Confidence-Aware Graph Regularization with Heterogeneous Pairwise Features

Yuan Fang*
Univ. of Illinois at Urbana-Champaign
201 N Goodwin Avenue
Urbana, IL 61801, USA
fang2@illinois.edu

Bo-June (Paul) Hsu
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
paulhsu@microsoft.com

Kevin Chen-Chuan Chang
Univ. of Illinois at Urbana-Champaign
201 N Goodwin Avenue
Urbana, IL 61801, USA
kcchang@illinois.edu

ABSTRACT

Conventional classification methods tend to focus on features of individual objects, while missing out on potentially valuable pairwise features that capture the relationships between objects. Although recent developments on graph regularization exploit this aspect, existing works generally assume only a single kind of pairwise feature, which is often insufficient. We observe that multiple, heterogeneous pairwise features can often complement each other and are generally more robust in modeling the relationships between objects. Furthermore, as some objects are easier to classify than others, objects with higher initial classification confidence should be weighed more towards classifying related but more ambiguous objects, an observation missing from previous graph regularization techniques. In this paper, we propose a Dirichlet-based regularization framework that supports the combination of heterogeneous pairwise features with confidence-aware prediction using limited labeled training data. Next, we showcase a few applications of our framework in information retrieval, focusing on the problem of query intent classification. Finally, we demonstrate through a series of experiments the advantages of our framework on a large-scale real-world dataset.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

graph regularization, pairwise features, confidence, query intent classification, applications in information retrieval

*Work done during an internship at Microsoft Research.

SIGIR'12, August 12–16, 2012, Portland, Oregon, USA.

1. INTRODUCTION

Many applications in information retrieval (IR) call for effective classification techniques. For instance, the problem of text categorization is fundamental to a myriad of real-world tasks, such as automated email organization, spam detection, and information filtering. As another example, fine-grained query intent classification enables a search engine to direct users to the intended search verticals, greatly enhancing user experience.

While any conventional machine learning algorithm can be used for classification in IR applications, straightforward adaptation is generally unfruitful, since there exist challenges beyond traditional classification tasks. One prominent issue is the sparsity of feature vectors. For instance, in our query intent classification dataset, about 95% of the queries contain no more than five words.

Recent developments in graph-based regularization [20, 18, 2, 3] somewhat tackles the problem of feature sparsity. Instead of object-level features that are independently extracted from each object, we now consider pairwise features that are extracted from each *pair* of objects. Hence, each object can potentially be paired with any other object for feature extraction. In this work, we employ pairwise features derived from similarity functions between objects, such that a graph can be constructed by linking similar objects with edges. The edges can be used to propagate classification evidence from one object to another based on the key assumption of *consistency*—similar objects are likely to have similar class labels [20]. In other words, the classification of an object can be "helped" or regularized by its neighbors on the graph. In comparison, conventional classification algorithms cannot easily incorporate pairwise features without blowing up the parameter space and subsequently requiring more labeled training data.

Given the above benefits, graph-based regularization has also been adopted in IR applications, most notably through the use of a click graph [12, 7, 11] in query intent classification. A click graph usually consists of queries and websites as its vertices. A query q and a website w are connected by an edge if q results in a click landing on w. Intuitively, two queries connected to the same website are related, and thus should share a similar prediction.

While making the same assumption of consistency, we are motivated by two core observations, which entail key intuitions towards a more effective graph-based regularization framework.

First, most existing regularization framework only deals with a single kind of pairwise feature. However, heterogeneous pairwise features often exist. For instance, in query intent classification, queries can be related through not only clicks, but also other pairwise features, including lexical similarity ("acer laptop" and "laptop" are related due to word overlap), co-session (the second query could be a refinement of the first in the same session), and the simi-

larity of search result pages (two queries that retrieve similar pages may be similar themselves). These different pairwise features potentially complement each other, as some may only cover a small fraction of all queries (*e.g.*, not all shopping queries have associated product clicks). In addition to sparsity, some pairwise features may lack the robustness to capture similarities between two objects. *E.g.*, while both queries "hp 3 in 1" and "canon printer" refer to printers, they contain no common words and thus their similarity cannot be represented by lexical pairwise features. To support heterogeneous pairwise features, the challenge lies in how different similarity functions can be aggregated to optimize the regularization, which is missing from previous regularization frameworks.

Second, in existing graph regularization, the extent to which the classification of an object o influences o' only depends on their similarity. Greater similarity between o and o' means that they have stronger influences on each other. While this is reasonable, a second factor, largely overlooked by previous work, is the confidence of classification. If we are more confident about the prediction on o, we expect it to influence its neighbors on the graph more. On the other hand, if we are unsure about the prediction on o, we should minimize its influence. In other words, we use objects that are easier to classify (with higher classification confidence) to help predict harder ones (with lower classification confidence), but less so the other way round. Existing regularization works do not provide for a mechanism to incorporate classification confidence. In contrast, our modeling of objects with Dirichlet priors allows us to interpret the observation counts as confidence.

Finally, another challenge of classification lies in the requirement of a large amount of labeled training data, especially in real-world IR applications with a large number of classes. Our experiments on query intent classification involve 2043 classes, and thus require much more labeled training queries than, say, a binary classification task. Our framework deals with the shortage of labeled training data in two ways. To begin with, like other graph-based regularization frameworks (*e.g.*, [20, 18]), classification evidence is propagated across the graph along the edges. By using more unlabeled data, we generally obtain a denser graph that promotes such propagation, and hence improve the performance. Our experiments show that adding more unlabeled data yields superior results, despite using the same set of labeled training data. Next, our framework allows for fewer parameters, independent of the number of classes. Hence, even with limited training data and a large number of target classes, our technique is effective.

In this paper, we propose a regularization framework that can be applied to different applications, hinged on our insights above. To summarize, we make the following contributions:

- **Motivation:** We recognize the need for heterogeneous pairwise features that complement each other, and confidence-aware regularization that distinguishes objects of different initial classification confidence. (Sect. 1)
- **Framework**: We develop a generic graph-based regularization framework. The framework supports arbitrary heterogeneous pairwise features, is confidence-aware, and works with a limited amount of training data. (Sect. 3)
- **Applications**: We showcase how our framework can be applied to a few applications in IR, primarily focusing on query intent classification in the shopping domain. (Sect. 4)
- **Experiments**: We conduct extensive experiments on a real-world dataset for query intent classification. Our approach substantially outperforms traditional regularization methods which do not consider heterogeneous pairwise features or confidence, validating our observations. (Sect. 5)

2. RELATED WORK

Pairwise features. Many IR classification tasks are plagued by feature sparsity. Thus, a significant amount of research has studied augmenting the feature vector (*e.g.*, [4, 16, 15, 5] for query-intent classification). Unfortunately, these techniques are generally not universal to other classification tasks. Driven by the hypothesis that "similar objects share similar labels" [20], we consider pairwise features that allow for feature extraction from any pair of objects, potentially expanding the feature space.

Although it is difficult to incorporate pairwise features in conventional classification algorithms without blowing up the parameter space, they have been used in recent graph-based regularization frameworks [20, 18, 2, 3] and related random walk approaches [17, 7]. Examples of their applications in IR include the use of click graphs [12, 10, 7, 11] in query intent classification, and document affinity matrices [8, 9] in text retrieval. However, unlike this paper, most of these works do not consider heterogeneous pairwise features, which complement each other and thus play an important role in improving classification accuracy. While a recent work [11] explores a content-based pairwise feature in addition to co-clicks, its treatment is limited as it lacks a mechanism to effectively aggregate arbitrary pairwise features. In particular, their parameters are manually selected, making it difficult to extend to more pairwise features. In comparison, we learn the parameters via an iterative optimization process.

Confidence. Existing regularization frameworks [20, 18, 2, 3] and their task specific realizations (*e.g.*, [12, 10, 7, 11] for query intent classification and [8, 9] for text retrieval) do not recognize the importance and hence take advantage of the classification confidence associated with objects. We observe the need of confidence-aware regularization, and exploit it to improve classification accuracy. Specifically, we are more confident about objects that are easier to classify, which should have a larger influence towards the prediction of their neighbors.

Limited training data. To deal with the shortage of labeled training data, many task specific techniques exist (*e.g.*, [1, 12, 13, 15] for query-intent classification). However, they are often highly tailored and cannot be easily extended to a generic framework. In our framework, we benefit from using more unlabeled data by exploiting the relationships between objects, leveraging the semi-supervised nature of graph-based regularization [20, 18, 2, 3]. In addition, our framework involves a small parameter space independent of the number of target classes, and thus performs well even with limited training data and thousands of target classes.

3. REGULARIZATION FRAMEWORK

In this section, we develop our graph regularization framework. It is universal to different classification tasks, including our example application on query intent classification in Sect. 4.

Given a set of objects O and a set of classes C, each object $o \in O$ has some distribution over C. We regularize the class distribution of o by the class distributions of objects similar to o, *i.e.*, adjust the prediction on o according to objects similar to it. To start, we capture objects and their pairwise similarity features using a graph. Next, based on the graph, we introduce our regularization model.

3.1 Object-Relationship Graph

We model objects and their pairwise features with a graph $G = (O, R)$. O is the set of objects that define the *vertices*. R is the set of relationships between objects that describe the *edges* of the graph, which are derived from the pairwise features between ob-

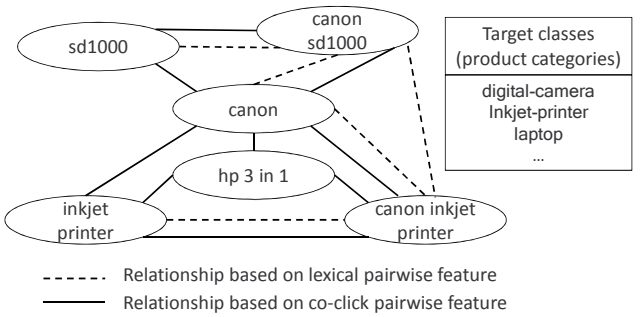

Figure 1: Example graph for classification of shopping queries.

jects. We use the toy graph in Fig. 1 for the task of query intent classification in the shopping domain as our running example.

Vertices. Each object or the target entity of classification is a vertex on the graph. For example, the query "canon" in Fig. 1 is a vertex to be classified as a digital camera or inkjet printer, among others.

Edges. An edge on the graph is represented as a triplet $(o, o', \tau) \in R$, which models the relationship between two objects o and o' based on a pairwise feature τ. Unlike traditional object-level features that are independently extracted from each object (*e.g.*, words in each document in text classification), τ is in fact a feature defined on a *pair* of objects that describes their mutual similarity. Conceptually, τ can be understood as a particular similarity function on two objects. For instance, as Fig. 1 illustrates, it can be the cosine similarity between the word vectors of two queries (lexical pairwise feature), or the number of co-clicks for two queries (co-click pairwise feature). Thus, there can exist as many edges between two objects as the number of pairwise features, such as between "canon" and "canon sd1000" in Fig. 1.

The strength of a relationship is encoded as the non-negative weight of its corresponding edge (o, o', τ), and is thus a function of the two objects o, o' and the pairwise feature τ, denoted as $W(o, o', \tau)$. $W(o, o', \tau)$ can be understood as the similarity between o and o' with respect to τ. By convention, $W(o, o', \tau) = 0$ iff the edge $(o, o', \tau) \notin R$. In this paper, we consider bi-directional, symmetric relationships, which have equal strengths in both directions, *i.e.*, $W(o, o', \tau) = W(o', o, \tau)$.

3.2 Regularization Model

Given K target classes $\{1, \ldots, K\}$, each object $o \in O$ has an underlying probability distribution over the K classes that captures the possible user intents. *E.g.*, in Fig. 1 the distribution for the query "canon" could be (digital-camera:0.3, inkjet-printer:0.2, ...), indicating that the user may be looking for a digital camera with probability 0.3, an inkjet printer with probability 0.2, *etc.*

As such a distribution is inherently latent, we undertake a "distribution over distributions" instead. Specifically, we model the class distribution of each object o using a Dirichlet prior $Dir(\boldsymbol{\alpha}_o)$, parameterized by a vector $\boldsymbol{\alpha}_o = (\boldsymbol{\alpha}_o[1], \ldots, \boldsymbol{\alpha}_o[K])$. Informally, it describes the distribution over all possible latent class distributions of o, given that each class $i \in \{1, \ldots, K\}$ has been observed $\boldsymbol{\alpha}_o[i] - 1$, potentially fractional, times. The Dirichlet prior also conveniently allows us to treat the total count of observations σ_o as the classification confidence—we are more confident in the prior if σ_o is larger. We call σ_o the prior *confidence* of o:

$$\sigma_o \triangleq \sum_{i=1}^{K} (\boldsymbol{\alpha}_o[i] - 1) = \sum_{i=1}^{K} \boldsymbol{\alpha}_o[i] - K \tag{1}$$

In particular, we only consider unimodal Dirichlet priors, *i.e.*, $\forall i \in \{1, \ldots, K\}, \boldsymbol{\alpha}_o[i] > 1$, such that $\sigma_o > 0$. As we shall describe later, such a prior may be derived from an initial object-level classifier or previous iterations of the regularization algorithm.

3.2.1 Regularization by Neighbors

Given a prior over possible class distributions for an object o, we can treat its neighbors as additional evidence to support its classification. In particular, we consider the neighbors as observation data, weighted by their similarity, from which we can compute a posterior distribution over possible class distributions for o. Alternatively, we can interpret this as a form of regularization in that the original belief on o is regularized by the belief on its neighbors.

Let $N(o)$ denote the set of neighbors of o on the graph. Specifically, for each neighbor $o' \in N(o)$, we can interpret its Dirichlet prior $Dir(\boldsymbol{\alpha}_{o'})$ as observing $\boldsymbol{\alpha}_{o'}[i] - 1$ instances of class i for each of the K classes when drawing $\sigma_{o'}$ instances from the underlying multinomial distribution for o'. Under the assumption that similar objects share similar labels and hence observations, we deem that each neighbor o' contributes to the classification of o an observation of $S(o, o')(\boldsymbol{\alpha}_{o'}[i] - 1)$ counts for each of the K classes, weighted by the similarity $S(o, o')$ between o and o'. As o can be connected to o' via multiple pairwise features, we define the similarity between the two objects as a linear combination of their similarity $W(o, o', \tau)$ across all pairwise features:

$$S(o, o') \triangleq \sum_{\tau} \lambda_\tau W(o, o', \tau), \tag{2}$$

where each λ_τ is a parameter specifying the overall influence of τ towards $S(o, o')$. We will learn these parameters in Sect. 3.3 to optimize the regularization.

Since the Dirichlet distribution is *conjugate* to the multinomial distribution, factoring in additional multinomial observations still results in a Dirichlet posterior. Specifically, given a Dirichlet prior $Dir(\boldsymbol{\alpha})$, the posterior distribution after observing $\boldsymbol{\beta}[i]$ counts in each class i is simply $Dir(\boldsymbol{\alpha} + \boldsymbol{\beta})$, where $\boldsymbol{\beta} = (\boldsymbol{\beta}[1], \ldots, \boldsymbol{\beta}[K])$. Hence, for an object o with prior $Dir(\boldsymbol{\alpha}_o)$, the additional weighted observation data from each neighbor o' would give us the Dirichlet posterior $Dir(\widehat{\boldsymbol{\alpha}}_o)$, where:

$$\widehat{\boldsymbol{\alpha}}_o = \boldsymbol{\alpha}_o + \sum_{o' \in N(o)} S(o, o')(\boldsymbol{\alpha}_{o'} - \mathbf{1}) \tag{3}$$

Similar to the prior confidence (Eq. 1), we can compute the posterior confidence for the object o as below:

$$\widehat{\sigma}_o = \sum_{i=1}^{K} \widehat{\boldsymbol{\alpha}}_o[i] - K = \sigma_o + \sum_{o' \in N(o)} S(o, o')\sigma_{o'} \tag{4}$$

However, since all observations are potentially noisy in practice, having more neighbors and hence more observation data does not necessarily imply a higher posterior confidence. As a heuristic, we normalize the observation data by the number of contributing objects weighted by their similarity, $S_o = 1 + \sum_{o' \in N(o)} S(o, o')$. This results in a normalized Dirichlet posterior $Dir(\widetilde{\boldsymbol{\alpha}}_o)$ and a normalized posterior confidence $\widetilde{\sigma}_o$ for each object o:

$$\widetilde{\boldsymbol{\alpha}}_o - \mathbf{1} = \frac{\widehat{\boldsymbol{\alpha}}_o - \mathbf{1}}{S_o} = \frac{1}{S_o}\left(\boldsymbol{\alpha}_o - \mathbf{1} + \sum_{o' \in N(o)} S(o, o')(\boldsymbol{\alpha}_{o'} - \mathbf{1})\right) \tag{5}$$

$$\widetilde{\sigma}_o = \frac{\widehat{\sigma}_o}{S_o} = \frac{1}{S_o}\left(\sigma_o + \sum_{o' \in N(o)} S(o, o')\sigma_{o'}\right) \tag{6}$$

Intuitively, this normalization step causes observations from neighbors of o to "adjust" our confidence in o (Eq. 6), rather than to raise it monotonically in the unnormalized version (Eq. 4).

We can understand Eq. 5 as a form of regularization. Informally, our original belief on object o, as prescribed by its prior $Dir(\boldsymbol{\alpha}_o)$, is regularized by its neighbors, which change our belief and produce the regularized distribution $Dir(\widetilde{\boldsymbol{\alpha}}_o)$. Note that in the special case that o has no neighbors ($N(o) = \emptyset$), Eq. 5 would give us $\widetilde{\boldsymbol{\alpha}}_o = \boldsymbol{\alpha}_o$, *i.e.*, no regularization is done.

Confidence-aware prediction. Given the regularized Dirichlet distribution $Dir(\widetilde{\boldsymbol{\alpha}}_o)$ for an object o, we can make a prediction on o by first choosing the most likely class distribution, *i.e.*, the mode distribution $\widetilde{\mathbf{m}}_o$ of $Dir(\widetilde{\boldsymbol{\alpha}}_o)$. Subsequently, we assign classes to o according to $\widetilde{\mathbf{m}}_o$, say, by taking the top k classes with the largest probabilities in $\widetilde{\mathbf{m}}_o$, or all classes above a given probability threshold. Since $\widetilde{\boldsymbol{\alpha}}_o[i] > 1, \forall i$, the mode has a closed form solution:

$$\widetilde{\mathbf{m}}_o = (\widetilde{\boldsymbol{\alpha}}_o - \mathbf{1})/\widetilde{\sigma}_o, \tag{7}$$

where $\widetilde{\sigma}_o$ is the normalized posterior confidence of o in Eq. 6.

Unlike previous regularization frameworks [20, 18, 2, 3], the prediction by $\widetilde{\mathbf{m}}_o$ is *confidence-aware*, as we will now demonstrate. Dividing both sides of Eq. 5 by $\widetilde{\sigma}_o$, we obtain:

$$\frac{\widetilde{\boldsymbol{\alpha}}_o - \mathbf{1}}{\widetilde{\sigma}_o} = \frac{\boldsymbol{\alpha}_o - \mathbf{1} + \sum_{o' \in N(o)} S(o,o')(\boldsymbol{\alpha}_{o'} - \mathbf{1})}{\widetilde{\sigma}_o S_o} \tag{8}$$

In Eq. 8, the left hand side is simply $\widetilde{\mathbf{m}}_o$. On the right hand side, $\boldsymbol{\alpha}_o - \mathbf{1} = \sigma_o \mathbf{m}_o, \forall o \in O$, where $\mathbf{m}_o$ is the mode distribution of $Dir(\boldsymbol{\alpha}_o)$. Moreover, taking Eq. 6 into consideration, Eq. 8 can be further expressed as:

$$\widetilde{\mathbf{m}}_o = \frac{\sigma_o \mathbf{m}_o + \sum_{o' \in N(o)} S(o,o')\sigma_{o'}\mathbf{m}_{o'}}{\sigma_o + \sum_{o' \in N(o)} S(o,o')\sigma_{o'}} \tag{9}$$

Similar to the harmonic method [20], Eq. 9 implies that the regularized mode $\widetilde{\mathbf{m}}_o$ is a weighted average of the original (unregularized) modes $\mathbf{m}_o$ and $\mathbf{m}_{o'}, \forall o' \in N(o)$. However, unlike [20], the weight on each neighbor o' is $S(o,o')\sigma_{o'}$, which factors in both the similarity $S(o,o')$ and the confidence $\sigma_{o'}$. Hence, our model is confidence-aware, *i.e.*, objects with higher confidence $\sigma_{o'}$ will be weighed more in the regularization. As motivated in Sect. 1, such a property is desirable, since an object with low confidence would potentially introduce noise and thus its influence should be downplayed. On the other hand, [20] and other regularization models [18, 2, 3] only assign weights according to the similarity between the objects, without utilizing the confidence.

Alternative interpretation. Our regularization (Eq. 5) is equivalent to the minimization of a cost function, the essence of many graph-based regularization frameworks [20, 18, 2, 3]. Specifically, we minimize the following function over $\widetilde{\boldsymbol{\alpha}}_o, \forall o \in O$:

$$\mathcal{E} = \frac{1}{2}\sum_{o \in O}\left(\|\widetilde{\boldsymbol{\alpha}}_o - \boldsymbol{\alpha}_o\|^2 + \sum_{o' \in N(o)} S(o,o')\|\widetilde{\boldsymbol{\alpha}}_o - \boldsymbol{\alpha}_{o'}\|^2\right), \tag{10}$$

where $\|\cdot\|$ is the Euclidean distance. Intuitively, the regularization should not change the original belief on o too much, *i.e.*, we want $Dir(\widetilde{\boldsymbol{\alpha}}_o)$ to be close to the prior $Dir(\boldsymbol{\alpha}_o)$. Additionally, $Dir(\widetilde{\boldsymbol{\alpha}}_o)$ should be close to the prior $Dir(\boldsymbol{\alpha}_{o'})$ of each neighbor $o' \in N(o)$, from which observations are contributed towards o.

To see the equivalence, we minimize $\mathcal{E}$ by setting its derivative with respect to $\widetilde{\boldsymbol{\alpha}}_o$ to zero:

$$\frac{\partial \mathcal{E}}{\partial \widetilde{\boldsymbol{\alpha}}_o} = \widetilde{\boldsymbol{\alpha}}_o - \boldsymbol{\alpha}_o + \sum_{o' \in N(o)} S(o,o')(\widetilde{\boldsymbol{\alpha}}_o - \boldsymbol{\alpha}_{o'}) = \mathbf{0}, \tag{11}$$

which is algebraically equivalent to Eq. 5.

3.2.2 Regularization by Indirectly Related Objects

In the above model (Eq. 5), an object o is only regularized by its neighbors on the graph. In most scenarios, indirectly related objects, such as "neighbors of neighbors", also provide additional evidence for o. The evidence from "neighbors of neighbors" can be captured by regularizing $Dir(\widetilde{\boldsymbol{\alpha}}_o)$ again. In general, we propose an iterative regularization algorithm, which can potentially model the evidence from any object with a path to o on the graph.

Iterative regularization. As our regularized distribution $Dir(\widetilde{\boldsymbol{\alpha}}_o)$ is also Dirichlet, we can feed it back as input for regularization again in the exact same way as we have done in Sect. 3.2.1, treating $Dir(\widetilde{\boldsymbol{\alpha}}_o)$ as the new Dirichlet prior for o. The process can be repeated for any number of iterations.

Let $\boldsymbol{\alpha}_o^{(0)} \triangleq \boldsymbol{\alpha}_o$ denote the Dirichlet parameter of the initial prior for an object o. Furthermore, let $\boldsymbol{\alpha}_o^{(t)}$ be the regularized Dirichlet parameter after t iterations, $\forall t > 0$. Similar to Eq. 5, it is easy to see that $\forall t \geq 1$,

$$\boldsymbol{\alpha}_o^{(t)} - \mathbf{1} = \frac{1}{S_o}\left(\boldsymbol{\alpha}_o^{(t-1)} - \mathbf{1} + \sum_{o' \in N(o)} S(o,o')\left(\boldsymbol{\alpha}_{o'}^{(t-1)} - \mathbf{1}\right)\right) \tag{12}$$

As shown, we only need to know the Dirichlet parameters from the previous iteration to compute $\boldsymbol{\alpha}_o^{(t)}$. To start the iterative process, we bootstrap from $t = 0$ by specifying $\boldsymbol{\alpha}_o^{(0)}, \forall o$, for the initial prior of each object. How they are initialized is an orthogonal issue to our regularization framework. Different strategies may be applied based on the application, as we will discuss in Sect. 4.

Finally, we make predictions using the mode distributions after t iterations of regularization, $\mathbf{m}_o^{(t)} = (\boldsymbol{\alpha}_o^{(t)} - \mathbf{1})/\sigma_o^{(t)}$, in the same manner as discussed in Sect. 3.2.1.

Number of iterations. The iterative regularization process can be interpreted as a form of information propagation on the graph. In each iteration, information is propagated from each object to its neighbors. The propagation is repeated such that indirectly related objects are also affected through longer-range dependencies.

The total number of iterations T is an important parameter to decide, with similar effects as the length parameter in random walks [17, 7]. In particular, when $T = 1$, it reduces to a simple weighted voting by neighbors. But as $T \to \infty$, $\boldsymbol{\alpha}_o^{(T)}$ becomes the same for every object o in each connected component of the graph, similar to what have been discussed in [17, 7].

Intuitively, if T is too small, only very short-range dependencies can be captured, ignoring potentially useful long-range ones. On the other hand, if T is too large, all dependencies regardless of its range will be captured, making the result less discriminative. Thus, an optimal T would ensure a desirable trade-off between short and long-range dependencies. While the optimal T can be selected either manually [7] or heuristically [17], we propose to learn it in the next subsection.

3.3 Parameters Learning

In addition to the total number of iterations T, each pairwise feature τ has a parameter λ_τ that specifies the overall influence of τ towards the similarity between two objects (Eq. 2). Thus, our parameters are T and Λ, where $\Lambda \triangleq \{\lambda_\tau : \forall \tau\}$.

We propose to learn the parameters with a heuristic objective function defined on only T and Λ. Since $|\Lambda|$ is generally small as restricted by the number of possible pairwise features between objects (in our experiments $|\Lambda| = 2$), only a few labeled objects are needed for learning. In comparison, traditional supervised classification techniques train a model for each target class. In real-world

applications where the number of target classes is large ($K = 2043$ in our experiments), an enormous amount of training data would be required, as we would expect at least a few labels for each class. On the other hand, with our heuristic objective function, we require far fewer than K to achieve good classification accuracy.

Our first attempt defines a global error function to optimize the parameters. However, due to computational challenges, we subsequently propose a local error function for the dynamic selection of parameters in each iteration. As the parameters are chosen to minimize the error functions, any existing numerical optimization technique can be used. In particular, we apply Powell's method [14] to solve the optimization problem.

3.3.1 Global Optimization

Given a labeled training set $O_L \subset O$, and an initialization for the Dirichlet parameters $\Omega = \{\boldsymbol{\alpha}_o^{(0)} : o \in O\}$ at $t = 0$, we can define an error function as the minimization objective:

$$G_{\text{err}}(O_L, \Omega, \Lambda, T) = \frac{1}{|O_L|} \sum_{o \in O_L} \left\| \mathbf{m}_o^{(T|\Omega,\Lambda)} - \mathbf{u}_o^* \right\|^2, \quad (13)$$

where $\mathbf{m}_o^{(T|\Omega,\Lambda)}$ denotes the mode after T iterations of regularization (using Eq. 12), with the initialization Ω and parameters Λ. $\mathbf{u}_o^*$ is the "gold standard" distribution derived from the labels of o. $\| \cdot \|$ is the Euclidean distance. (Note that alternative distance measures such as KL-divergence may also be used.) We select parameters that minimize this global error, *i.e.*, $\{\Lambda^*, T^*\} = \arg\min_{\Lambda,T} G_{\text{err}}(O_L, \Omega, \Lambda, T)$.

When $T = 1$, $\mathbf{m}_o^{(T|\Omega,\Lambda)}$ can be computed for all $o \in O_L$ using only the neighbors of objects in the labeled training set O_L. When $T = 2$, we need to consider "neighbors of neighbors" as well. For larger T's, we potentially need to compute iteratively over the entire graph of labeled and unlabeled objects. This makes the computation of the error function very costly. As many optimization techniques require repeated computation of $G_{\text{err}}(\cdot)$ for different values of Λ and T, such an error function is computationally infeasible.

3.3.2 Iterative Optimization

As an efficient alternative to minimizing the global error, we propose an algorithm where we regularize the Dirichlet parameters $\boldsymbol{\alpha}_o^{(t)}$ (Eq. 12) and update the parameters Λ in alternating fashion. In each iteration, we regularize the classification using the Λ learned from the previous iteration. As the Dirichlet distribution on every object changes after regularization, we update Λ by minimizing the error function for the next iteration. This is similar in spirit to the Expectation-Maximization (EM) algorithm. However, unlike EM which maximizes the likelihood function of the parameters, we are minimizing an error function. Our approach involves the following two steps in each iteration $t, \forall t \geq 0$:

(1) **Regularization step.** If $t = 0$, initialize $\boldsymbol{\alpha}_o^{(0)}, \forall o \in O$, according to the given $\Omega^{(0)} = \{\boldsymbol{\alpha}_o^{(0)} : \forall o \in O\}$. If $t > 0$, compute $\Omega^{(t)} = \{\boldsymbol{\alpha}_o^{(t)} : \forall o \in O\}$ from $\Omega^{(t-1)}$ (Eq. 12), using parameters $\Lambda^{(t)}$. Note that $\Omega^{(t-1)}$ and $\Lambda^{(t)}$ are already computed from the previous iteration.

(2) **Minimization step.** To update Λ for the next iteration, we minimize a local error function over Λ:

$$L_{\text{err}}^{(t+1)}(O_L, \Lambda) \triangleq G_{\text{err}}(O_L, \Omega^{(t)}, \Lambda, 1), \quad (14)$$

which is the global error for $T = 1$ with initialization $\Omega^{(t)}$. We find $\Lambda^{(t+1)} = \arg\min_\Lambda L_{\text{err}}^{(t+1)}(O_L, \Lambda)$ for the next iteration. We continue the iteration until the minimum error converges.

Although in the regularization step, we still need to update the entire graph, it is acceptable as it is done only once per iteration. In contrast, to solve the optimization problem in each minimization step, the error function has to be computed numerous times. Thus it is important the error function involves only light-weight computation. Our local error function (Eq. 14) is defined using the global error function (Eq. 13) with $T = 1$, which can be easily computed using only the neighbors of the objects in the training set O_L, where typically $|O_L| \ll |O|$. *E.g.*, in most of our experiments, $|O_L|$ is less than 0.1% of $|O|$.

Another distinction in our iterative optimization is that we do not explicitly learn T, the maximum number of iterations. Instead, we terminate the iterations when the minimum local error $\min_\Lambda L_{\text{err}}^{(t)}(\cdot)$ converges. Although such convergence is guaranteed as established below, similar to EM, there is no guarantee that the global minimum $\min_{\Lambda,T} G_{\text{err}}(\cdot)$ can be achieved.

Proposition 1: As defined in Eq. 14, the sequence of local minimum errors $\{\min_\Lambda L_{\text{err}}^{(t)}(O_L, \Lambda)\}_{t=1}^{\infty}$ converges to a finite limit. ■

PROOF: Let Λ_0 denote that all parameters λ_τ are zero. Then,

$$\min_\Lambda L_{\text{err}}^{(t)}(O_L, \Lambda) \leq L_{\text{err}}^{(t)}(O_L, \Lambda_0) \stackrel{*}{=} \min_\Lambda L_{\text{err}}^{(t-1)}(O_L, \Lambda).$$

Note that $(*)$ holds since Λ_0 implies that no regularization is done—we get the same error as the minimum error of the previous iteration. Thus, the sequence is monotonically non-increasing. Additionally, the minimum error is bounded from below by zero. Any bounded monotone sequence converges to a finite limit. ■

4. APPLICATIONS IN IR

We now discuss the applications of the regularization framework on real-world classification tasks in information retrieval. We first formalize what components of the regularization framework must be realized in any application. Next, we showcase a specific application on query intent classification in the shopping domain, followed by brief discussions of applications on other tasks. Finally, we describe the target scenarios of our applications.

4.1 Realization of the Framework

Realizing our regularization framework requires a *vertex model* and one or more *edge models*.

Vertex model. As we see in Eq. 12, to enable iterative regularization, we need an initial model for $t = 0$, characterized by the Dirichlet parameters $\boldsymbol{\alpha}_o^{(0)}, \forall o \in O$. Since each object o is a vertex in the graph, we call this initialization a *vertex model*. As $\boldsymbol{\alpha}_o^{(0)} = \sigma_o^{(0)} \mathbf{m}_o^{(0)} + \mathbf{1}$, which entails both the mode distribution and the confidence, we can equivalently set $\boldsymbol{\alpha}_o^{(0)}$ by initializing $\mathbf{m}_o^{(0)}$ and $\sigma_o^{(0)}$ separately, as we do in Sect. 4.2.

Edge model. In Sect. 3, the strength of a relationship between two objects o, o' is abstracted into a weight function $W(o, o', \tau)$, where τ is the pairwise feature that induces the relationship. Since relationships are edges on the graph, we call the realization of the weight function $W(q, q', \tau)$ for a pairwise feature τ an *edge model*.

4.2 Example Applications

4.2.1 Query Intent Classification

We address the problem of query intent classification in the shopping domain. Given a query $q \in Q$ from a query log of an e-commerce website, and a set of K predefined product categories, the task is to map the query to a product category, with a limited

amount of labeled queries. Additional resources such as existing product metadata may also be leveraged. In the following, we present how we can realize the vertex and edge models for this task. Note that the set of queries Q in the given query log form the vertices on the graph instead of the generic objects O in Sect. 3.

Vertex models. Any conventional query classification algorithm can be used to initialize the mode $\mathbf{m}_q^{(0)}$ for each query $q \in Q$, as long as the output of the algorithm can be converted to a probability distribution over the K categories, which is taken as the most likely distribution, *i.e.*, the mode distribution $\mathbf{m}_q^{(0)}$. We first introduce a vertex model based on unigram language models, followed by possible alternatives.

Language Vertex Model. As input, we leverage existing product metadata from online shopping sites such as Bing Shopping[1]. Each product is associated with metadata that include attributes such as name, brand and description, as well as the category to which the product belongs. From this data, we build a unigram language model θ_i for each category $i \in \{1, \ldots, K\}$ based on all observed words from the attribute values of the products in that category, weighted by product popularity. Given a query q, to estimate $\mathbf{m}_q^{(0)}$, we evaluate the query likelihood $p(q|\theta_i), \forall i$, and convert it to a probability distribution over categories by applying Bayes' rule: $p(\theta_i|q) \propto p(q|\theta_i)p(\theta_i)$, using the prior of the categories as estimated by the total popularity of all products in each category.

On the other hand, to initialize the confidence, we build a background model θ_b by considering all products in our metadata. The intuition is that queries with a lower background model likelihood are easier to classify, as a low likelihood means that the words in the query are rare and more likely to be identified as belonging to those few categories that contain these rare words, resulting in higher classification confidence. On the other hand, a high likelihood implies that the query words are very common and appear in many categories, leading to lower confidence classification. Empirical study shows that the probability of a correct classification of a query is fairly correlated with its negative log likelihood. Thus, we initialize the confidence as follows:

$$\sigma_q^{(0)} = -\log p(q|\theta_b) \tag{15}$$

The language vertex model only requires a weak supervision from the product metadata, which is available from shopping websites. No training labels for individual queries are needed.

Alternative Vertex Models. To initialize the mode distribution, we can also use supervised but substantially more accurate classifiers. The disadvantage of using a supervised classifier is that it requires a large amount of labeled training queries to achieve good classification accuracy. However, after applying our regularization framework, using the weakly supervised unigram models can achieve comparable results to using a well-trained fully supervised classifier for the vertex model, as we shall observe in the experiments.

Eq. 15 is only a simple heuristic for the purpose of initializing the distribution confidence. Other heuristics can be used, such as the query length (longer queries provide additional features to guide its classification), or the consistency of the outputs from multiple classifiers (queries with more consistent outputs are arguably easier to classify). However, developing more theoretical confidence estimation methods, such as those explored by works on query difficulty [6], is beyond the scope of this paper.

Edge models. Among many possible pairwise features between queries, we focus on the lexical (τ_{lex}) and co-click (τ_{click}) features.

[1]http://www.bing.com/shopping

Lexical Edge Model. Given two queries q and q' from a query log, we can view each as a set of words, denoted by $\phi(q)$ and $\phi(q')$, respectively. We can then define the lexical edge model as follows:

$$W(q, q', \tau_{\text{lex}}) = \begin{cases} 1 & \phi(q) \subseteq \phi(q') \text{ or } \phi(q') \subseteq \phi(q) \\ 0 & \text{else} \end{cases} \tag{16}$$

This edge model is a symmetric, binary similarity measure: we draw an edge between two queries if one of them contains all words in the other. *E.g.*, $W(\text{“canon”}, \text{“canon camera”}, \tau_{\text{lex}}) = 1$, whereas $W(\text{“canon printer”}, \text{“canon camera”}, \tau_{\text{lex}}) = 0$.

Although this edge model is simple, it is more effective in preliminary experiments than other more sophisticated models such as cosine similarity. One possible explanation is that our binary similarity results in less noise than cosine similarity.

Co-click Edge Model. We can also establish relationships between queries using co-click pairwise features. Intuitively, if two queries have more clicks that land on product pages belonging to the same category, they are more closely related. In many cases, we may only be able to deduce that two clicks lead to the same category (*e.g.*, they land on the same URL, or URLs with the same prefix such as example.com/digital_camera/...), but we may not know the actual category, or how to map the vendor category to ours. Formally, we define the co-click edge model as follows, adapted from the co-citation/click measures in [19, 10]:

$$W(q, q', \tau_{\text{click}}) = \log\left(1 + \sum_c \#(q, c)\#(q', c)\frac{N}{N_c}\right) \tag{17}$$

In this edge model, each c is a category, which could be a “virtual category” with no direct correspondence to a target category, for reasons discussed above. $\forall q \in Q$, $\#(q, c)$ denotes the number of clicks associated with query q that leads to category c. N is the total number of clicks across all queries, whereas N_c is the number of total clicks that lead to category c. Thus, N/N_c has an effect similar to inverse document frequency, *i.e.*, more popular categories contribute less to the similarity between two queries. We also use a logarithm function to model a sublinear growth of the similarity with respect to the number of clicks.

Additional Edge Models. The following pairwise features can also be considered, although they are not used in our experiments.

- User sessions. In the same user session, the second query is often a refinement of the first. Thus, whether two queries co-occur in the same session and the frequency of such co-occurrence can be another pairwise feature.
- Search results. On an existing e-commerce system, a query can retrieve a set of related products (*i.e.*, the search results). A pairwise feature between two queries could be the similarity of their search results. Alternatively, search results from a generic search engine can also be used, where similarity between retrieved pages can be measured instead.

4.2.2 Other Applications

We briefly discuss how we can realize our regularization framework on other applications in IR. As discussed in Sect. 4.2.1, the vertex model is often easy to realize using an existing classifier for the initial mode distribution and various heuristics for the initial confidence. Hence, for the following applications, we only discuss the choice of the pairwise features for their edge models, which are often neglected in conventional classification.

Offer classification. Given a product offer from a vendor, which is presented as a list of attribute-value pairs (*e.g.*, product name, brand, image, description, as well as product-specific attributes such

as focal length for cameras), the task is to map it to our predefined categories. (Imagine we are an e-commerce website receiving feeds on offers from different vendors, whose attribute schemas and category taxonomies vary.) The following pairwise features could be used: (i) the graphical similarity between two product images; (ii) the overlap of product-specific attribute names, since products from the same category often share similar attribute names even across vendors; (iii) lexical similarity between the names and descriptions of two products.

Text classification. This is a classic task in IR that assigns each document to a class. With additional resources such as access statistics, social media and embedded hyperlinks, we suggest the following pairwise features: (i) co-readership, the number of readers who accessed both of the two documents within some time window; (ii) social tagging, the distributional similarity of the user tags on social sharing sites; (iii) hyperlinks in the documents, where similar documents likely point to similar pages; (iv) lexical similarity in the contents of two documents.

4.3 Target Application Scenarios

As our regularization framework requires multiple iterations over the entire graph $G = (O, E)$, it is not intended for online classification. Instead, it is designed for offline computation, where the precomputed predictions for each object in O is stored in an index. Given a seen object o (*i.e.*, $o \in O$), its precomputed prediction is directly fetched from the index. If o is previously unseen (*i.e.*, $o \notin O$), we can use the vertex model to classify it on-the-fly. Alternatively, we may potentially look up o's neighbors on the graph using an inverted index, locality sensitive hash, or other means, and subsequently regularize the classification of o using its neighbors for one iteration online.

As an example, let us consider a real scenario involving a live search system on the task of query intent classification. In this situation, as time passes, the live system collects a growing amount of data, including queries and clickthroughs. Thus, it is ideal if our regularization framework can be applied periodically offline (say, on a daily or weekly basis), in order to leverage more data and update the precomputed index accordingly. As our experiments show (Sect. 5.3), our approach runs fast enough on a single machine to be computed daily for at least tens of millions of queries.

Alternatively, we can apply our framework and take objects with highly confident predictions as labeled training data. Using these automatically generated training data, we can iteratively train a supervised classifier, extending the approach described in [12].

5. EXPERIMENTS

To analyze the performance of our regularization framework, we conduct extensive experiments on our showcase application of query intent classification, using a large scale real-world dataset.

The objective of our experiments is to validate that the proposed framework can utilize heterogeneous pairwise features as well as classification confidence to improve the performance, especially in a scenario with limited training data. This objective cannot be accomplished by comparing our approach with existing state-of-the-art techniques for query intent classification. Instead, we choose different schemes of our framework as the baselines (*e.g.*, using the same framework, but we assume only one kind of pairwise feature or uniform confidence). Note that in each iteration, our framework makes a similar form of updating on the predictions (Eq. 9) as the well-known graph regularization method Harmonic [20]. Hence, different schemes of our framework are well represented baselines for existing graph regularization techniques.

(a) "canon 35"

canon 35 mm lens
canon 35 f 2
35 mm wide angle 1.4 canon lens

(b) "hp laptop hard drive"

hard drive 1tb
seagate harddrive
western digital 2tb external

Figure 2: Examples of (a) lexical and (b) co-click neighbors.

5.1 Experimental Setting

Dataset. We obtain a hierarchy of product categories from Bing Shopping. As our regularization framework targets generic classification problems, which may not have hierarchical categories as labels, we use only the most detailed 2043 leaf categories (*i.e.*, $K = 2043$), instead of taking advantage of the category taxonomy. Thus, our task is to predict the leaf category for each query.

In our experiments, we use a query log containing four million distinct queries. Some of these queries have been labeled by human judges, which we split into two disjoint subsets for training and testing. Specifically, the training set consists of 10K labeled queries for parameter learning, and the test set consists of over 10K labeled queries for evaluation. Although we reserve 10K labeled queries for training, we only use 1K of them in all experiments except in Sect. 5.2.4, where we vary the number of training queries.

We also use clickthrough data to build the co-click edge model. There are about 11 million clicks associated with about $1/4$ of all queries. The fact that most queries do not have any click motivates the necessity of heterogeneous pairwise features.

Vertex model. Unless otherwise stated, we use the *language vertex model* (Sect. 4.2.1) in all experiments, which is weakly supervised and requires no labeled training queries. However, to demonstrate the robustness of our framework, we also compare it with an alternative vertex model in Sect. 5.2.2.

Evaluation metrics. For each query, we sort the categories according to their predicted probabilities, from which we can apply the following metrics.

- Top-k accuracy: the fraction of test queries which contain a correct result within the top k hypothesized categories;
- Precision-recall plot: a plot of precision against recall while varying the prediction cut-off threshold.
- Optimal f-score: the best f-score achievable among all points in the precision-recall plot.
- Precision @ 0.5 recall: precision at a recall level of 0.5. In practical scenarios, higher precision is usually more important than higher recall—we would rather not predict than suggest a wrong category.

5.2 Accuracy Study

5.2.1 Case Study

We first present some illustrative results, providing insights on why our approach works.

Query 1: "canon 35". This query is misclassified as camcorder by the unigram language model, since both words in the query are not sufficiently discriminative. However, its lexical neighbors reveal some interesting information, as illustrated in Fig. 2(a). Given these neighbors, it is not surprising that our approach can correctly classify this query as camera-lens.

Query 2: "hp laptop hard drive". This query is misclassified as laptop by the unigram model, since the words "hp laptop" strongly suggest laptops. This time, its co-click neighbors, as shown in Fig. 2(b), allow our approach to correctly classify it as hard-drive.

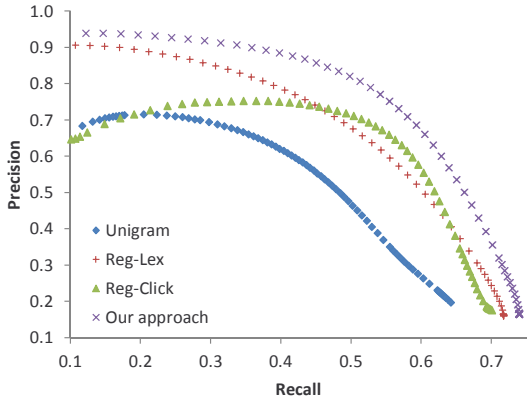

Figure 3: Precision-recall plot.

5.2.2 *Effect of Heterogeneous Pairwise Features*

We compare our approach to the vertex model and regularization baselines that only consider one kind of pairwise feature.

- Unigram: The unigram model used in our vertex model.
- Reg-Lex: The same framework as ours, but with only the lexical pairwise features (*i.e.*, the lexical edge model).
- Reg-Click: The same as Reg-Lex but with co-click pairwise features instead (*i.e.*, the co-click edge model).

Overall results. The results are reported in Fig. 3 and 4. While all regularization methods generally outperform the unigram model especially at high recall levels, we make two further observations.

First, neither Reg-Lex nor Reg-Click clearly dominates the other, which only uses lexical and co-click features, respectively. Normally we would expect Reg-Click to perform much better than Reg-Lex, since the former uses co-click pairwise features, a form of user feedback that may be more indicative than pairwise lexical features. For instance, many queries may be lexically similar yet unrelated (*e.g.*, "laptop" and "laptop bag"), or related but exhibit no lexical similarity (*e.g.*, "hp 3 in 1" and "canon printer"). Instead, our results can be explained by the sparsity of the click-through data, where only $1/4$ of the queries have received at least one click, but almost all queries have lexically similar neighbors.

Second, by using both pairwise features, our approach performs substantially better than Reg-Lex and Reg-Click. One reason is that a single kind of pairwise feature cannot cover all queries. A second reason is that the combined evidence of both pairwise features is more informative than any single one alone, thus directly improving queries that have both features and indirectly improving queries that connect to neighbors with both features.

Result breakdown. We further analyze the performance on two subsets of queries—queries with clicks and ones without—as shown in Fig. 5. As expected, Reg-Click performs well for queries with clicks, but has no effect on queries without clicks. On the other hand, Reg-Lex improves the accuracy for both subsets of queries. It is not surprising that our approach outperforms both Reg-Click and Reg-Lex for queries with clicks, since in this subset, both pairwise features improve the classification accuracy. However, we also outperforms all baselines for queries without clicks, although using co-click pairwise features alone has no effect on this subset. In this case, because classification evidence is iteratively propagated across all edge models in our approach, improvements to queries with clicks will influence their lexical neighbors, not all of which may have clicks. By combining heterogeneous pairwise features, we can extend the reach of each individual pairwise feature to influence and improve the performance of more queries via heterogeneous long-range dependencies.

Algorithm	Top-3 accuracy	Optimal f-score	Prec @ 0.5 recall
Unigram	0.663	0.564	0.465
Reg-Lex	0.735 (+10.8%)	0.660 (+17.1%)	0.676 (+45.4%)
Reg-Click	0.731 (+10.1%)	0.659 (+16.9%)	0.712 (+53.3%)
Our approach	**0.763** (+15.0%)	**0.728** (+29.2%)	**0.821** (+76.7%)

Figure 4: Accuracy comparison.

(a) Queries with clicks

Algorithm	Top-3 accuracy	Optimal f-score	Prec @ 0.5 recall
Unigram	0.721	0.609	0.593
Reg-Lex	0.795 (+10.2%)	0.704 (+15.6%)	0.780 (+31.6%)
Reg-Click	0.832 (+15.3%)	0.798 (+30.9%)	0.920 (+55.1%)
Our approach	**0.836** (+15.9%)	**0.801** (+31.5%)	**0.925** (+56.0%)

(b) Queries without clicks

Algorithm	Top-3 accuracy	Optimal f-score	Prec @ 0.5 recall
Unigram	0.573	0.495	0.238
Reg-Lex	0.641 (+11.9%)	0.583 (+17.9%)	0.445 (+87.2%)
Reg-Click	0.573 (+0.0%)	0.495 (+0.0%)	0.238 (+0.0%)
Our approach	**0.648** (+13.1%)	**0.592** (+19.7%)	**0.483** (+102.9%)

Figure 5: Accuracy comparison for queries with/without clicks.

Alternative vertex model. To demonstrate the robustness of our framework, we also evaluate our approach with an alternative *regression vertex model*. Specifically, we initialize the modes using a supervised logistic regression model (Regression) trained from a large number of labeled queries, with query words as features. To initialize the confidence, we use the query length as a heuristic, as empirical study suggests a correlation between longer queries and better classification accuracy of the regression model—longer queries contain more word features to guide the classification.

The results of using the regression vertex model are shown in Fig. 6. Similar to using the language vertex model as reported in Fig. 4, our approach outperforms the baselines by a clear margin, even though the improvements are not as substantial. We expect such results, since the regression model is a much stronger model than the unigram model, and is thus harder to improve upon. This is analogous to ensemble methods in machine learning, where it is generally easier to improve over weak learners than strong learners. Interestingly, despite initializing from a much stronger vertex model, the final results from regularizing the regression model is only slightly better than those from regularizing the unigram model. Thus, subsequent experiments use the language vertex model, which enjoys the additional benefit of being weakly supervised.

5.2.3 *Effect of Confidence*

Next, we investigate the effect of the confidence initialization on regularization. Specifically, we compare our heuristic method in Eq. 15 (Heuristic) with the following two strategies.

We call the first strategy NoConf, which uses the same framework as ours, except that no confidence information is applied. In other words, uniform confidence is used to initialize each query.

We label the second strategy Simulated, which also uses the same framework as ours. However, we initialize the confidence differently, since our confidence estimation (Eq. 15) is only a simple heuristic. To minimize the adverse effect of weak confidence

Algorithm	Top-3 accuracy	Optimal f-score	Prec @ 0.5 recall
Regression	0.719	0.680	0.739
Reg-Lex	0.746 (+3.9%)	0.697 (+2.5%)	0.773 (+4.6%)
Reg-Click	0.773 (+7.5%)	0.722 (+6.2%)	0.809 (+9.5%)
Our approach	**0.784** (+9.1%)	**0.729** (+7.3%)	**0.823** (+11.4%)

Figure 6: Accuracy comparison with regression vertex model.

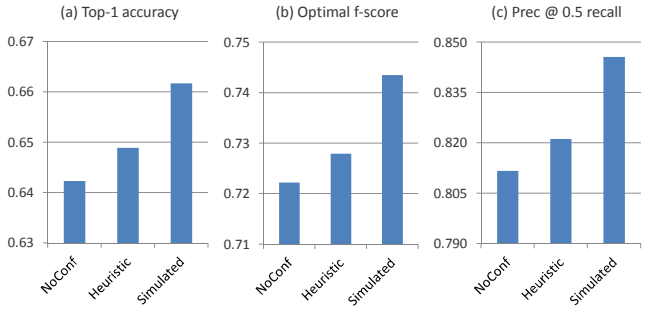

Figure 7: Accuracy with different confidence initializations.

estimation, we generate more reliable confidence values via a simulation. Specifically, for each labeled query, we sample a random value x from two Gaussian distributions. If the top-1 category predicted by the unigram model is correct, then $x \sim N(0.8, 0.1)$; otherwise, $x \sim N(0.2, 0.1)$. Subsequently, we initialize the confidence of the labeled query with x, clipped to the range $[0, 1]$. For each unlabeled query, we initialize its confidence to 0.512, which is the top-1 accuracy of the unigram model on the test queries.

The results of the different confidence initialization strategies are presented in Fig. 7. Although Heuristic only uses a simple heuristic (Eq. 15) to initialize the confidence, it performs better than NoConf, which is not confidence-aware. On the other hand, Simulated performs the best, since it initializes confidence in a more reliable way. Thus, with more sophisticated confidence estimation, we can expect even better results from our framework.

5.2.4 *Effect of Varying Amount of Data*

In the following experiments, we vary the amount of data in three aspects: the number of labeled training queries, total queries, and clickthroughs. It is important to study the effects of varying the amount of data, since in practical scenarios, data is collected from a live system gradually.

Number of training queries. Recall from Sect. 3.3, we require some labeled queries as training data in order to learn the parameters Λ. We examine the number of training queries needed to reach convergence on accuracy. Varying from 10 to 10K training queries, we record the corresponding performance in Fig. 8(a). The results reveal that with merely 10 training queries, we can already achieve a substantial improvement over the unigram baseline. Furthermore, with as few as 100 training queries, the performance has nearly converged. Considering that 100 is much smaller than the 2043 categories in this classification task, we demonstrate that our model requires very few training queries indeed. This is not surprising as we are not directly modeling the categories, which would require 2000+ training queries at the minimum, one for each category. Instead, we use the training queries only to learn Λ, the weights of pairwise features. Coupled with the weakly supervised language vertex model, our approach reaches convergence with very few training queries in total.

Number of total queries. The majority of our queries are unlabeled and not directly used for parameter learning. However, as we increase the number of unlabeled queries, we also improve the connectivity of the graph. To evaluate the effect, we randomly sample a subset of all queries, and apply our approach using this subset as the total set of queries. The sample size varies between 1% and 100% of all queries. To eliminate any differences due to labeled training queries, our samples include the same set of 1K training queries throughout. The results are presented in Fig. 8(b). Despite using the same set of training queries, the accuracy of our approach increases as the amount of total queries used increases. Although

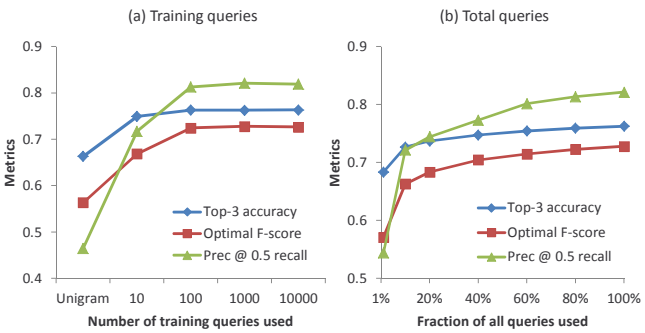

Figure 8: Accuracy versus amount of training/total queries.

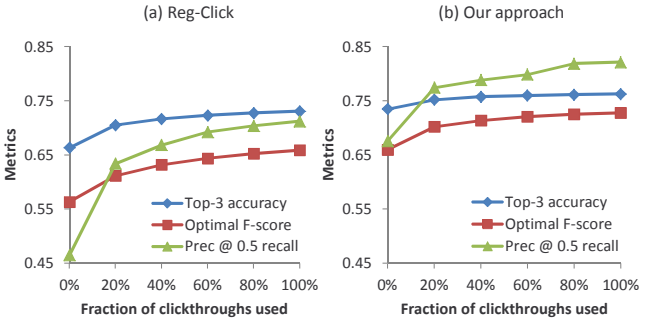

Figure 9: Accuracy versus the amount of clickthroughs.

most of the improvement is achieved when the amount of queries is initially increased from 1% to 20%, further increases result in smaller but continual improvements. From this trend, we expect that the accuracy can be further improved, but to a lesser extent, with an even larger query set.

Number of clickthroughs. Finally, we investigate the effect of varying the number of clickthroughs. Specifically, we randomly sample between 0% and 100% of clickthroughs from our query log, while using all queries and the same set of training queries. Fig. 9(a) and (b) illustrate the effect the amount of clickthroughs has on Reg-Click and our approach[2], respectively. In both methods, using a larger number of clickthroughs improves accuracy. Similar to the effect of increasing the number of total queries, the improvements diminish as more clickthroughs are used. Nevertheless, the improvements have not converged, suggesting that additional clickthrough data will further increase performance.

5.3 Efficiency Study

Although our regularization framework is designed to be applied offline periodically (Sect. 4.3), efficiency is still important. For instance, to apply it daily, it shall not take longer than a day. The following experiments evaluate the efficiency of our approach.

5.3.1 *Rate of Convergence*

As our approach is iterative, the number of iterations required to achieve convergence is crucial to efficiency. Our experiments show that convergence is fast, with 2 to 4 iterations in most settings. In particular, when using all available data, Fig. 10(a) illustrates the convergence of classification accuracy with a precision-recall plot. Iteration-0 corresponds to the initial vertex model. Regularization in Iteration-1 results in a larger improvement, whereas Iteration-2 shows smaller improvement. There is virtually no improvement from Iteration-3 (not shown in the figure), which indicates that the

[2]The number of clickthroughs does not affect the performance of Reg-Lex, which only utilizes lexical pairwise features.

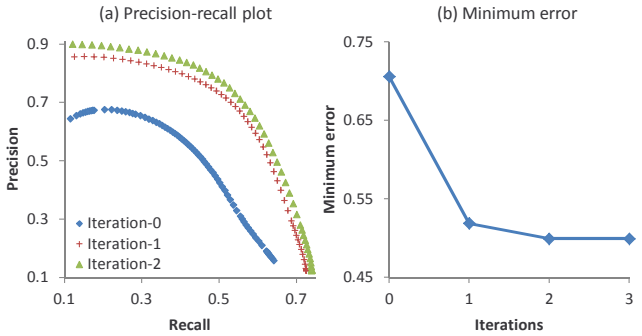

Figure 10: Convergence of (a) accuracy; (b) minimum error.

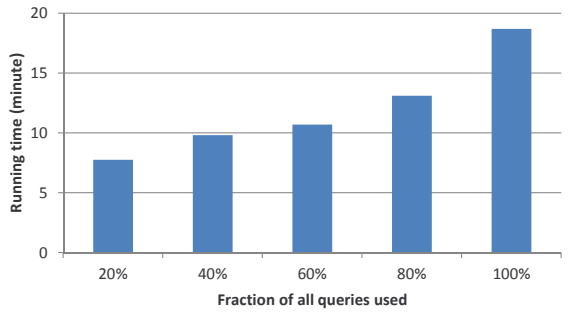

Figure 11: Execution time versus amount of total queries.

accuracy has converged. From another perspective, Fig. 10(b) illustrates the sequence of minimum errors in each minimization step (Eq. 14), which converges after only 3 iterations.

5.3.2 *Execution Time*

We start with a simple complexity analysis. As we scan all edges and vertices of the graph once in each iteration, the complexity is $O((E+V)T)$, where T is the number of iterations, E is the number of edges and V is the number of vertices (*i.e.*, queries). E potentially increases faster than V—the number of edges introduced by a new query potentially grows with the number of existing queries. Hence, the overall execution time would grow faster than a linear rate as we introduce more queries. Note that we do not consider the time to build the graph, which can be incrementally constructed whenever a new query is collected. Thus, it is unnecessary to build the graph from scratch every time.

In Fig. 11, we record the execution time on a single 8-core PC with 32GB memory. Consistent with our analysis above, the execution time grows slightly faster than a linear rate against the number of total queries. Nevertheless, when all four million queries are used, the execution time is just under 20 minutes. This is a reasonable time frame for our framework to be applied frequently (*e.g.*, on a daily basis), at least on the scale of tens of millions of queries. Furthermore, as the number of new queries within a fixed time interval tends to decrease over time, the total number of queries should grow sublinearly. Lastly, as Sect. 5.2.4 shows, including more queries brings in gradually less boost in accuracy. Thus, we do not expect to handle an arbitrarily large number of queries, since it would not achieve a noticeable benefit beyond a certain point.

6. CONCLUSION

In this paper, we proposed a Dirichlet-based graph regularization framework, driven by our key observations on heterogeneous pairwise features and confidence-aware classification. Building on the proposed framework, we discussed example realizations of several real-world IR tasks, particularly query intent classification in the shopping domain. Finally, we conducted extensive experiments on a large-scale real-world dataset to verify our observations. With limited training data, our approach effectively leverages heterogeneous pairwise features and classification confidence.

7. REFERENCES

[1] S. Beitzel, E. Jensen, O. Frieder, D. Lewis, A. Chowdhury, and A. Kolcz. Improving automatic query classification via semi-supervised learning. In *ICDM*, pages 42–49, 2005.

[2] M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. *Learning Theory*, pages 624–638, 2004.

[3] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. of Machine Learning Research*, 7:2399–2434, 2006.

[4] A. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang. Robust classification of rare queries using web knowledge. In *SIGIR*, pages 231–238, 2007.

[5] H. Cao, D. Hu, D. Shen, D. Jiang, J. Sun, E. Chen, and Q. Yang. Context-aware query classification. In *SIGIR*, pages 3–10, 2009.

[6] D. Carmel and E. Yom-Tov. Estimating the query difficulty for information retrieval. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 2(1):1–89, 2010.

[7] N. Craswell and M. Szummer. Random walks on the click graph. In *SIGIR*, pages 239–246, 2007.

[8] F. Diaz. Regularizing query-based retrieval scores. *Information Retrieval*, 10(6):531–562, 2007.

[9] F. Diaz. Improving relevance feedback in language modeling with score regularization. In *SIGIR*, pages 807–808, 2008.

[10] X. He and P. Jhala. Regularized query classification using search click information. *Pattern Recognition*, 41(7):2283–2288, 2008.

[11] M. Ji, J. Yan, S. Gu, J. Han, X. He, W. Zhang, and Z. Chen. Learning search tasks in queries and web pages via graph regularization. In *SIGIR*, pages 55–64, 2011.

[12] X. Li, Y. Wang, and A. Acero. Learning query intent from regularized click graphs. In *SIGIR*, pages 339–346, 2008.

[13] X. Li, Y. Wang, and A. Acero. Extracting structured information from user queries with semi-supervised conditional random fields. In *SIGIR*, pages 572–579, 2009.

[14] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes*. Cambridge Univ. Press, 3rd edition, 2007.

[15] D. Shen, Y. Li, X. Li, and D. Zhou. Product query classification. In *CIKM*, pages 741–750, 2009.

[16] D. Shen, J. Sun, Q. Yang, and Z. Chen. Building bridges for web query classification. In *SIGIR*, pages 131–138, 2006.

[17] M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In *NIPS*, volume 14, pages 945–952, 2001.

[18] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS*, pages 595–602, 2004.

[19] D. Zhou, B. Schölkopf, and T. Hofmann. Semi-supervised learning on directed graphs. *NIPS*, 17:1633–1640, 2005.

[20] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *ICML*, pages 912–919, 2003.