Selection and Interpretation of Embedding Subspace for Query Classification

Kyoung-Rok Jang
Sung-Hyon Myaeng
kyoungrok.jang@kaist.ac.kr
myaeng@kaist.ac.kr
KAIST

Hee-Cheol Seo
Joohee Park
heecheol.seo@navercorp.com
james.joohee.park@navercorp.com
Naver

ABSTRACT

Due to the difficulty of understanding and deciphering the content of text embeddings, the embeddings are often used as features as a whole for classification. In this paper, we show this is not always desirable for tasks like query classification and propose a method for identifying a *subspace* — a list of vector elements — of query embeddings, which strongly corresponds to a particular class. We hypothesize that an individual or a group of embedding elements can serve as useful conceptual features. Also proposed is a method for interpreting a subspace as a list of words that characterizes the subspace. To make subspaces more conceptually distinct, we test our approach with high dimensional sparse representations of query embeddings, which can be generated by applying a sparse coding method to ordinary dense embeddings. Our proposed method is expected to help understanding how classification models process embeddings and subsequently provide guidance to make classifiers more accurate and efficient.

CCS CONCEPTS

• **Information systems → Query representation**; **Query intent**; • **Human-centered computing → Empirical studies in visualization**;

KEYWORDS

query classification, text embeddings, interpretation, sparse representations

ACM Reference format:
Kyoung-Rok Jang, Sung-Hyon Myaeng, Hee-Cheol Seo, and Joohee Park. 2019. Selection and Interpretation of Embedding Subspace for Query Classification. In *Proceedings of SIGIR 2019 Workshop on ExplainAble Recommendation and Search, Paris, France, July 25, 2019 (EARS'19)*, 9 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

It is common practice to use word or text embeddings as input for classification tasks (e.g. sentiment analysis [30], document classification [52]), since embeddings are useful for capturing the meaning of a word and text. Unfortunately, it is also common to blindly use the whole embeddings as input to classifiers although some part of embeddings may not be useful for a given task. It is because the content of embeddings is hard to interpret, making it challenging to select which part of embeddings to use.

In this paper, we propose a method of identifying and interpreting a *subspace* — a list of vector elements (i.e., dimensions) — of query embeddings that play a critical role in query classification. We will use the terms 'elements' and 'dimensions' interchangeably. Query classification is a problem of assigning a query one or more predefined classes based on the query's topic or intention. For instance, for a query "birthday gift", a classifier needs to infer that it intends to buy a gift, and classify it to the `Shopping` category.

Previous studies showed that an embedding's elements loosely represent distinct 'concepts.' For instance, the authors of [33] proposed a model to bridge the gap between distributional semantics (vector space) and conceptual space. The model dynamically selects specific dimensions associated with seed terms, which form a subspace of terms defining the related concept. Inspired by this, we hypothesize that embedding dimensions, individually or as a group, can serve as unique and meaningful *conceptual* features for classification tasks. We expect heterogeneous query classes (e.g. `News` vs. `Shopping`) would have distinct subspaces that characterize each of them. Identifying such subspaces would provide more in-depth insight on how the underlying classification model understands and processes embeddings.

We argue that identifying class-specific subspaces is essential to classify queries accurately. Because a user's intention is not explicitly stated in a query which usually comprises a few keywords, it would be critical to identify a subspace for common concepts or features that represent the queries belonging to the same class. For instance, even if two queries are not so close in their semantic as commonly reflected in the embeddings, they are expected to be classified into the same class when they share the same intent. For instance, the queries "parent's day gift" and "bleach" should be classified as `Shopping` class. If embeddings are used as features as a whole, it would be challenging for a classifier to select a small set of intent-oriented elements and decide both queries belong to `Shopping`. While the *Attention* techniques [5] can address this issue, our method of not only identifying but also interpreting subspaces have an added advantage of making the classification process more explicit and explainable; more details be explained in later sections.

To select and describe subspaces, we first encode queries into embeddings by using a DisC model [2], which is a simple and fast *Bag-of-n-gram* embedding model that generates text embeddings by combining embeddings of n-grams. We then train a classifier that takes query embeddings as input, treating embedding elements as

EARS'19, July 25, 2019, Paris, France

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

features. We use Gradient Boosting Tree as the classification model. Next, we use a SHAP (SHapley Additive exPlanation) model [29] to identify the subspace that is strongly connected to a specific query class. Finally, we automatically provide a verbal description of the identified subspace for human interpretation.

To generate more precise and conceptually consistent interpretation result, we test our method with high-dimensional sparse representations of queries, motivated by earlier works that discovered ordinary dense embeddings can be decomposed to have more *conceptually coherent* dimensions by applying *sparse coding* methods [3, 37, 51]. Compared to lower dimensional dense embeddings, sparse embeddings have the following advantages: 1) They can express a wider range of conceptual features gathered from a text corpus, and 2) Each word is described by a small number of significant features [37], which coincides with the findings from feature-norming experiment that showed only 10–20 characteristics are usually enough to describe a given concrete concept [22]. We expect these properties of sparse representations can help generate more accurate and fine-grained subspace selection and interpretation results, which we will show through our experiment.

Although our application in this paper is query classification, our method can be applied to other kinds of classification tasks that take embeddings as features, provided that there is an applicable *feature attribution* method for the classifier being used.

The main contributions of this paper can be summarized as follow:

- We show that using ordinary word embeddings as a whole for a query classification task is problematic, i.e. not sufficiently effective.
- We present our method of identifying, selecting, and interpreting embedding subspaces that are important for improved and transparent query classification..
- We show that by using our method with *sparse* query embeddings we obtain clearer interpretation of important subspaces.

2 RELATED WORK

Query Classification. Query classification has been studied quite extensively [10]. With KDDCUP 2005 competition that sparked interests on this research, various methods are proposed ranging from click-through data [7, 53] to deep neural network models [26]. Also, many different taxonomies for query classification have been introduced from the perspective of topics or user intents (e.g. [6, 11]).

Word Embeddings and Bag of n-grams Embeddings. Word embedding models [9, 41] are one of mainstream research topics in neural network based text processing and became an indispensable tool for NLP tasks after the seminal paper published by Mikolov [34], which is based on the idea proposed in [8]. The basic idea is to optimize the embeddings of words to be similar if the words share similar contexts. The idea borrows the theoretical basis from distributional semantics: "a word is characterized by the company it keeps." [23].

The bag of n-grams model, specifically Distributed Co-occurrence (DisC) model [2, 52] encodes text as linear combination of n-gram embeddings. Although the model doesn't encode the full sequence information of text as the LSTM does, it does consider the order of words to some degree by considering n-grams. This approach turned out to be a fast and effective method for various classification tasks [2].

Model Interpretability. Interpretability can be defined as "the degree to which a human can understand the cause of a decision" [35]. Making machine learning models interpretable help understand *why* the models succeed or fail and could give us better intuition about the problem and higher trust in the solution. More fundamental need for the interpretability stems from an *incompleteness* in the problem formalization [19].

The easiest way to attain interpretability is to use the simple models (e.g. linear regression) that are inherently interpretable, thanks to their simple structure (e.g. linear regression, decision tree), at the cost of performance on complex problems. For a model with many abstraction layers (e.g. deep neural networks, ensemble models), a possible way to interpret the result is to train an interpretable model that can *approximate* the prediction of the model (i.e. surrogate models). LIME [44] and SHAP [29] models are recent developments of such local surrogate models. It is called *local* because it aims to explain individual predictions, not the global behavior of the target model. Especially, "SHAP connects game theory with local explanations, uniting several previous methods [4, 13, 16, 27, 44, 47, 48] and representing the only possible consistent and locally accurate additive feature attribution methods based on expectations" [29].

Sparse Embeddings. Sparse representations are a class of representations motivated from the highly sparse representations observed in a human brain [24, 40]. Under the sparse representations each sample is characterized by only a few important features out of a large number of available features (i.e. sparse and high-dimensional), making the samples robust to noise and interference from random input [1].

Applying sparse representations on word and text embeddings is not a new idea. Largely there are two ways to make the word embeddings sparse: post-processing pre-trained dense embeddings using techniques such as *sparse coding* [20, 37], or directly learning the sparse representation of words by imposing sparsity constraints to the learning objective [15, 50]. The benefits of high dimensional sparse embeddings are that each dimension is conceptually more coherent relative to that of dense embeddings and that only a few representative concepts of each word are preserved. We expect these properties of sparse embeddings will help generate a more accurate and precise interpretation of model prediction.

3 TASK

3.1 Goal

Our goal is to classify the Korean queries to query 'collections' (e.g. `News`, `Shopping`) defined by Naver[1], the Korean web search engine run by NHN corporation. In Naver search engine, the indexed documents are categorized into 48 collections (e.g. `Image`, `Blog`). Our task is to predict a collection relevant to a given query's intention. For instance, if a user submits the query *"spring pretty wallpaper"*, the system should predict the query's collection as `Image`, search

[1] https://naver.com

for the relevant resources from Image collection, then finally return the result with *grid view* of images, which is an optimized view for Image collection. Since the relevancy of a predicted query's collection will greatly affect the user experience, classifying query collections accurately is an important task.

3.2 Problem and Challenge

In Naver, the query classification is performed based on click-through data. The problem of this approach is that it cannot handle new, unseen queries that have no historical data. It's similar to 'cold start' problem in recommender systems. Since up to 80% of daily queries is "unseen", they become a severe obstacle to the query classification system (e.g. Nvidia GTX1080ti → Review, Nvidia RTX2080 → ?).

The embedding-based approach can be a promising direction to tackle the problem, especially compared to the keyword-based approach, which is susceptible to unseen keywords. By sharing the click-through histories between semantically similar queries, we could compensate for the missing histories of unseen queries (e.g. RTX2080 ~ GTX1080ti, Nvidia RTX2080 → Review).

But even with embeddings, finding class-specific characteristics between queries is not an easy task. It is because queries are short, suggestive, and diverse with abundant variations such as model numbers. For instance, there may be cases when the semantic match between two queries is low but should be classified as the same collection (e.g. "Gang-nam delicious restaurants", "Sony RX0m2" → Review). There may be other cases when the semantic match between two queries is high but should be classified as different collections (e.g. "Jeju island flight" → Trip, "Jeju island flight crash" → News). The point to stress is that using embeddings as a whole is not always effective for classifying this type of queries.

Instead, we need a method to identify the subspace (a list of embedding elements) of query embeddings that signifies each collection, which would help the classification achieve higher accuracy (Figure 1). This approach is plausible because there is evidence that the elements of textual embeddings represent specific senses or concepts that compose text while exhibiting a range of *specificity* depending on whether the embeddings are dense or sparse [21, 33, 37, 50, 51].

We argue that with the knowledge of important subspaces, we can predict a query's collection more accurately and efficiently, similar to the benefits we can get by performing *feature selection*. Also, with the help of subspace interpretation technique proposed in this paper, we can elucidate how the classification model understands and utilizes the elements of embeddings, enabling us to perform interpretable model debugging process. It is why we are proposing a method of selecting and interpreting important subspace for query classification, which we will explain in Section 4 and 4.4.

4 METHOD

4.1 Query Embedding Generation

To convert queries to embeddings we used Distributed Co-occurrence (DisC) [2] model. DisC model is a sentence embedding model inspired by a Bag of *n*-gram (BonG) representation of a document, each element of which represents all possible *n*-grams collected from the texts being represented. The DisC model considers the order of words to some degree by modeling *n*-grams and is reported to be a strong baseline for a variety of text classification tasks [2]. LSTM-based models (e.g. [25, 42]) or Transformer-based models (e.g. [18, 43]) are currently more popular and powerful options for modeling short and long texts. But we decided to use DisC model because of the following reasons:

Figure 1: Classifying the query "high-res wallpaper" by inspecting the strength of a subspace that is connected to Image collection. Identifying the important embedding subspace for query classification can be helpful to make the classification more accurate

- **Complexity**: A search query usually comprises few words, but the exact word order of the entire query is not always critical. Applying the models like BERT [18] which consider complex relationships between tokens and sentences is likely to be an overkill.
- **Efficiency**: The DisC model requires just linear operations of pre-trained word embeddings. It is more suitable to be applied to systems which require high throughput and responsiveness, such as search engines.
- **Ease of Interpretability**: Due to the model's simple architecture, the representations generated by the DisC model are easier to interpret. The interpretation method will be explained in Section 4.4.

DisC embeddings are generated by concatenating the sum of *n*-gram embeddings gathered from a target text (Definition 4.1 and 4.2).

Definition 4.1 (Compositional n-gram embedding). Represent *n*-gram $g = (w_1, \ldots, w_n)$ as the element-wise product $v_g = v_{w_1} \odot \cdots \odot v_{w_n}$ of the embeddings of its constituent words. Note that DisC embeddings require pre-trained word embeddings (e.g. fastText [9]) to generate *n*-gram embeddings.

Definition 4.2 (DisC embedding). The DisC embedding of a piece of text is a concatenation for $(v_1, v_2, \ldots, v_n)$ where v_n is the sum of the *n*-gram embeddings of all *n*-grams in the document (for $n = 1$ this is just the sum of word embeddings).

The number of dimensions of the resulting embeddings is determined by choice of *n*. For instance, if we decide to use up to 2-grams with 300-dimensional word embeddings, the number of dimensions will be 600, in which 1–300th dimensions for the sum of 1-gram embeddings and 301–600th dimensions for the sum of 2-grams embeddings. Figure 2 shows an example of the composition process of DisC embeddings.

4.2 Query Classifier

We use a Gradient Boosting algorithm for the classification task, where a classification model is developed by stacking weak predictors (e.g. shallow decision trees) sequentially. More specifically we

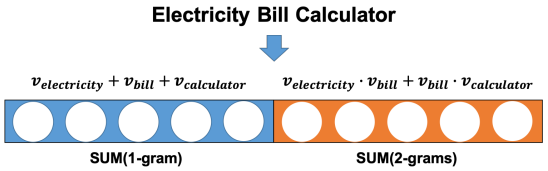

Figure 2: This figure shows an example of how a query embedding is constructed using the DisC model. When up to 2-grams are used, the resulting embedding becomes 600 dimensional, which is twice as big as the original word embeddings. Note that 2-gram embeddings are constructed with an element-wise product of the constituent word embeddings.

use `xgboost` which is an optimized distributed gradient boosting library [14]. Gradient Boosting is known to be especially powerful for classification tasks. We opt for Gradient Boosting because of its high performance, as it is essential to start with a highly accurate classification model to select and interpret subspaces that dictate the classification result.

4.3 Subspace Selection

Selecting a subspace is the same as identifying a list of embedding elements that play an essential role in deciding the class. We can utilize feature attribution methods for this. Among many feature attribution methods, we choose the SHAP (SHapley Additive exPlanations) model [29], because it is the only possible consistent and locally accurate feature attribution method known to us.

The SHAP model is unique in that it is a unified method to interpret prediction models (i.e. query classification model in our case) by merging a variety of interpretation methods that belong to *additive feature attribution* class, including [17, 27, 44, 49]. It implements a single unique solution in this kind of interpretation models that satisfies three desirable properties of explanation model: *local accuracy*, *missingness*, and *consistency*, all of which can be achieved by calculating Shapely values [27] for each feature w.r.t. the model's prediction.

A Shapely value computes an importance score of a feature by measuring the effect on the model's prediction when that feature is included (Equation 1).

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S) \right] \quad (1)$$

Where F is the set of all the features, $S \subseteq F$ all feature subsets, $f_{S \cup \{i\}}$ a model trained with the feature i included, and f_S a model trained without the feature. With $f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S)$ we compare the output of the two models on the current input to see the contribution of the feature. The computation is done over all the query embeddings belonging to a particular class so that we can identify the salient elements of the embeddings.

4.4 Subspace Interpretation

To interpret a subspace, we try to provide visual clues of the semantics of subspace using examples (currently words). The idea is to search for the words that have high values in the elements that belong to a subspace. It's a common technique utilized in the *feature visualization* field, which is reviewed in [12, 39]. Unlike LSTM-based sentence embedding models that produce a *composite* embedding, the query embeddings generated by DisC make it easy to discern the critical embedding elements associated with the constituent words and word sequences of varying lengths. Naturally, a list of top k words representing an element can be seen as tied to k-grams rather than the entire query or a sentence.

The method of interpreting a single embedding dimension by searching the nearest k words is already presented in previous studies [3, 20, 37]. Our approach is slightly different in that we average the values of all the dimensions that constitute target subspace (Equation 2). It is because we want to provide an interpretation of a subspace, not a single dimension. We found that this simple method is effective for understanding the semantics of a subspace and that the interpretation matches the human intuition. The subspace score of a candidate word is computed as follows:

$$SubspaceScore(w) = \frac{\sum_{i \in S} d_i}{|S|} \quad (2)$$

where S is the dimension indices of the target subspace, $|S|$ is the size of S, and d_i represents the value of i-th dimension of a word w.

The list of words that have high subspace score can be considered as the interpretation of that subspace. The same word will have multiple subspace scores because the subspace for different query class usually comprised a different set of embedding elements. Figure 3 shows a query embedding with the subspace for the class `SHOPPING` circled and the interpretation for the subspace written on the right.

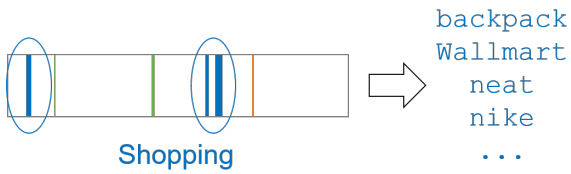

Figure 3: An example of subspace interpretation for `Shopping`

4.5 Comparison between Sparse vs. Dense Embeddings

We test our methods with two types of embeddings — dense embeddings and sparse embeddings. Dense embeddings refer to those derived from the well-known methods like `word2vec` or `fastText`. Those are usually low-dimensional vectors (e.g. 300), and most of the dimensions are filled with non-zero values. In contrast, sparse embeddings are high-dimensional (e.g. 2,000) vectors, and most of the dimensions are zeros.

We expect sparse embeddings to be suitable for selecting and interpreting subspaces more precisely because of the following properties:

(1) **Sparse**: Each sample is represented using only a few important dimensions.

(2) **High-dimensional**: The dimensions represent more specific concepts.

This means sparse embeddings may contain less noise and have more conceptually consistent (i.e. less ambiguous) dimensions, which are desirable properties for our task. For a more detailed explanation and analysis of sparse representations, please refer to [1, 37].

To generate sparse *query* embeddings, we first decompose dense word embeddings into sparse *word* embeddings using Winner-Take-All autoencoder [32], then take the same steps described in Section 4 and 4.4. Winner-Take-All autoencoder is a slight modification of k-sparse autoencoder [31], which is the autoencoder that preserves only k largest activations of a hidden layer.

k-Sparse Autoencoders [31]
1) Perform the feedforward phase and compute $\boldsymbol{z} = W^\top \boldsymbol{x} + \boldsymbol{b}$
2) Find the k largest activations of $\boldsymbol{z}$ and set the rest to zero. $\boldsymbol{z}_{(\Gamma)^c} = 0 \quad \text{where} \quad \Gamma = \text{supp}_k(\boldsymbol{z})$
3) Compute the output and the error using the sparsified $\boldsymbol{z}$. $\hat{\boldsymbol{x}} = W\boldsymbol{z} + \boldsymbol{b}'$ $E = \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2$
3) Backpropagate the error through the k largest activations defined by Γ and iterate.

5 EXPERIMENT

5.1 Dataset

Our experiment is based on a classification dataset provided by Naver, which amounts to a total of about 900 million queries belonging to 40 collections, i.e. classes. Since the distribution of collections is skewed, we decided to use only the largest seven collections for our experiment. Table 1 shows their statistic. The queries we use for our experiment takes over 70% of the whole dataset, so can be regarded as the representatives. We randomly sampled about 200,000 queries per collection to result in about 1,400,000 queries in total. The ratio of training, development, and test sets is 8:1:1. Note that the queries are in Korean.

Table 1: Dataset statistic (before sampling)

Collection	Count	Ratio (%)
News	140,742,157	15.77
Image	116,241,259	13.02
Blog	105,939,264	11.87
Review	95,662,634	10.72
Website	68,152,695	7.64
Shopping	56,322,435	6.31
Local	49,990,447	5.60
Total	**633,050,891**	**70.93**

The samples comprise two fields: 'collection' and 'query.' A query's collection is recorded based on user clicks; if a user submits a query then clicks a document that belongs to a specific collection, that becomes the query's collection. Since users may click documents from different collections after issuing the same query, the same query can point to multiple collections, generating separate samples. We regard all such cases as correct labels.

5.2 Process

The overall procedure is depicted in Figure 4. We explain each step in detail below.

Figure 4: Experiment Process

5.3 Query Segmentation

Korean does not strictly require word spacing, and users have freedom, especially for constructing noun phrases. But since the DisC model we use to generate query embeddings is built upon word embeddings (Section 4.1), queries should be separated into words or morphemes (Figure 5). We did this segmentation using a proprietary Korean morphological analyzer developed by Naver[2].

Figure 5: An example of Korean query segmentation (translated into English)

5.4 Query Embedding Generation

We used `fastText` model to generate word embeddings. We constructed the training corpus by merging Korean Wikipedia dump (2019-01-23) and the queries provided by Naver. The corpus contains about 10 million lines of Korean text and is preprocessed using the same morphological analyzer used for query segmentation. We used `gensim`[3] library to generate `fastText` embeddings using the default parameters provided by the library.

DisC embeddings of the queries are then generated based on the word embeddings. We used up to only 2-grams of queries to generate DisC embeddings because a large proportion of queries are shorter than three words. The resulting query embeddings have 600 dimensions: 1–300 dimensions represent 1-gram space, and 301–600 dimensions represent 2-gram space.

[2]There are some free Korean morphological analyzers available, e.g. https://bitbucket.org/eunjeon/mecab-ko/src/master/
[3]https://radimrehurek.com/gensim/

5.5 Sparsify Query Embeddings

We compared the result between the dense and sparse query embeddings, as mentioned in Section 4.5. To generate sparse query embeddings, we first decompose the dense word embeddings to sparse embeddings. For that, we use Winner-Take-All Autoencoder with a batch size of 64, the target dimension of 2,000, allowing only N% of dimensions to have non-zero values. The Ns we tested is presented in Section 6.

Then we generated DisC embeddings based on the sparse embeddings using the same settings we used with the dense embeddings. So the resulting sparse query embeddings have 4,000 dimensions, 1–2,000 dimensions for 1-grams and 2,001–4,000 dimensions for 2-grams.

5.6 Classifier Hyperparameter Tuning and Training

We used `xgboost` to train the classifier. The library requires hyperparameters (e.g. `max_depth`, `gamma`) to be determined before start training. We used the dev set for finding optimal hyperparameters.

To efficiently search the best hyperparameters, we utilized Bayesian Optimization (please refer to [46]). Bayesian Optimization works by approximating the behavior of unknown objective functions using Bayesian learning or Gaussian process, then try to search the most *fruitful* hyperparameter space. We used `bayesian-optimization`[4] library in our experiment. Using the found parameters, we trained the classifier with the train set.

5.7 Subspace Selection and Interpretation

To identify subspaces, we used SHAP library[5] released by the authors of [29]. Specifically, we used `TreeExplainer` module [28] which is designed to be used with Gradient Boosting Tree models like `xgboost`. The SHAP model requires not only a trained model to be analyzed but also additional samples to calculate the feature attribution scores. We utilized the test set as the additional samples. After the calculation, each dimension has seven scores for seven collections, respectively. We determined the subspace for each collection by selecting top-k dimensions per collection.

We then generated the interpretation of each subspace using the method described in Section 4.4. The interpretation result is different depending on the choice of k, which will be shown in the next section.

6 RESULT

6.1 Classification Performance

Although the main impetus of this paper is not to show how the best performing query classifier can be constructed, examining the classification performance of the different versions of the classifier can help to have confidence on our method of subspace selection and interpretation. In particular, it is worth comparing the performance of dense embeddings and sparse embeddings with different *sparsity* (i.e. the ratio of non-zero dimensions) settings. Note that the hyperparameters for different embedding types are automatically searched as described in Section 5.6 and might be different depending on embedding types.

Table 2 shows the overall classification performance of the embedding variations. We achieved the best performance with the dense embeddings. With the sparse embeddings, we see decreasing performances as we decrease the sparsity (i.e. the ratio of non-zero). It is expected because in sparse embeddings only a small portion (e.g. 1%) of dimensions are activated, and a decrease in sparsity incurs information loss. Nonetheless, the sparse embeddings seem to be successful in capturing the 'essence' of the original dense embeddings because they show competitive classification performance.

Table 2: Classification performance. The *sparsity* indicates the average ratio of non-zero dimensions in sparse embeddings.

Embeddings	Precision	Recall	f1-score
Dense	**0.649**	**0.653**	**0.644**
Sparse, sparsity = 5%	0.644	0.651	0.639
Sparse, sparsity = 2%	0.638	0.645	0.633
Sparse, sparsity = 1%	0.635	0.643	0.630
Sparse, sparsity = 0.5%	0.630	0.638	0.625

We also show confusion matrix of two classifiers: Dense and Sparse (sparsity = 5%). Note that the classification accuracy is high with `Website` and `Shopping` but low with `Blog` and `Image` (Figure 6). The queries of `Website` and `Shopping` are expected to show relatively high semantic & symbolic regularities, whereas the queries of `Blog` and `Image` can be talking about anything. We conjecture this incurred the gap in the classification performance. We show the query samples in Table 3 to give the sense of conceptual variation in queries that belong to such collections.

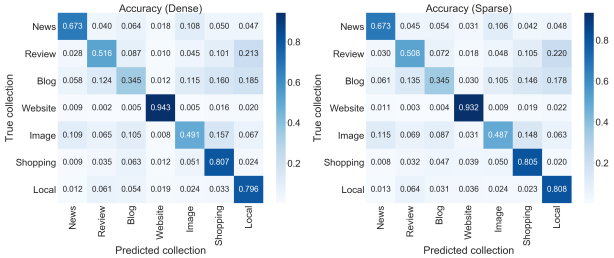

Figure 6: Confusion matrix of Dense and Sparse (5%)

6.2 Subspace Selection

We present the identified subspace for each collection, which is a list of top k important dimensions selected based on SHAP values. For brevity, we only show the top 5 dimensions here, but we can decide on the k according to applications' need. With the dense embeddings the `Blog` and `Local` shares the 260th dimension as the

[4] https://github.com/fmfn/BayesianOptimization
[5] https://github.com/slundberg/shap

Table 3: Query samples. The keywords are translated from Korean and joined by underscore for readability. Classifying `Website` and `Shopping` was highly accurate, whereas classifying `Blog` and `Image` wasn't (Figure 6).

Collection	Queries
`Website`	linkedin, naver, youtube, uk_google, facebook
`Shopping`	nike_polo, apple_watch_stand, lacoste_sling_bag, maserati_headlight_price
`Blog`	relative_pitch_test, backpack_for_trip, Gangnam_restaurants, low-capital_cafe, cellulite
`Image`	tom_cruise, batchroom_tongs, adidas_exin, men_short_haircut, strawberry_cake

most prominent dimension, implying that there is some conceptual overlap between two collections. It is also noticeable that the most of the salient dimensions belong to 1-gram space (dense: 1–300th, sparse: 1–2000th) of query embeddings, implying that the order of keywords in queries is usually less important (Table 4).

Table 4: The result of subspace selection. The numbers represent the index of dimensions ordered by their SHAP values in reverse order.

Collection	Embeddings	
	`Dense`	**`Sparse` (5%)**
`News`	228, 300, 120, 45, 253	890, 1604, 1272, 400, 1207
`Review`	11, 70, 178, 238, 28	1124, *3054*, 1207, 593, 71
`Blog`	**260**, 275, 230, 172, 28	278, *2053*, *3568*, 1371, *3127*
`Website`	3, 101, 203, 100, 84	1331, 1120, 1617, 278, 1124
`Image`	130, 42, 123, 200, 18	400, 96, 466, 1207, 264
`Shopping`	186, 240, 18, 124, 242	1749, 890, 1170, 747, 1207
`Local`	**260**, 136, 299, 134, 79	687, *2867*, 1590, 918, 890

The dimensions that constitute subspaces are usually different depending on a target collection. This couldn't happen if a few most informative dimensions are always important for the classification, confirming that our hypothesis — embedding dimensions can serve as unique and meaningful features — is right (Table 4 and 5).

We next show how the SHAP values decrease as the ranks of dimensions decrease. For readability we only present the result of two collections, `Website` and `Blog`, in which the classifier showed the highest and the lowest performance respectively (Figure 7).

`Dense` *(left)*. The SHAP values decrease for about top 5% dimensions and then much more gently afterward. This suggests that there are a few dimensions that significantly contribute to the classification. The trend is weaker in `Blog`, in which the classifier showed the lowest performance among all the collections. This means the classifier is having trouble to identify a small set of high confidence dimensions.

Table 5: The ratio of unique dimensions among every collection. The dimensions of `Sparse` embeddings tend to stay as *unique* features for each collection. The ratio is calculated as $\frac{length(set(top\ n\%\ dims))}{length(top\ n\%\ dims)}$.

Top n% dimensions	Embeddings	
	`Dense`	**`Sparse` (5%)**
1%	**0.905**	0.871
2%	0.821	**0.839**
3%	0.786	**0.802**
4%	0.762	**0.778**
5%	0.695	**0.743**

`Sparse` *(right)*. The SHAP values change almost identically between the two collections, although they differ widely in terms of classification accuracy. An interesting pattern observed is that after the first drop, there is another drop of SHAP values occurring right after the first 2000 dimensions, which corresponds to the number of dimension for 1-gram. It indicates that 1-gram embeddings are more important than 2-gram embeddings of queries in determining the query's class.

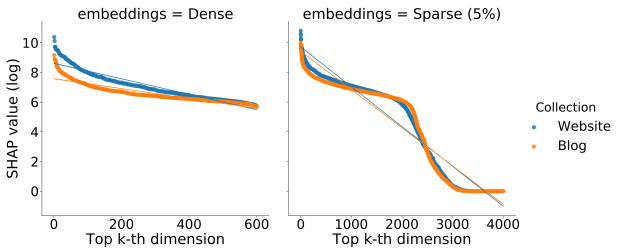

Figure 7: The figure shows the trend of how SHAP values change as the importance of the dimensions decreases. The straight lines indicate the regressions that fit the data points. The y axis (SHAP value) is in log-scale.

6.3 Subspace Interpretation

A subspace interpretation is generated by listing the words whose embeddings have on average high values for the dimensions that constitute the subspace. If such words are deemed to represent the unique characteristics of a query class, it is an indication that the subspace selection and its interpretation is done correctly. We only used top 5% of all the dimensions (top 30 for `Dense`, top 200 for `Sparse`) to calculate the *subspace_score* of words introduced in Equation (2). The interpretation result is shown in Table 6.

The interpretation of the dense embeddings seems inconsistent and noisy, whereas that of sparse embeddings seems conceptually coherent and relevant to the collection. We conjecture that this is because the dimensions of the dense embeddings are conceptually not distinct enough to generate consistent words. In other words,

Table 6: Subspace interpretation. The words are translated from Korean and joined by an underscore for readability. We show only the top 5 words for brevity. The interpretations that seem to conceptually consistent and match with target collection's common concepts are marked as bold.

Collection	Embeddings	
	`Dense`	`Sparse` ($sparsity = 5\%$)
News	loser, bg, dxbg, Hye-ri Jo (athlete), rbr	**control_room, visiting_nurse, deueob, attractions, memorial_altar**
Review	Monrovia, shipboard, flat_cap, firebird, uss_missouri	**coffee_shop, hot_spring, cafe, newsbreak, luxury_seafood**
Blog	shellfish_barrel, down_line, turn_around, turning_around, webtoon_publishing_platform	remaining, waiter, by-election, blood_donation, stew_ingredients
Website	Ahsan, Shetland, ggc, nbgc, rvr	long_wallet, Giheung (area), re-install, incomplete, install
Image	hjho, iho, brow, nescafe, Kobaco (government agency)	Ilsan (area), Nampodong (area), massage, Bucheon (area), Ilsan_maddudong (area)
Shopping	ad_hoc, tide, secure, throb, Symbicort (medicine)	**high_glossy, Daiso (shop), tidy, bag, west_Bundang (area)**
Local	stabilize, emergency_exit, evacuation, stopping, overload	**Bucheon, Gwangan, Ansan, Geomdan, Bupyeong (all areas)**

different concepts are mixed into the same dimension, possibly making them *ambiguous*. It was predicted by the authors of [37], that "the same compact set of features (of low-dimensional embeddings) may not be sufficient to describe all semantic domains of a full adult vocabulary." The result that the dense embeddings have ambiguous dimensions but showed the best classification performance requires further investigation.

To quantitatively evaluate the conceptual consistency (i.e. interpretability) of subspace interpretation, we applied *topic coherence* method. Topic coherence is a method of automatically measuring the interpretability of topic models, which is reported to be closely correlated with human evaluation [36, 38]. We used the topic coherence model introduced in [45]. It is observed that the interpretation generated by `Sparse` generally exhibits higher cohesion scores (Table 7).

Table 7: Topical cohesion of subspace. The higher the better.

Collection	Embeddings	
	Dense	**Sparse (5%)**
News	**-16.100**	-17.906
Review	-14.961	**-12.997**
Blog	-16.929	**-15.104**
Website	-18.453	**-11.202**
Image	-18.870	**-9.310**
Shopping	-19.363	**-15.439**
Local	-17.355	**-13.741**

7 CONCLUSION

In this paper we presented our method of selecting and interpreting embedding subspace for query classification task. We also showed that the sparse embeddings gave a more precise result. Although we validated the proposed method with query classification, our method can be applied to other text classification tasks provided that we have applicable feature attribution methods for classification models. We hope our method help understand and improve neural text classifier's performance, which will be our future work.

ACKNOWLEDGMENTS

This research was supported by the Naver Corp. Any opinions, findings and conclusions expressed in this material do not necessarily reflect the sponsors.

REFERENCES

[1] Subutai Ahmad and Luiz Scheinkman. 2019. How Can We Be So Dense? The Benefits of Using Highly Sparse Representations. (3 2019).
[2] Sanjeev Arora, Mikhail Khodak, and Nikunj Saunshi. 2018. a Compressed Sensing View of Unsupervised. In *International Conference on Learning Representations*. 1–21.
[3] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. Linear Algebraic Structure of Word Senses, with Applications to Polysemy. 6, 1 (1 2016), 483–495.
[4] Sebastian Bach, Alexander Binder, GrÃľgoire Montavon, Frederick Klauschen, Klaus Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE* 10, 7 (2015).
[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. *Neural Machine Translation by Jointly Learning to Align and Translate*. Technical Report.
[6] Steven M. Beitzel, Eric C. Jensen, Abdur Chowdhury, and Ophir Frieder. 2008. Varying approaches to topical web query classification. *Proceedings of the 3rd international conference on Scalable information systems* (2008), 783.
[7] Steven M. Beitzel, Eric C. Jensen, David D. Lewis, Abdur Chowdhury, and Ophir Frieder. 2007. Automatic classification of Web queries using very large unlabeled query logs. *ACM Transactions on Information Systems* 25, 2 (2007), 9–es. https://doi.org/10.1145/1229179.1229183
[8] Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Janvin. 2001. A neural probabilistic language model. *Journal of Machine Learning Research* 3

(2001), 1137–1155.
[9] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. [n. d.]. *Enriching Word Vectors with Subword Information.* Technical Report.
[10] David J. Brenes, Daniel Gayo-Avello, and Kilian Perez-Gonzalez. 2009. Survey and evaluation of query intent detection methods. In *Proceedings of the 2009 workshop on Web Search Click Data - WSCD '09*. ACM Press, New York, New York, USA, 1–7.
[11] Huanhuan Cao, Derek Hao Hu, Dou Shen, Daxin Jiang, Jian-Tao Sun, Enhong Chen, and Qiang Yang. 2009. Context-aware query classification. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval - SIGIR '09*. ACM Press, New York, New York, USA, 3. https://doi.org/10.1145/1571941.1571945
[12] Shan Carter, Zan Armstrong, Ludwig Schubert, Ian Johnson, and Chris Olah. 2019. Activation Atlas. *Distill* 4, 3 (3 2019), e15. https://doi.org/10.23915/distill.00015
[13] A. Charnes, B. Golany, M. Keane, and J. Rousseau. 1988. Extremal Principle Solutions of Games in Characteristic Function Form: Core, Chebychev and Shapley Value Generalizations. Springer, Dordrecht, 123–133. https://doi.org/10.1007/978-94-009-3677-5{_}7
[14] Tianqi Chen and Carlos Guestrin. [n. d.]. *XGBoost: A Scalable Tree Boosting System.* Technical Report. https://github.com/dmlc/xgboost
[15] Yunchuan Chen, Ge Li, and Zhi Jin. 2017. Learning Sparse Overcomplete Word Vectors Without Intermediate Dense Representations. 3–15.
[16] Anupam Datta, Shayak Sen, and Yair Zick. 2016. Algorithmic Transparency via Quantitative Input Influence: Theory and Experiments with Learning Systems. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 598–617. https://doi.org/10.1109/SP.2016.42
[17] Anupam Datta, Shayak Sen, and Yair Zick. 2016. Algorithmic Transparency via Quantitative Input Influence: Theory and Experiments with Learning Systems. In *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 598–617. https://doi.org/10.1109/SP.2016.42
[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (2018).
[19] Finale Doshi-Velez and Been Kim. 2017. Towards A Rigorous Science of Interpretable Machine Learning. (2017).
[20] Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A Smith. [n. d.]. *Sparse Overcomplete Word Vector Representations.* Technical Report. 1491–1500 pages.
[21] Peter Gärdenfors. 2004. *Conceptual spaces: The geometry of thought.* MIT press.
[22] Thomas L Griffiths, Mark Steyvers, and Joshua B Tenenbaum. 2007. Topics in Semantic Representation. (2007). https://doi.org/10.1037/0033-295X.114.2.211
[23] Zellig S Harris. 1954. Distributional Structure. *WORD* 10, 2-3 (1954), 146–162. https://doi.org/10.1080/00437956.1954.11659520
[24] Pentti. Kanerva and Pentti. 1988. *Sparse distributed memory.* MIT Press. 155 pages.
[25] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-Thought Vectors. *CoRR* abs/1506.0 (2015).
[26] Ji Young Lee and Franck Dernoncourt. 2016. Sequential Short-Text Classification with Recurrent and Convolutional Neural Networks. (2016).
[27] Stan Lipovetsky and Michael Conklin. 2001. *Analysis of Regression in Game Theory Approach.* Vol. 17. 319–330 pages. https://doi.org/10.1002/asmb.446
[28] Scott M Lundberg, Gabriel G Erion, and Su-In Lee. [n. d.]. *Consistent Individualized Feature Attribution for Tree Ensembles.* Technical Report. http://github.com/slundberg/shap
[29] Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems 30* (2017), 4765–4774.
[30] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. *Learning Word Vectors for Sentiment Analysis.* Technical Report. 142–150 pages.
[31] Alireza Makhzani and Brendan Frey. 2013. k-Sparse Autoencoders. (2013).
[32] Alireza Makhzani and Brendan Frey. 2014. Winner-Take-All Autoencoders. (9 2014).
[33] Stephen Mcgregor, Kat Agres, Matthew Purver, and Geraint Wiggins@qmul Ac Uk. 2015. From Distributional Semantics to Conceptual Spaces: A Novel Computational Method for Concept Creation Geraint A. Wiggins. *Journal of Artificial General Intelligence* 6, 1 (2015), 55–86. https://doi.org/10.1515/jagi-2015-0004
[34] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. (1 2013).
[35] Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. , 38 pages. https://doi.org/10.1016/j.artint.2018.07.007
[36] David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing Semantic Coherence in Topic Models. *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP-11)* 2 (2011), 262–272.
[37] Brian Murphy, Partha Pratim Talukdar, and Tom Mitchell. 2012. Learning Effective and Interpretable Semantic Models using Non-Negative Sparse Embedding. *COLING* December 2012 (2012), 1933–1950.
[38] David Newman, JH Lau, and Karl Grieser. 2010. Automatic evaluation of topic coherence. *Technologies: The 2010* June (2010), 100–108.
[39] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. 2017. Feature Visualization. *Distill* 2, 11 (11 2017), e7. https://doi.org/10.23915/distill.00007
[40] Bruno A. Olshausen and David J. Field. 1997. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research* 37, 23 (12 1997), 3311–3325. https://doi.org/10.1016/S0042-6989(97)00169-7
[41] Jeffrey Pennington, Richard Socher, and Christopher D Manning. [n. d.]. *GloVe: Global Vectors for Word Representation.* Technical Report.
[42] Matthew E Peters, Mark Neumann, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. [n. d.]. *Deep contextualized word representations.* Technical Report.
[43] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. *Open AI* (2019).
[44] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. (2 2016).
[45] Michael Röder, Andreas Both, and Alexander Hinneburg. [n. d.]. Exploring the Space of Topic Coherence Measures. ([n. d.]). https://doi.org/10.1145/2684822.2685324
[46] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. [n. d.]. *Taking the Human Out of the Loop: A Review of Bayesian Optimization.* Technical Report.
[47] Lloyd S. Shapley. 1952. A Value for n-Person Games. (1952).
[48] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning Important Features Through Propagating Activation Differences. (4 2017).
[49] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning Important Features Through Propagating Activation Differences. (4 2017).
[50] Fei Sun, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. 2016. Sparse Word Embeddings Using L1 Regularized Online Learning. In *IJCAI*.
[51] Valentin Trifonov, Octavian-Eugen Ganea, Anna Potapenko, and Thomas Hofmann. 2018. Learning and Evaluating Sparse Interpretable Sentence Embeddings. (2018), 200–210.
[52] Sida Wang and Christopher D Manning. 2012. *Baselines and Bigrams: Simple, Good Sentiment and Topic Classification.* Technical Report. 8–14 pages.
[53] Ji-Rong Wen. [n. d.]. *Query Clustering Using User Logs.* Technical Report.