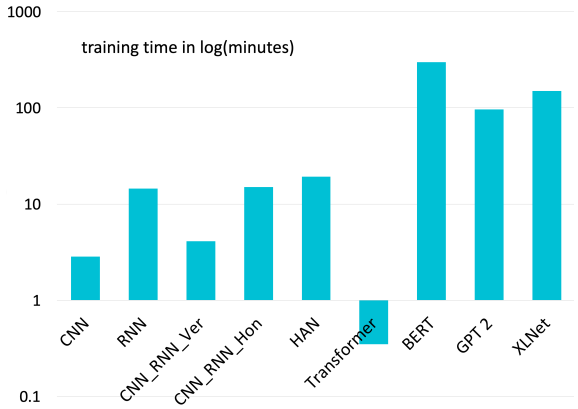


Figure 8: Training time in log-scale minutes of DNN models.

assumption here is that the more complex the model, the more possibility of overfitting. FastText, CNN, RNN, and pre-trained transformer generates very close precision on training and validation set. We omit them from the figure. First, we hope to understand if combining CNN and RNN would overfit training data. It looks like that the vertical combination (i.e., going deeper) does not overfit in our implementation, but the horizontal combination (i.e., going wider) has the overfitting risk. HAN is robust to overfitting too. Second, for the modern transfer learning NLP models, BERT has similar training / validation set performance, but GPT 2 and XLNet obviously fit training data better without improving test set precision. It is very interesting to understand why BERT is robust to overfitting.

Because these models are pre-trained based on different large-scale data set, it is hard to do apple-to-apple comparison. But if we have to explain it from the model structure and training method perspective, the essential difference of BERT from GPT is that BERT uses bi-directional information and uses a continuous bag of words (CBOW) way to predict '[MASK]' from context, while GPT is standard left-to-right language modeling which predicts current word from previous words. Possibly, the GPT way can memorize training data very well given input is short, but when new next words appear in test set, it does not generalize so well. Note here only last word's embedding in GPT is used as features for classification. The difference between BERT and XLNet is that XLNet does not introduce the special '[MASK]' token into training data, instead, it predict a randomly selected word in an autoregressive way where a set of words are selected from whole context window as observed sequence. The initial motivation is that it solves the mismatch issue of BERT between training data and downstream fine tuning data. One weak conjecture here is that '[MASK]' might play a role as dropout and regularize the model. Thus BERT is more robust than XLNet on this Q2I data set.

At last, we plotted the training or fine tuning time of the DNN models in figure 8. Note that the time before BERT is by Intel Xeon(R) CPU E5-2686 v4 with 2.30GHz, while the time of BERT and after is by Nvidia Tesla V100 GPU. We use CPU purposely whenever it is possible in case there is no GPU device at a different inference time. Usually a typical DNN model like CNN / RNN finishes training in 10 minutes on a single epoch. But BERT fine tuning takes 2.5+ hours with a much more powerful GPU.

Table 2: P@1 of annotation versus raw text.

	RawQuery	+Annotations	+RawText
P@1	76.78%	91.04% (**+14.3**%)	76.33% (**-0.45**%)

Table 3: P@1 of different annotation numbers.

AntnNum	1	5	10	15	20
P@1	87.12%	91.04%	92.11%	92.15%	92.23%

Table 4: P@1 of different pin numbers.

PinNum	1	2	3	4	5
P@1	91.04%	91.98%	91.68%	91.71%	91.84%

4.3 Impact of Augmentation Details

We evaluate different augmentation setups in this section for best online performance based on the FastText model.

Annotation Versus Raw Text. The first thing we hope to verify is whether the annotation algorithm in section 3.4 is necessary or not. We compared the performance of the annotation and pin's title plus description-based augmentation in table 2. It is pretty clear that annotation-based augmentation is much better than raw text-based augmentation. More than that, title + description is not helping Q2I at all, it is even slightly worse than just raw query. This is probably because that such plain text information is noisy. Therefore, we are confident about the necessity of the auxiliary annotation algorithm.

Different number of annotations. We enumerated the number of annotations of the top engaged pin with step size 5 in table 3. We observe that 5+ annotations are significantly better than just top 1 annotation. It is also clear that the gain of augmentation is marginal after 5 annotation terms. It's good enough to just use 5 because inference time increases linearly with the length of input text.

Different number of top engaged pins. We examine the top 5 engaged pins with 5 annotation terms per pin in table 4. The best performance is achieved with top 2 pins. Increasing more pins is not helping Q2I. On average, each pin has more than 10 interests. It is reasonable that two pins can cover most of the query's possible interests.

In general, the more augmented terms the more complexity of offline workflow and online inference. There is always a tradeoff between classification precision and system latency. We choose top engaged pin with top 5 annotations using FastText as model in production.

4.4 Possible Q2I Applications

With the 90+% precision of Q2I, we are confident to nominate some important applications that have been successfully deployed in our systems. We hope it may inspire other practitioners to mine queries' interests in their own applications.

Interest-based ads targeting. We deliver ads (a.k.a promoted pins) to pinners to satisfy their interests. We refer audience to the official description of interest targeting on Pinterest's page [2]. In particular, pinner's search queries are important building blocks for profiling their interests. It is very important to guarantee pinners' smooth experience when they search or browse something in home feeds by showing ads with the same interests. Otherwise, it can be distracting and hurt pinners' value, which hurts Pinterest's value in long run. A good Q2I model we built in previous sections is for this purpose.

Search relevance feature. Due to the fact that pin's text data can be sparse, the search queries of Pinterest do not necessarily match backend index well as in a typical text-based engine like Google or Bing. Therefore, mapping both query to a higher level of concept like interest would increase retrieval recall. More importantly, by matching queries' interests to pins' interests can serve as a good discriminative feature to improve relevance model.

Guided search term recommendation. It is not uncommon that a user is not sure of how to compose search query properly, especially when she is not sure of the final results she expects. Guided search term is a feature that provides query recommendation to user in real-time. With Q2I, we can use 1) predicted interest; or 2) top or trending queries of the same predicted interests as recommendations.

Shopping intent detection. Pinners often use Pinterest for exploring products they may purchase. If a query is classified to some top tier interests like "Woman's Fashion" and "Home Decor", it probably means the pinner is in shopping mode. Therefore, we should retrieve more product pins and boost them in results. Q2I's predictions are very useful components for detecting the shopping intent.

5 CONCLUSIONS AND FUTURE DIRECTIONS

In this paper, we solve the Q2I classification problems via a novel data augmentation approach. We mine the interests encoded in the top pin of the query's search results and append them to the input query. The average number of words has increased to 13.6 from 3.3, which essentially improves precision by 10+%. To ensure a fair comparison, we have extensively studied state-of-the-art deep neural network models on Q2I and conclude that FastText still rivals best DNN solution with much simpler model structure and potentially much shorter training and inference time. In the future, there are two directions we will explore further:

Study more data augmentation methods, for example, build query to query graph based on the common pins in their search results. In this way, we can propagate information between queries to help queries with little information. Another motivation is that pre-trained NLP models are best at natural language data. Probably feeding such information into them can improve model performance significantly. One candidate solution is using image captioning techniques to generate natural sentences from pin's visual content.

The other clear possible improvement is to use ensemble models. The DNN models we studied here are based on different architectures. Usually the combination of different models in an ensemble can compensate for each model's shortcomings. So either simple majority voting or a supervised layer on top of base DNN model predictions should beat a single best model.

ACKNOWLEDGMENTS

We thank Jennifer Zhao for the discussion and proof read of this paper, ads quality team for the fruitful collaboration, and Vijay Narayanan for his technique leadership on the Q2I project.

[2] https://business.pinterest.com/en/pinterest-ad-targeting-capabilities

REFERENCES

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

[2] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder. *CoRR* abs/1803.11175 (2018). arXiv:1803.11175 http://arxiv.org/abs/1803.11175

[3] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 785–794. https://doi.org/10.1145/2939672.2939785

[4] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. 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*. 1724–1734.

[5] Corinna Cortes and Vladimir Vapnik. 1995. Support-Vector Networks. *Machine Learning* 20, 3 (1995), 273–297. https://doi.org/10.1007/BF00994018

[6] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. 2978–2988.

[7] 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

[8] Venkatesh Ganti, Arnd Christian König, and Xiao Li. 2010. Precomputing search features for fast and accurate query classification. In *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*. 61–70. https://doi.org/10.1145/1718487.1718496

[9] Rafael S. Gonçalves, Matthew Horridge, Rui Li, Yu Liu, Mark A. Musen, Csongor I. Nyulas, Evelyn Obamos, Dhananjay Shrouty, and David Temple. 2019. Use of OWL and Semantic Web Technologies at Pinterest. In *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*. 418–435.

[10] Edouard Grave, Tomas Mikolov, Armand Joulin, and Piotr Bojanowski. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*. 427–431.

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

[12] Bernard J. Jansen, Danielle L. Booth, and Amanda Spink. 2008. Determining the informational, navigational, and transactional intent of Web queries. *Inf. Process. Manage.* 44, 3 (2008), 1251–1266.

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

[14] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. http://arxiv.org/abs/1412.6980

[15] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*. 1378–1387.

[16] Michal Laclavik, Marek Ciglan, Sam Steingold, Martin Seleng, Alex Dorman, and Stefan Dlugolinsky. 2015. Search Query Categorization at Scale. In *Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015, Florence, Italy, May 18-22, 2015 - Companion Volume*. 1281–1286.

[17] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent Convolutional Neural Networks for Text Classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. 2267–2273.

[18] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *CoRR* abs/1909.11942 (2019). arXiv:1909.11942

[19] Eileen Li. 2019. Pin2Interest: A scalable system for content classification. https://medium.com/pinterest-engineering/pin2interest-a-scalable-system-for-content-classification-41a586675ee7.

[20] Tie-Yan Liu. 2011. *Learning to Rank for Information Retrieval*. Springer. https://doi.org/10.1007/978-3-642-14267-3

[21] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-Task Deep Neural Networks for Natural Language Understanding. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. 4487–4496.

[22] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019). arXiv:1907.11692 http://arxiv.org/abs/1907.11692

[23] 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

[24] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab K. Ward. 2016. Deep Sentence Embedding Using Long Short-Term Memory Networks: Analysis and Application to Information Retrieval. *IEEE/ACM Trans. Audio, Speech and Language Processing* 24, 4 (2016), 694–707.

[25] 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.

[26] Alec Radford and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training. In *arxiv*.

[27] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language Models are Unsupervised Multitask Learners. (2018).

[28] 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.

[29] Victor Sanh, Lysandre Debut, and Thomas Wolf. 2019. Smaller, faster, cheaper, lighter: Introducing DistilBERT, a distilled version of BERT. https://medium.com/huggingface/distilbert-8cf3380435b5.

[30] Holger Schwenk, Loïc Barrault, Alexis Conneau, and Yann LeCun. 2017. Very Deep Convolutional Networks for Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*. 1107–1116.

[31] Bharath Sriram, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. 2010. Short text classification in twitter to improve information filtering. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, 2010*. 841–842.

[32] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to Fine-Tune BERT for Text Classification? *CoRR* abs/1905.05583 (2019). arXiv:1905.05583 http://arxiv.org/abs/1905.05583

[33] 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.

[34] Heath Vinicombe. 2019. Understanding Pins through keyword extraction. https://medium.com/pinterest-engineering/understanding-pins-through-keyword-extraction-40cf94214c18.

[35] Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. 2017. Combining Knowledge with Deep Convolutional Neural Networks for Short Text Classification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. 2915–2921.

[36] Zhongyuan Wang, Kejun Zhao, Haixun Wang, Xiaofeng Meng, and Ji-Rong Wen. 2015. Query Understanding through Knowledge-Based Conceptualization. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. 3264–3270.

[37] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *CoRR* abs/1906.08237 (2019).

[38] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical Attention Networks for Document Classification. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*. 1480–1489.

[39] Andrew Zhai, Hao-Yu Wu, Eric Tzeng, Dong Huk Park, and Charles Rosenberg. 2019. Learning a Unified Embedding for Visual Search at Pinterest. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*. 2412–2420. https://doi.org/10.1145/3292500.3330739

[40] Ye Zhang and Byron C. Wallace. 2017. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017 - Volume 1: Long Papers*. 253–263.

[41] Jinfeng Zhuang and Yu Liu. 2019. PinText: A Multitask Text Embedding System in Pinterest. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*. 2653–2661. https://doi.org/10.1145/3292500.3330671