Interactive Construction of User-Centric Dictionary for Text Analytics

Ryosuke Kohita kohi@ibm.com
Issei Yoshida **Hiroshi Kanayama** **Tetsuya Nasukawa** {issei,hkana,nasukawa}@jp.ibm.com

IBM Research

Abstract

We propose a methodology to construct a term dictionary for text analytics through an interactive process between a human and a machine. The interactive approach helps the creation of flexible dictionaries with precise granularity required in text analysis. This paper introduces the first formulation of interactive dictionary construction to address this issue. To optimize the interaction, we propose a new algorithm that effectively captures an analyst's intention starting from only a small number of sample terms. Along with the algorithm, we also design an automatic evaluation framework that provides a systematic assessment of any interactive method for the dictionary creation task. Experiments using real scenario based corpora and dictionaries show that our algorithm outperforms baseline methods, and works even with a small number of interactions. Also, we provide our dataset for future studies[1].

1 Introduction

Since the emergence of practical interests in text analytics that finds insights from massive documents (Nasukawa and Nagano, 2001), there are several requirements for enhancing valuable discoveries. The one critical issue we tackle in this paper is an **effective construction of a term dictionary** (Godbole et al., 2010). The term dictionary, which is an **arbitrary set of terms**, is used in text analytics to represent interesting analysis perspectives (Nasukawa and Nagano, 2001; Nasukawa, 2009); for example, dictionaries of "product names" and "evaluative description" are required for mining customer reputations about products. The motivation of this paper is how to reduce the human workload for the dictionary construction as much as possible. To this end, we establish a methodology of **interactive dictionary construction** that incrementally captures an analyst's intention starting from a small number of sample terms and enables him/her to effortlessly expand terms in the intended dictionary through suggestions by a machine.

[1]https://github.com/kohilin/IDC-evalset.git

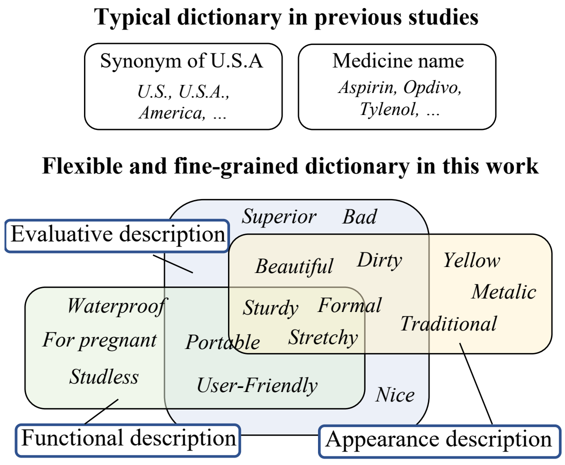


Figure 1: Typical dictionaries in previous works (upper) and fine-grained dictionaries in this work (lower)

The term dictionary for text analytics is expensive to be constructed because we need to focus more on terms with flexible granularity for in-depth analysis (Takeuchi et al., 2009; Godbole et al., 2010; Mostafa, 2013). For instance, if the analyst wants to examine product evaluation from both its function and appearance, he/she then needs to separately create those dictionaries whose boundaries are vague and overlapped (Figure 1). In short, we need to group any terms the analyst wants together depending on documents and the objective of analysis, which forces an ad hoc construction of the term dictionary. This situation is rather severe in the real-world tasks because the vocabulary size for an exhaustive search of the texts is vast, and the analyst will go through repeated trial and error of creating dictionaries until he/she reaches findings.

At present, there is a demand for a machine that decreases the cost of the ad hoc dictionary construction. As the dictionary construction can be considered as a type of collecting terms, there is a related research field — set expansion that expands a small set of terms by means of bootstrapping (Pantel and Pennacchiotti, 2006). This approach automatically finds new terms for the given set from documents in accordance with a predefined exploration strategy (Pantel et al., 2009; He and Xin, 2011). Although such an automatic procedure is advantageous for reducing the human workload, the quality of the collected terms is suspicious for a term dictionary. For example, a good analysis requires more fine-grained dictionaries than the original targets in set expansion such as distinct ontological terms (e.g., country name, Shen et al. 2017, 2018).

Several studies have incorporated a human in the term collection process (Godbole et al., 2010; Coden et al., 2012). Specifically, dictionaries are built in an interactive process where the human gives feedback to the machine and the machine suggests candidates based on the given feedback (Alba et al., 2017, 2018). Such a human-in-the-loop approach has been an active topic in other fields as well, for instance, image classification (Cui et al., 2016), dialogue system (Li et al., 2017), and audio annotation (Kim and Pardo, 2018). We can generally expect that a reliable feedback provided by human makes a system more accurate. With respect to dictionary construction, however, experimental results in this vein are limited due to the empirical evaluation by just a few participants and the use of a coarse dictionary as the test items. In short, it is a still open question — what is a critical issue for interactive construction of fine-grained term dictionary for text analytics?

Moving in the same promising direction of leveraging both a human and a machine, we establish a well-defined and effective methodology for constructing the term dictionary. In summary, our contribution in this paper is fourfold: (i) We formulate the interactive process of a term collection, which brings clarity to the problem to be solved (§2). (ii) We develop a method that captures an analyst's intention from a small number of samples with our formulation as the basis (§3). (iii) We propose an automatic evaluation framework that provides a systematic assessment for interactive methods (§4). (iv) Our experimental results show that the proposed method surpasses baseline methods such as set expansion, word embedding and a linear classifier on the crowdsourced dataset. The dataset emulates the real-world scenario of flexible and fine-grained dictionary construction, and we distribute the dataset to the public (§5).

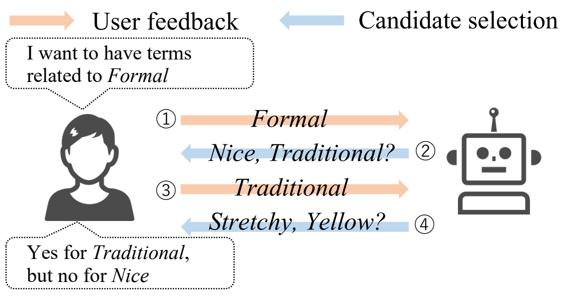


Figure 2: Interactive dictionary construction

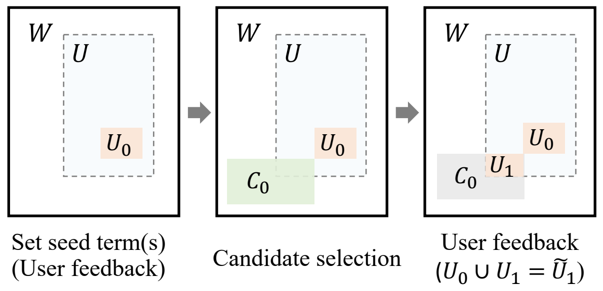
2 Task Definition

In this section, we provide the definitions and notations used throughout this paper.

First, a **term** is a string representation of a certain notion such as "*apple*" and "*New York*". A **dictionary** is a collection of terms. A **user** denotes the person who wants to construct a dictionary, and **system** denotes the machine that helps the user. Let W be the whole set of terms in documents. Our objective is to rapidly find as many terms of the user's interest $U \subset W$ as possible.

As seen in Figure 2, interactive dictionary construction is defined as an iterative process in which each iteration consists of the following steps: 1) **User feedback** in which the user selects terms for the dictionary from the current candidate terms, and 2) **Candidate selection** in which the system finds candidate terms for the next user feedback. For the i-th iteration ($i = 0, 1, 2, \ldots$), let C_i be the set of terms that the system finds in the candidate selection step and U_i be the set of terms that the user selects from C_{i-1} in the user feedback step as positive examples. Here, U_0 is a special feedback we call *seed terms* that are directly given by the user first. Note that, because we wish to expand the dictionary, each term in C_i should be new to the user in the $(i + 1)$-th iteration.

In the i-th step of the user feedback ($i \geq 1$), we assume that the user can annotate which terms in C_{i-1} are in U without being aware of the whole



W
U
U_0
C_0
U_1
Set seed term(s) (User feedback)
Candidate selection
User feedback ($U_0 \cup U_1 = \widetilde{U}_1$)

Figure 3: Task definition

U. So, $U_i \subset U$ for each i. Let $\widetilde{U}_i := \cup_{m=0}^{i} U_m$ be the set of words of the user's interest that is found by the end of the i-th iteration.

However, it is impractical to define our objective as an optimization problem for the asymptotic convergence of $\widetilde{U}_i$ because the user feedback is done by a human, and i cannot be large. Hence, we try to maximize $|C_i \cap U|$, the number of suggested terms that match the user's interest. Also, since C_i is manually selected by a human user, the proper size of C_i is practically limited to $5 \sim 10$.

Figure 3 shows the steps from setting the seed terms to giving the first feedback to the first candidates. Using the example in Figure 2, U_0 is $\{$*Formal*$\}$, C_0 is $\{$*Nice, Traditional*$\}$, U_1 is $\{$*Traditional*$\}$, and $C_0 \backslash U$ is $\{$*Nice*$\}$. The system then next selects C_1 based on U_0 and U_1 (i.e., $\widetilde{U}_1$) from W except for the shown terms $C_0 \cup \widetilde{U}_1$. It is important that we design the system to be effective so that the overlapped area of C_i and U becomes larger.

There are two major challenges for this problem; one is number of seed terms, and the other is term overlaps of different dictionaries. In terms of the first issue, we have only a few seed terms for the target dictionary at the first iteration. If the system requires more seed terms, the advantage of the system drops because it contradicts our purpose to decrease the human workload in constructing the dictionary. Therefore, we need a method that captures the user's intention from a smaller number of samples. In terms of the second issue, identifying terms of user's interest is difficult because boundaries between dictionaries are often overlapped in text analytics as seen in Figure 1. In other words, the system need to be more sensitive to subtle semantic differences only with a few feedbacks.

3 Method

In this section, we first describe a previous candidate selection model, SetExpan algorithm (Shen et al., 2017) that inspired our method (§3.1). Subsequently, we introduce our method as the weighted version of SetExpan with improvements in dealing with interactive settings (§3.2∼). Throughout this section, we discuss the i-th step of candidate selection for a certain i. For simplicity, C_i and $\widetilde{U}_i$ are denoted as C and $\widetilde{U}$, respectively.

3.1 Candidate Selection: Similarity Scoring based on Feature Collection

As we stated in §2, the objective of the task is to suggest C that contains as many terms in U as possible. Recall that $\widetilde{U}$ is a set of positive examples for terms of the user's interest that are found in previous steps. Following the strategy taken in set expansion (Shen et al., 2017), a straightforward and reasonable approach to determine C is to define $Sim(e, e'|F)$ which returns a similarity score for two terms e and e' based on a set of features F, and then to select terms that are most similar to the positive terms in $\widetilde{U}$.

The issue is how to obtain the ideal F that assigns a higher score to terms potentially included in U. Shen et al. (2017) formulates this feature selection problem as choosing features with the number of fixed-size Q so that the positive terms are most similar to each other:

$$F^* = \arg\max_{|F|=Q} \sum_{1 \leq i \leq j \leq n} Sim(e_i, e_j|F), \qquad (1)$$

where $\widetilde{U} := \{e_1, \ldots, e_n\}$. They propose using the Jaccard coefficient for $Sim(e_i, e_j|F)$, which narrows the optimization problem to a binary decision on whether to use each feature. This combinatorial problem is NP-hard; hence, they use heuristics to choose an approximation of F^*.

3.2 From Feature Selection to Feature Weighting with Predefined Similarity

Instead of explicitly choosing features to use in the similarity calculation, we consider using all of the possible features $\{f_1, \ldots, f_L\}$ with the weight $w_k \in \mathbb{R}$ for each feature f_k. In addition, we define our optimization problem as finding the best w_k for f_k $(k = 1, \ldots, L)$.

Let us develop a formula that extends (1) and takes w_k into consideration. First, in such a formula, $Sim(e_i, e_j|F)$ should be a weighted sum of the similarity score for each feature f_k, denoted as $Sim(e_i, e_j|f)$. By replacing F with $\boldsymbol{w}$ in the expression of the similarity function, we have

$$Sim(e_i, e_j|\boldsymbol{w}) = \sum_{k=1}^{L} w_k \cdot Sim(e_i, e_j|f_k). \quad (2)$$

Next, to define the similarity between a term e and $\widetilde{U}$, we assume that the similarity is the average of similarities between e and $e_i \in \widetilde{U}$, that is,

$$Sim(e, \widetilde{U}|\boldsymbol{w}) := \frac{1}{n}\sum_{i=1}^{n} Sim(e, e_i|\boldsymbol{w}). \quad (3)$$

The initial formulation of our optimization problem is thus as follows:

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} \sum_{1\le i\le n} Sim(e_i, \widetilde{U}|\boldsymbol{w}). \quad (4)$$

We show in the Appendix that our formulation of (4) can be considered as the weighted version of (1) under the natural condition that $Sim(e_i, e_i|f_k) = Sim(e_j, e_j|f_k)$ for any i, j, and k, and $\sum_{k=1}^{L} w_k = 1$. It is easy to set $Sim(e, e'|f_k)$ satisfying this condition. For a feature f_k, we define a vector $\boldsymbol{v}_{f_k}(e)$ of an e and define $Sim(e, e'|f_k)$ as the standard inner product of $\boldsymbol{v}_{f_k}(e)$ and $\boldsymbol{v}_{f_k}(e')$. Then by normalizing all these vectors, $Sim(e_i, e_i|f_k) = \|\boldsymbol{v}_{f_k}(e_i)\| = 1$ holds for any i; hence, the condition is satisfied, and that is a conventional cosine similarity of word vectors (Levy et al., 2015). Thus, any mapping from W to a vector space is available as a feature such as the tf-idf of terms and discrete features (Manning et al., 2008), word2vec (Mikolov et al., 2013), or GloVe (Pennington et al., 2014). Note that the dimension of the vector space may be different among the features.

Hence, we assume $\boldsymbol{v}_{f_k}(e)$ is defined for each feature f_k and any e. When we use $Sim(e, e'|f_k) = \boldsymbol{v}_{f_k}(e) \cdot \boldsymbol{v}_{f_k}(e')$, (2) is computed by

$$Sim(e_i, e_j|\boldsymbol{w}) = \sum_{k=1}^{L} w_k \cdot \boldsymbol{v}_{f_k}(e_i) \cdot \boldsymbol{v}_{f_k}(e_j), \quad (5)$$

and by a simple calculation, (3) is equal to

$$Sim(e, \widetilde{U}|\boldsymbol{w}) = \sum_{k=1}^{L} w_k \cdot \boldsymbol{v}_{f_k}(e) \cdot \boldsymbol{v}_{f_k}(\widetilde{U}), \quad (6)$$

where $\boldsymbol{v}_{f_k}(\widetilde{U}) := \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{v}_{f_k}(e_i)$ is the centroid vector for $\{\boldsymbol{v}_{f_k}(e_i)\}_{i=1,\dots,n}$ in the feature space of f_k. We simply call $\boldsymbol{v}_{f_k}(\widetilde{U})$ the centroid of $\widetilde{U}$. Formulas (5) and (6) demonstrates that the similarity between any two terms can be measured by combining the characteristics of the L different feature spaces. We "select" the feature spaces in which terms in $\widetilde{U}$ become similar to each other by adjusting the weights, as shown in Figure 4.

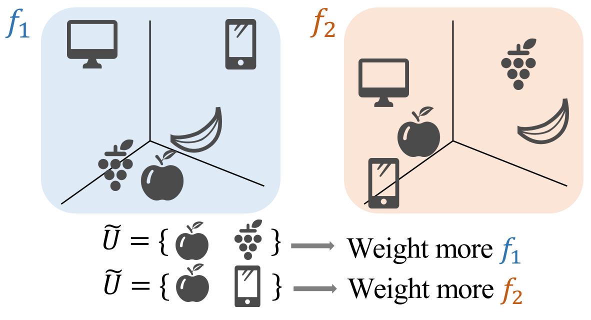

f_1
f_2
$\widetilde{U}$ = { } Weight more f_1
$\widetilde{U}$ = { } Weight more f_2

Figure 4: Feature weighting puts weights on feature spaces by placing terms of user's interest nearby.

Note that our feature weighting formulation is categorized as a conventional linear regression that finds f_k characterizing $\widetilde{U}$ via the weights. Instead of calculating the weights for bare features of each term, our method estimates those for differently predefined feature spaces (i.e., the similarity scores in these spaces). It aims to mitigate the difficulty of finding optimal weights for the vast number of features only from few labeled samples. However, the drawback is that this sacrifices a model's degree of freedom; therefore, we test the effectiveness of our proposed model compared to an ordinary linear classifier in the experiment.

3.3 Optimization by User Feedback

Although the initial formulation (4) proved to be a natural extension of the discrete version of feature selection, it does not always work as expected. In this section, we discuss the reason for this and how we can improve the initial formulation of our optimization problem.

By substituting (2) and (3) into (4), the objective $\sum_{1\le i\le n} Sim(e_i, \widetilde{U}|\boldsymbol{w})$ is a linear function of $\boldsymbol{w}$. Assuming that $\sum_{k=1}^{L} w_k = 1$, the optimal $\boldsymbol{w}$ is determined by putting all the weight values on a particular feature space which has the highest score in the averaged similarity between the terms in $\widetilde{U}$ and the centroid of $\widetilde{U}$. This is equivalent to selecting only one feature space for the similarity computation. Such extreme optimization is not suitable for our interactive setting because the target dictionary is obscure, especially in earlier iterations. We want the system to diversify the candidate terms to broadly cover the user's interests and allow the user to discover related vocabularies for a customized dictionary. To address this issue, we modify our formulation of (4) as

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} \min_{1 \le i \le n} Sim(e_i, \widetilde{U} | \boldsymbol{w}). \quad (7)$$

We maximize the minimum similarity score between a term in $\widetilde{U}$ and the centroid of $\widetilde{U}$. The idea here is to reduce the distance between the farthest positive term and the centroid. This strategy is analogous to those used in active learning, where examples near the separating hyperplane are actively leveraged (Schohn and Cohn, 2000). Our objective function $\min_{1 \le i \le n} Sim(e_i, \widetilde{U}|\boldsymbol{w})$ is a concave function of $\boldsymbol{w}$ (see Appendix); therefore, we can solve it by (for example) gradient descent.

We can also leverage negative feedback, i.e., unselected terms in C, to make the system more sophisticated. Let $N := C \setminus U = \{z_1, \ldots, z_m\}$, then we can extend (7) by

$$\begin{aligned} \boldsymbol{w}^* &= \arg\max_{\boldsymbol{w}} \Big\{ \min_{1 \le i \le n} Sim(e_i, \widetilde{U}|\boldsymbol{w}) \\ &- \max_{1 \le j \le m} Sim(z_j, \widetilde{U}|\boldsymbol{w}) \Big\}. \end{aligned} \quad (8)$$

The second term on the right-hand side of (8) increases the distance between the closest negative term and the centroid of $\widetilde{U}$. Again the objective function of (8) is a concave function of $\boldsymbol{w}$; thus, the information of both positive and negative examples is taken into consideration to learn the optimal $\boldsymbol{w}^*$.

3.4 Feedback Denoising

Although our min-maximize optimization strategy diversifies candidates, it may be disadvantageous in terms of the system being affected by outliers. It happens that several terms in $\widetilde{U}$ (especially for manually fed terms such as seeds) distribute differently in possessing feature spaces compared to the rest of the positive terms. Such a case holds up the learning because the maximum similarity score of the outliers to the centroid is low. The left side of Figure 5 shows an example of this problem: specifically, the system cannot put a higher weight value on f_1 because the optimization target, which is the most distant one from the centroid ("watermelon" in this case), is biased to f_2.

Feedback denoising is a simple solution to this problem. We apply a clustering algorithm (e.g., K-Means) to terms in $\widetilde{U}$, and obtain K term sets $\widetilde{U}_{(0)}, \widetilde{U}_{(1)}, ..., \widetilde{U}_{(K)}$. Then, we conduct the optimization by replacing $\widetilde{U}$ in (7) and (8) with $\widetilde{U}_{(K^*)}$ where $K^* = \arg\max_K |\widetilde{U}_{(K)}|$, that is, the majority class among terms in $\widetilde{U}$ as shown in the right side of Figure 5. This is effective for denoising irregular terms with respect to feature distribution, and for guiding the system to a promising $\boldsymbol{w}^*$.

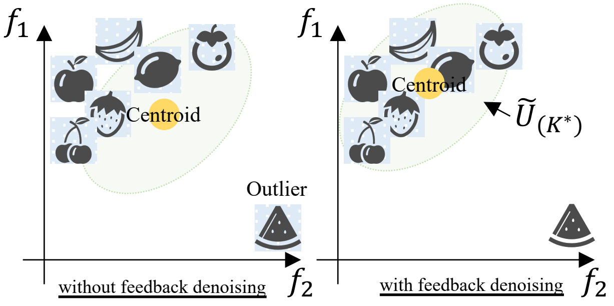


Figure 5: The difference in terms used in learning (blue) with/without feedback denoising.

4 Evaluation Framework

In this section, we explain an automatic evaluation framework for interactive dictionary construction. By using a predefined dictionary as the oracle dictionary U^*, we emulate the manual feedback process and apply a new evaluation metric to estimate the effectiveness of building a dictionary with consideration of the human interaction.

4.1 Human Emulation

We describe the emulation process with U^*, and the entire flow of the emulation procedure is in Algorithm 1. At the beginning of the emulation process, a small number of seed terms are randomly chosen from U^*, and U_0 is initialized with them ($l.1$). The number of iterations I ($l.2$) and the number of suggested terms per iteration $|C|$ ($l.3$) are also determined. The iteration consisting of user feedback and candidate selection is then launched. In every i-th iteration, the system first suggests the C_i based on the known positive terms $\widetilde{U}_{i-1}$ ($l.5$). After receiving the suggested C_i, the automatic evaluation process takes the intersection of C_i and U^*, and records the overlapped terms as U_i ($l.6$). It also takes the difference set of C_i and U^* as the negative terms N_i ($l.7$). If the system is trainable, its training process runs before moving to the next iteration ($l.8 - 10$).

Algorithm 1 Human emulation with oracle dictionary

```
1:  SET seed terms $U_0$ from $U^*$
2:  SET number of iterations $I$
3:  SET number of suggested terms per iteration $|C|$
4:  for $i = 1$ to $I$ do
5:      $C_i \leftarrow$ Suggest from $\widetilde{U}_{i-1}$
6:      $U_i \leftarrow C_i \cap U^*$
7:      $N_i \leftarrow C_i \backslash U^*$
8:      if System is trainable then
9:          Run training with $\widetilde{U}_i$ (and $\widetilde{N}_i$)
10:     end if
11: end for
```

4.2 A Metric for Effectiveness Estimation

In addition to the automatic evaluation process, we introduce a new metric that takes the interaction quality into account when evaluating the accuracy of the candidate selection.

The final goal of dictionary construction is to obtain a complete set of terms consistent with U^*; however, there is a limitation stemming from a user's workload in real scenarios. Given that an effective system should suggest terms of user's interest in earlier iterations, we propose *weighted coverage per iteration* ($WCpI$) as the evaluation metric for interactive dictionary construction:

$$WCpI = \frac{\sum_{i=1}^{I} (1-\alpha)^{i-1} \frac{|\widetilde{U}_i|}{\min\{i|C|, |U^*|\}}}{\sum_{i=1}^{I} (1-\alpha)^{i-1}}, \quad (9)$$

where α is the hyperparameter to adjust the importance of the iteration number. We illustrate the intuition of $WCpI$ in Figure 6. $WCpI$ is an area ratio of accumulated positive terms from system suggestions to its upper bound in each iteration. In short, it measures how many correct suggestions the system can provide in the comparison with a "perfect" system that never suggests unrelated terms.

We can also regulate the importance of iteration number by adjusting α. Specifically, a larger value of α underestimates the importance of terms found in the later iterations, in other words, it attaches importance to terms found in the earlier iterations. As an intuitive explanation based on an actual scenario, α is like representing a constant probability for the user to quit dictionary construction midway through. The graphs in Figure 6 compare the calculation of $WCpI$ for the same system suggestions. The right one with $\alpha = 0.1$, in which we assume the user quit creating a dictionary with 10% probability at every iteration, has a higher $WCpI$ than the left one with $\alpha = 0.0$.

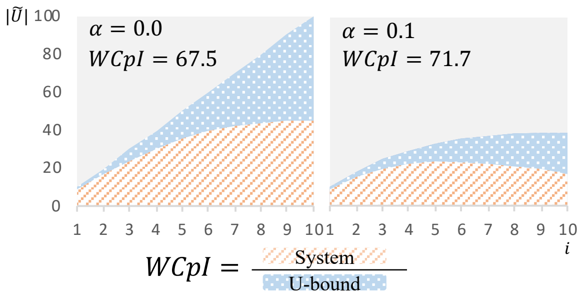


Figure 6: Weighted coverage per iteration $WCpI$. TheThe x- and y-axes are the number of iterations and accumulated positive terms, respectively. The blue and red areas represent the upper bound and system performance, respectively. The left side is for $\alpha = 0.0$ and the right side is for $\alpha = 0.1$ when $|C| = 10$, $|U^*| > 100$, $|U_i| = 10 - i$, and $I = 10$.

5 Experiments

We conduct an experiment following the automatic evaluation framework by using public datasets and oracle dictionaries created through crowdsourcing. In the experiment, we compare several methods in addition to our proposed method. As emulation parameters, we set number of seed terms ($|U_0|$), the number of terms in one suggestion ($|C|$), and number of total iterations (I) to 3, 10, and 30, respectively. Note that we tried different numbers of seeds (1 and 5), but the overall tendencies were the same.

5.1 Dataset

We used crowdsourcing to create oracle dictionaries on the Amazon review corpus (Blitzer et al., 2007), which is publicly available.[2] First, we explain the corpus processing and the procedure to construct the oracle dictionaries. We then describe the evaluation items. Our evaluation items will be publicly available for the system evaluation in future research.

Corpus. The corpus originally consists of sub corpora from 25 domains. Given that size and domain vary, we pick five domains; *apparel* (APP), *baby* (BAB), *camera & photo* (CAM), *health & personal care* (HEL), and *sports & outdoors* (SPO). We process the raw texts with spaCy [3] and its dis-

[2]https://www.cs.jhu.edu/ mdredze/datasets/sentiment/
[3]https://spacy.io/

tributed English model [4]. We then construct the vocabulary with words and noun chunks that appeared more than five times except for standard stopwords. Note that all terms in the vocabulary are identified after lemmatization by spaCy.

Oracle Dictionaries. For each selected corpus, we create oracle dictionaries through crowdsourcing.[5] In the task for workers, we provide predefined dictionaries and ask the worker to choose one or more dictionaries to which a given term belongs. For example, we prepared three independent dictionaries *nursery items* dictionaries for *sleeping, movement, and safety* in the BAB corpus, and asked a worker to judge which dictionary includes the term "*car seat*". With respect to each corpus, we define multiple dictionaries and request three workers to make judgments for every term in the vocabulary. We determine that a term is included in a dictionary when at least one of three workers choose the dictionary for the term. Note that we filter noisy users and their answers beforehand according to the reliability score estimated by the crowdsourcing service [6]. Finally, we also manually clean each dictionary. Excluding dictionaries consisting of less than 15 terms or too much noise, we eventually obtain 22 dictionaries. We list the dictionaries and example terms in Table 1.

Evaluation Item. We generate ten evaluation items per dictionary, for 220 items in total. An evaluation item consists of a unique set of seed terms (U_0) and the remaining terms in the corresponding dictionary as the oracle ($U^* := U \backslash U_0$). We suggest that fewer seed terms are adequate for evaluating an interactive dictionary construction method; because the purpose is to gather terms with a minimum human effort as mentioned in §2.

5.2 Methods

We compare four methods: Word2Vec, SetExpan, logistic regression, and our proposed method with several configurations. All methods possess the same vocabulary W, and all methods excluding Word2Vec use the same feature spaces: tf-idfs of Bag of Words, unigrams, bigrams, and word embeddings. Any feature space is applicable, though.

[4]https://spacy.io/models/en#en_core_web_sm

[5]https://www.figure-eight.com/

[6]Although we also tried other thresholds such as correspondence between three workers, this criterion provided the best balance of data cleanliness and size.

Word2Vec: Word2Vec is a popular and promising method for representing word meanings in a continuous vector space, and the vector similarity is naturally applicable to interactive dictionary construction (Alba et al., 2018). We use two computation methods of candidate selection based on Word2Vec. The first is `w2v(avg)` and involves simply taking cosine similarity with an averaged vector of terms among $\widetilde{U}$. The second is `w2v(rank)` and calculates the mean reciprocal rank from terms in $\widetilde{U}$. Both select the candidates in order of their estimated scores. The embeddings are learned for each corpus with the gensim implementation using the default parameters.[7]

SetExpan: We implement SetExpan (SE; Shen et al. 2017), which is a feature-selection method for conventional set expansion. The original version does not involve the user in the iteration and updates $\widetilde{U}_i$ according to its own criteria to filter incorrect terms. In our scenario, we provide the correct terms in the update phase of $\widetilde{U}_i$. We use the same input features with other methods and set the hyperparameters to those Shen et al. (2017) reported as best.

Logistic Regression: We include logistic regression in our comparison because the feature weighting is one of the conventional types of linear discriminant analysis. The logistic regression version, LR, takes a word representation and then predicts the probability of the word appearing in a current dictionary. For word representation, we concatenate vectors in each feature space (explained in §5.2) and then use the vector compressed into 300 dimensions with singular value decomposition. In every iteration, we train LR from scratch with positive and negative terms. For the negative terms at the first iteration (i.e., N_0), however, we randomly select $|U_0|$ of negative words from the entire vocabulary except for dictionary terms. We select candidates following the order of estimated probabilities. While we tried other regression models (SVM and Random Forest) and dimensions of the input vector (non-compression, 50, 100, 200, 500, and 1000), the above condition was the best configuration.

Feature Weighting with Predefined Similarity: We test six versions of our proposed methods:

- `FWPS`: Our base model without optimization

[7]https://radimrehurek.com/gensim/models/word2vec.html

Corpus	Dictionary name	Size	Examples
APP	Accessory	63	*flower, watch, glove, ring, case, scarf, garter, holder*
	Wearables for upper body	92	*visor, tuxedo, sweatshirt, tank, pajama, blanket, glass, outer*
	Wearables for lower body	83	*gown, robe, harness, sandal jersey, boot, loafer, nightgown*
	Items for outdoor	39	*sweatshirt, glove, trunk, bike, backpack, coat, hat*
BAB	Nursery items for transport	37	*carrier, stroller, leash, walker, backpack, strap, seat cover, sunshade*
	Nursery items for safety	74	*cover, sterilizer, infant car seat, seat belt, beeping, monitor, sunshade*
	Nursery items for sleeping	62	*cover, hammock, bedding, mattress, sleep sack, bumper, cushion, lamp*
	Enjoyments for baby	134	*car, playmat, crayon, ring, dad, bell, bird, toy box*
	Wearables for baby	52	*shoe, bouncer, towel, cloth, comforter, fleece, diaper, head support*
CAM	Scene of photograph	84	*summer, space, excursion, cruise, pool, face, wildlife, land*
	Subject of photograph	71	*space, performer, ocean, magic, young, garden, action, snow*
	Functions of camera	108	*remote switch, waterproof, trigger, telephoto, interface, portrait*
	External accessories	93	*station, remote switch, polarizer, trigger, case, battery, microphone*
HEL	Health equipment or product	54	*bathtub, air bed, vitamin, heater, read glass, flosser, supplement, pillow*
	Appearance description	58	*clear, oily, tint, sharp, handy, masculine, cheap, small*
	Functional description	86	*naturally, refill, powerful, rapid, oily, sharp, smooth shave, handy*
	Beauty equipment or product	88	*mirror, nivea, eyebrow, vanity, straightner, dryer, vitamin,fragrance*
SPO	Body	57	*blood, knuckle, eye, nose, chin, nail, knee, face, bone, palm*
	Wearables	70	*pedometer, roll, strap, vest, rattle, cloth, boat, tent, slip, altimeter*
	Items for exercise	148	*fanny pack, dumbell, pod, rower, bottle, pedometer, knee pad, rack, towel*
	Items for outdoors	132	*bicycle, opener, bottle, rack, towel, strap, guitar, fanny pack*
	Movements in exercise	86	*sit, twist, pull, stand, situp, running, roll, rowing, swing setter, punch*

Table 1: Examples of dictionaries.

where $\boldsymbol{w}$ is uniform distribution $\rightarrow$ §3

- +PickOne: Selecting only one feature space with the highest similarity scores among positive terms $\rightarrow$ §3.3
- +Op(p): With optimization using positive feedback $\rightarrow$ Eq.(7)
- +Op(p/n): With optimization using both positive and negative feedback $\rightarrow$ Eq.(8)
- +Fd(p): With +Op(p) and feedback denoising $\rightarrow$ §3.4
- +Fd(p/n):With +Op(p/n) and feedback denoising $\rightarrow$ §3.4

We use the K-means algorithm for +Op(p) and +Op(p/n) with $K = 3$, though the overall trend was almost the same with $K = 2$ and 5.

Hybrid: We also introduce a joint method HB that combines LR and an FWPS version. The strategy is simple; HB firstly uses FWPS 's mechanism to broadly cover candidate terms, and then switches to LR when the amount of feedback increases. This mechanism naturally solves LR's problems that require negative feedback from the beginning and demand a moderate number of labels for training. Any of the FWPS versions can be combined with LR; therefore, we chose the best one for our experiment. The switch timing is empirically set to the 5-th iteration.

5.3 Results and Discussion

Table 2 lists the $WCpI$ scores for each method across five corpora with $\alpha = 0.0$. In all domain texts, HB outperforms the others. The scores of LR are second highest, which implies that a combination with a FWPS model boosts performance. Among the versions of FWPS, +PickOne largely drops in score, which indicates the importance of the min-maximizing optimization strategy for this task (see §3.3). However, at least when $\alpha = 0.0$ that assumes the user never quit the process in the midway through, the performances of FWPS and other versions with optimized w are not different much. In particular, the negative feedback tends to degrade the performance. Subsequently, SE, w2v(avg), and w2v(rank) perform poorly. SE may not be suitable for gathering arbitrary terms from a non-large corpus because it was originally designed and tested for collecting ontological terms from large-scaled data (Shen et al., 2017). Also, we find that leveraging embeddings in a straightforward manner is not sufficient, especially for interactive dictionary construction.

Let us now discuss changes when adjusting the $WCpI$'s α listed in Table 3. Ignoring corpus differences, we take the average scores among all evaluation items. The most crucial change can be found in LR which significantly drops in score along with an increase in α. When $\alpha = 0.1$, the score of LR already becomes inferior to most

Methods	APP	BAB	CAM	HEL	SPO
`SE`	21.20	18.83	11.34	17.01	16.79
`w2v(avg)`	18.51	12.77	10.36	14.60	14.28
`w2v(rank)`	24.39	12.71	10.29	18.63	17.04
`LR`	51.99	36.76	31.18	38.17	37.59
`FWPS`	46.90	34.32	27.61	38.17	35.56
`+PickOne`	18.51	12.79	10.36	14.40	14.28
`+Op(p)`	45.60	33.73	26.13	36.43	35.29
`+Op(p/n)`	43.42	30.88	23.13	33.19	31.91
`+Fd(p)`	46.17	34.92	26.76	37.60	36.03
`+Fd(p/n)`	46.33	32.01	25.36	37.04	34.63
`HB(+Fd(p))`	**53.07**	**37.38**	**32.31**	**42.22**	**39.74**

Table 2: $WCpI$ scores across corpora ($\alpha = 0.0$)

Methods	α			
	0.0	0.1	0.3	0.5
`SE`	17.05	14.36	14.26	15.46
`w2v(avg)`	14.10	10.83	9.39	9.63
`w2v(rank)`	16.61	12.42	9.95	9.68
`LR`	39.14	30.06	23.39	22.45
`FWPS`	36.51	32.02	30.49	31.51
`+PickOne`	14.07	10.93	9.64	9.95
`+Op(p)`	35.44	31.79	30.72	31.58
`+Op(p/n)`	32.50	29.17	28.89	30.32
`+Fd(p)`	36.30	32.42	**31.10**	**31.95**
`+Fd(p/n)`	35.07	30.92	29.56	30.53
`HB(+Fd(p))`	**40.95**	**34.14**	30.97	31.62

Table 3: Change in $WCpI$ scores when increasing α. The scores are averaged among all evaluation items.

of the `FWPS` versions. Also, the scores of `FWPS` tend to be higher with a larger value of α. When $\alpha \geq 0.3$, `+Fd(p)` performs the best among all methods. In short, `LR` suggests correct terms in latter iterations; while `FWPS`, in particular with trainable ones (`+Op(p)`, `+Fd(p)`), suggests correct terms in earlier iterations.

Figure 7 directly describes the score differences with different alphas by showing the hit ratios defined as $|U_i|/|C|$ in terms of each iteration number for `LR`, `+Fd(p)`, and `HB`. Regardless of the number of seed terms, `LR` suggests fewer correct terms in earlier iterations, but its hit ratio stably goes beyond `+Fd(p)` after obtaining a moderate number of training labels (around five iterations, i.e., fifty labels). On the other hand, `+Fd(p)` performs better by a large margin in earlier iterations than `LR`. In short, our method using predefined term similarities overcomes the smaller sample issue which a conventional linear classifier suffers from and contributes to quick dictionary construction. This result is practically important because the analyst will go through repeated trial and error — observing documents from various points of views — by creating many small dictionaries. In addition, such contrasts are much stronger when

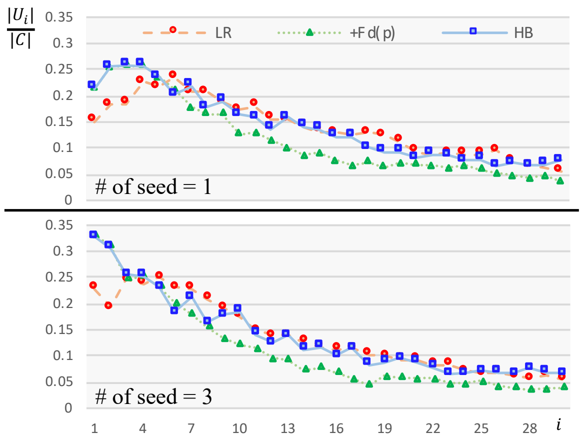


Figure 7: Hit ratio ($|U_i|/|C|$) in terms of each iteration number by `LR`, `+Fd(p)`, and `HB`. The upper and lower graphs start with one and three seeds, respectively.

we give only one seed term (the upper graph), which is also meaningful because the user often starts dictionary construction with only one seed term in a real situation.

`HB` enjoys both benefits of coverage by `LR` and quickness by `+Fd(p)`. In other words, a conventional classifier and our method are complementary; `LR` becomes favorable when the user prioritizes coverage than quickness, and `+Fd(p)` becomes favorable when vice versa. As a possible use case of `HB`, the analyst may quickly find interesting perspectives by creating various dictionaries with one of the `FWPS` methods, and once finding those, he/she switches to a linear classifier to expand the promising dictionaries more.

6 Conclusion

To the best of our knowledge, this paper proposes the first formulation of interactive dictionary construction for text analytics, which clarifies the critical issues to resolve. In response to those issues, we provide the method, the evaluation framework, and the experimental dataset. Also, our experimental results show the promising performances of our method in concern with real situations of text analytics. Our systematic study will pave the way to future research about the effective construction of dictionaries for text analytics.

Acknowledgement

We appreciate anonymous reviewers and their insightful comments. Also, we are grateful to Tadayuki Yoshida and Ryuki Tachibana for helpful discussions based on practical use-cases.

References

Alfredo Alba, Anni Coden, Anna Lisa Gentile, Daniel Gruhl, Petar Ristoski, and Steve Welch. 2017. Multi-lingual concept extraction with linked data and human-in-the-loop. In *Proceedings of the Knowledge Capture Conference*, pages 1–8.

Alfredo Alba, Daniel Gruhl, Petar Ristoski, and Steve Welch. 2018. Interactive dictionary expansion using neural language models. In *Proceedings of the 2nd International Workshop on Augmenting Intelligence with Humans-in-the-Loop*, pages 7–15.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447.

Anni Coden, Daniel Gruhl, Neal Lewis, Michael Tanenblatt, and Joe Terdiman. 2012. Spot the drug! an unsupervised pattern matching method to extract drug names from very large clinical corpora. In *Proceedings of the 2012 IEEE Second International Conference on Healthcare Informatics, Imaging and Systems Biology*, pages 33–39.

Yin Cui, Feng Zhou, Yuanqing Lin, and Serge J. Belongie. 2016. Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1153–1162.

Shantanu Godbole, Indrajit Bhattacharya, Ajay Gupta, and Ashish Verma. 2010. Building re-usable dictionary repositories for real-world text mining. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 1189–1198.

Yeye He and Dong Xin. 2011. Seisa: set expansion by iterative similarity aggregation. In *Proceedings of the 2011 World Wide Web Conference*, pages 427–436.

Bongjun Kim and Bryan Pardo. 2018. A human-in-the-loop system for sound event detection and annotation. *ACM Trans. Interact. Intell. Syst.*, 8(2).

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Jiwei Li, Alexander H. Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2017. Dialogue learning with human-in-the-loop. In *Proceedings of the 5th International Conference on Learning Representations*.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the 1st International Conference on Learning Representations*.

Mohamed M. Mostafa. 2013. More than words: Social networks' text mining for consumer brand sentiments. *Expert Systems with Applications*, 40(10):4241–4251.

Tetsuya Nasukawa. 2009. Text analysis and knowledge mining. In *Proceedings of the 8th International Symposium on Natural Language Processing*, pages 1–2.

Tetsuya. Nasukawa and Tohru. Nagano. 2001. Text analysis and knowledge mining system. *IBM Systems Journal*, 40(4):967–984.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 938–947.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 113–120.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Greg Schohn and David Cohn. 2000. Less is more: Active learning with support vector machines. In *Proceedings of the 17th International Conference on Machine Learning*, pages 839–846.

Jiaming Shen, Zeqiu Wu, Dongming Lei, Jingbo Shang, Xiang Ren, and Jiawei Han. 2017. Setexpan: Corpus-based set expansion via context feature selection and rank ensemble. In *Proceedings of the 2017 Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 288–304.

Jiaming Shen, Zeqiu Wu, Dongming Lei, Chao Zhang, Xiang Ren, Michelle T. Vanni, Brian M. Sadler, and Jiawei Han. 2018. Hiexpan: Task-guided taxonomy construction by hierarchical tree expansion. In *Proceedings of 24th International Conference on Knowledge Discovery & Data Mining*, pages 2180–2189.

Hironori Takeuchi, L. Venkata Subramaniam, Tetsuya Nasukawa, and Shourya Roy. 2009. Getting insights from the voices of customers: Conversation mining at a contact center. *Information Sciences*, 179(11):1584–1591.

A Appendix

A.1 Proof

We prove that our formulation of the optimization problem is a natural extension of that of SetExpan, assuming a reasonable normalization constraint for entity vectors and their weights. Notations follow from the main paper. Recall that the formulation of SetExpan is

$$F^* = \arg\max_{|F|=Q} \sum_{1\le i<j\le n} Sim(e_i, e_j|F), \qquad (1)$$

where Q is the number of features in F and is a fixed integer value.

Building on this, our formulation is

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} \sum_{i=1}^{n} Sim(e_i, \widetilde{U}|\boldsymbol{w}). \qquad (4)$$

Substitute (2) and (3) in the main paper into the above formulation to obtain

$$\begin{aligned}\boldsymbol{w}^* &= \arg\max_{\boldsymbol{w}} \sum_{i=1}^{n}\Big\{\frac{1}{n}\sum_{j=1}^{n} Sim(e_i, e_j|\boldsymbol{w})\Big\} \\ &= \arg\max_{\boldsymbol{w}} \frac{1}{n}\{2\times \sum_{1\le i<j\le n} Sim(e_i, e_j|\boldsymbol{w}) \\ &\quad + \sum_{i=1}^{n} Sim(e_i, e_i|\boldsymbol{w})\} \\ &= \arg\max_{\boldsymbol{w}}\Big\{\frac{2}{n}\sum_{1\le i<j\le n} Sim(e_i, e_j|\boldsymbol{w}) \\ &\quad + \frac{1}{n^2}\sum_{k=1}^{L} w_k\Big(\sum_{i=1}^{n}\|\boldsymbol{v}_{f_k}(e_i)\|\Big).\Big\}\end{aligned}$$

In the right-hand side of the last equation, the second term is a constant when all of the vectors $\{\boldsymbol{v}_{f_k}(e_i)\}_{i=1,\dots,n}$ have the same norm and $\sum_{k=1}^{L} w_k = 1$. Then our optimization problem is equivalent to

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} \sum_{1\le i<j\le n} Sim(e_i, e_j|\boldsymbol{w}),$$

which is a continuous version of (1).

Next, let us prove that in the modified version of our optimization problem ((7) in the main paper),

$$\min_{1\le i\le n} Sim(e_i, \widetilde{U}|\boldsymbol{w}) \qquad (10)$$

is a concave function of $\boldsymbol{w}$. Hence we can apply standard techniques of convex optimization to solve (7). First let us rewrite (10) as follows:

$$\begin{aligned}&\min_{1\le i\le n} Sim(e_i, \widetilde{U}|\boldsymbol{w}) \\ &= \min_{1\le i\le n}\sum_{k=1}^{L} w_k\cdot(\boldsymbol{v}_{f_k}(e_i)\cdot \boldsymbol{v}_{f_k}(\widetilde{U})) \\ &= \min_{1\le i\le n}\sum_{k=1}^{L} w_k\cdot x_{ik},\end{aligned}$$

where $x_{ik} = \boldsymbol{v}_{f_k}(e_i)\cdot \boldsymbol{v}_{f_k}(\widetilde{U})$. Then it is sufficient to prove the following lemma.

Lemma 1. *The following function is concave for $\boldsymbol{w}$ when $\boldsymbol{w}$ is defined on a convex set.*

$$g(\boldsymbol{w}) := \min_{1\le i\le n}\sum_{k=1}^{L} w_k\cdot x_{ik} \qquad (11)$$

Proof. It is a straightforward calculation by the definition of concavity. for any $\boldsymbol{w_1} = \{w_{11},\cdots,w_{1L}\}$, $\boldsymbol{w_2} = \{w_{21},\cdots,w_{2L}\}$, and $\lambda\in(0,1)$, we need to prove that

$$g((1-\lambda)\boldsymbol{w_1}+\lambda\boldsymbol{w_2}) \ge (1-\lambda)g(\boldsymbol{w_1})+\lambda g(\boldsymbol{w_2}).$$

We can compute the left-hand side of this equation by using (11):

$$\begin{aligned}&g((1-\lambda)\boldsymbol{w_1}+\lambda\boldsymbol{w_2}) \\ &= \min_{1\le i\le n}\sum_{k=1}^{L}((1-\lambda)w_{1k}+\lambda w_{2k})\cdot x_{ik} \\ &= \min_{1\le i\le n}\sum_{k=1}^{L}\{(1-\lambda)w_{1k}x_{ik}+\lambda w_{2k}x_{ik}\} \\ &\ge \min_{1\le i\le n}\sum_{k=1}^{L}(1-\lambda)w_{1k}x_{ik} \\ &\quad + \min_{1\le i\le n}\sum_{k=1}^{L}\lambda w_{2k}x_{ik} \\ &= (1-\lambda)\min_{1\le i\le n}\sum_{k=1}^{L} w_{1k}x_{ik} \\ &\quad +\lambda\min_{1\le i\le n}\sum_{k=1}^{L} w_{2k}x_{ik} \\ &= (1-\lambda)g(\boldsymbol{w_1})+\lambda g(\boldsymbol{w_2}).\end{aligned}$$

Here we use the inequality $\min_{1\le i\le n}(A_i+B_i) \ge \min_{1\le i\le n} A_i + \min_{1\le i\le n} B_i$ that holds for any sequences of real numbers $\{A_i\}_{i=1,\dots,n}$ and $\{B_i\}_{i=1,\dots,n}$. □

Since $\sum_{k=1}^{L} w_k = 1$, $0\le w_k$ $(k=1,\dots,L)$ is a convex set, we can apply this lemma to our objective function.