Computational Linguistics and Chinese Language Processing
Vol. 21, No. 1, June 2016, pp. 51-68

Enriching Cold Start Personalized Language Model Using Social Network Information[1]

Yu-Yang Huang[*], Rui Yan[+], Tsung-Ting Kuo[*], and Shou-De Lin[*]

Abstract

Personalized language models are useful in many applications, such as personalized search and personalized recommendation. Nevertheless, it is challenging to build a personalized language model for cold start users, in which the size of the training corpus of those users is too small to create a reasonably accurate and representative model. We introduce a generalized framework to enrich the personalized language models for cold start users. The cold start problem is solved with content written by friends on social network services. Our framework consists of a mixture language model, whose mixture weights are estimated with a factor graph. The factor graph is used to incorporate prior knowledge and heuristics to identify the most appropriate weights. The intrinsic and extrinsic experiments show significant improvement on cold start users.

Keywords: Language Model, Factor Graph, Social Network Analysis, Smoothing, Cold-Start Problem.

1. Introduction

Personalized language models on social network services are useful in many aspects (Xue *et al.*, 2009; Wen *et al.*, 2012; Clements, 2007). For instance, if the authorship of a document is in doubt, a language model may be used as a generative model to identify it. In this sense, a language model serves as a proxy of one's writing style. Furthermore, personalized language models can improve the quality of information retrieval and content-based recommendation

[1] A preliminary version of this paper appears as a short paper in Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 611–617, Baltimore, Maryland, USA, June 23-25 2014.

[*] Graduate Institute of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan.
E-mail: {r02922050, d97944007, sdlin}@csie.ntu.edu.tw

[+] Computer and Information Science Department, University of Pennsylvania, Philadelphia, U.S.A.
E-mail: ruiyan@seas.upenn.edu

systems, where documents or topics can be recommended based on the generative probabilities.

It is challenging, however, to build a personalized language model for users who just entered the system since the content posted by these users is insufficient to characterize them. These users are referred to as "cold start" users. Since the popularity of a system largely depends on whether new users will continue to stick to the system, it is even more critical to generate good recommendation results for new users. Therefore, this paper focuses on how to obtain a better personalized language model for cold start users.

To achieve the aforementioned purpose, the content written by friends on a social media site is exploited. It can be a reply or a post written by friends on sites like Facebook or Twitter. Here, the hypothesis is that friends, who usually share common interests, tend to discuss similar topics and use similar words than non-friends. In other words, we believe that a cold start user's language model can be enriched and better personalized by incorporating content written by friends.

Intuitively, a linear combination of language models can be used to mix the content written by friends into a user's language model. Nevertheless, it should be noticed that some documents are more relevant than others and should be weighted higher. To obtain better weights, some simple heuristics could be exploited. For example, we can measure the similarity or dissimilarity between a user language model and a document language model. In addition, documents being shared more frequently in a social network usually are considered to be more influential and should be expected to contribute more to the refinement of a user language model. More complex heuristics also can be derived. For instance, if we can categorize the documents and find that two documents are of the same category, then their weights should be more similar. The main challenge lies in how such heuristics can be utilized in a systematic manner to infer the weights of each document-level language model.

In this paper, we exploit the information on social network services in two ways. First, we impose the social dependency assumption via a finite mixture model. We model the true, albeit unknown, personalized language model as a combination of a biased user language model and a set of relevant document language models. Due to the noise inevitably contained in social media content, instead of using all available documents, we argue that, by properly specifying the set of relevant documents, a better personalized language model can be learnt. In other words, each user language model is enriched by a personalized collection of background documents.

Second, we propose a factor graph model to incorporate prior knowledge (*e.g.* the heuristics described above) into our model. Each mixture weight is represented by a random variable in the factor graph. An efficient algorithm is proposed to optimize the model and infer the marginal distribution of these variables. Useful information and heuristics are encoded into the model by a set of potential functions.

The main contributions of this work are summarized below.

- To solve the cold start problem encountered during the estimation of personalized language models, a generalized probabilistic framework is proposed. We incorporate social network information into user language models through the use of a factor graph model. An iterative optimization procedure utilizing perplexity is presented to learn the model parameters. To our knowledge, this is the first proposal to use a factor graph model to enrich language models.
- Perplexity is selected as an intrinsic evaluation, and experiment on authorship attribution is used as an extrinsic evaluation. The results show that our model yields significant improvements for cold start users.

2. Methodology

We describe how to construct and enrich a personalized language model in this section. In the first subsection, we propose a social-driven, personalized mixture language model. The original, poorly estimated user language model is enriched with a set of relevant document language models. In Section 2.2, a graphical model is presented to identify the mixture weights of each mixture component. The relative importance of each mixture component, *i.e.* document language model, is determined with the use of prior knowledge that comes from a social network. In Section 2.3, we describe how the model is optimized under a lack of labelled information.

2.1 Social-Driven Personalized Language Model

The language model of a collection of documents can be estimated by normalizing the counts of words in the entire collection (Zhai, 2008). To build a user language model, one naïve way is first to normalize word frequency $c(w, d)$ within each document, then average over all the documents in a user's document collection. The resulting unigram user language model is:

$$P_u(w) = \frac{1}{|\mathcal{D}_u|}\sum_{d\in\mathcal{D}_u}\frac{c(w,d)}{|d|} = \frac{1}{|\mathcal{D}_u|}\sum_{d\in\mathcal{D}_u}P_d(w) \tag{1}$$

where $P_d(w)$ is the language model of a particular document, $\mathcal{D}_u$ is the user's document collection, and $|\cdot|$ denotes the number of elements in a set. This formulation is basically an equal-weighted finite mixture model.

A simple yet effective way to smooth a language model is to linearly interpolate with a background language model (Chen & Goodman, 1996; Zhai & Lafferty, 2001). In the linear interpolation method, all background documents are treated equally. The entire document collection is added to the user language model $P_u(w)$ with the same interpolation coefficient. On social media, however, articles are often short and noisy. The user language models generated in this way are prone to overfitting. To obtain a better personalized user language model, we must take into consideration the complicated document-level correlations and dissimilarities, and this is where our idea was born.

Our main idea is to specify a set of relevant documents for the target user and enrich the user language model with these documents. Then, through the use of the information embedded in a social network, the relative importance of these documents is learnt. Suppose that the target user is u. Letting $\mathcal{D}_{rel}$ denote the content posted by people that are most relevant to u (*e.g.* friends on a social network), our idea can be concisely expressed as:

$$\widetilde{P_u}(w) = \lambda_u P_u(w) + \sum_{d \in \mathcal{D}_{rel}} \lambda_d P_d(w) \tag{2}$$

where λ_d is the mixture weight of the language model of document *d*, and $\lambda_u + \sum \lambda_d = 1$. Documents posted by irrelevant users are ignored as we believe the user language model can be personalized better by exploiting the social relationship in a more structured way. In our experiments, we choose the documents posted by friends as $\mathcal{D}_{rel}$.

Also note that we have made no assumption about how the “base” user language model $P_u(w)$ is built. In practice, it need not be models following maximum likelihood estimation, but any language model can be integrated into our framework to achieve a more refined model. Furthermore, any smoothing method can be applied to the language model without degrading the effectiveness.

2.2 Factor Graph Model

Now, we discuss how the mixture weights can be estimated. We introduce a *factor graph model* to make use of the diverse information on a social network. Factor graph (Kschischang *et al.*, 2006) is a bipartite graph consisting of a set of *random variables* and a set of *factors* that signifies the relationships among the variables. It is best suited to situations where the data is clearly of a relational nature (Wang *et al.*, 2012). The joint distribution of the variables is factored according to the graph structure. Using a factor graph model, we can incorporate the knowledge into the potential function for optimization and perform joint inference over documents.

A factor graph model is presented in Figure 1. As can be seen from Equation 2, there are $|\mathcal{D}_{rel}|$ unknown mixture weights to be estimated. For each mixture weight λ_d, we put a Bernoulli random variable y_d in the factor graph. The value $y_d = 1$ means that the document *d* should be included in the enriched personalized language model $\widetilde{P_u}$ of the target user. In this sense, a larger value of $P(y_d = 1)$ implies a higher mixture weight of *d*. In particular, we set λ_d to be proportional to $P(y_d = 1)$ in the final estimation.

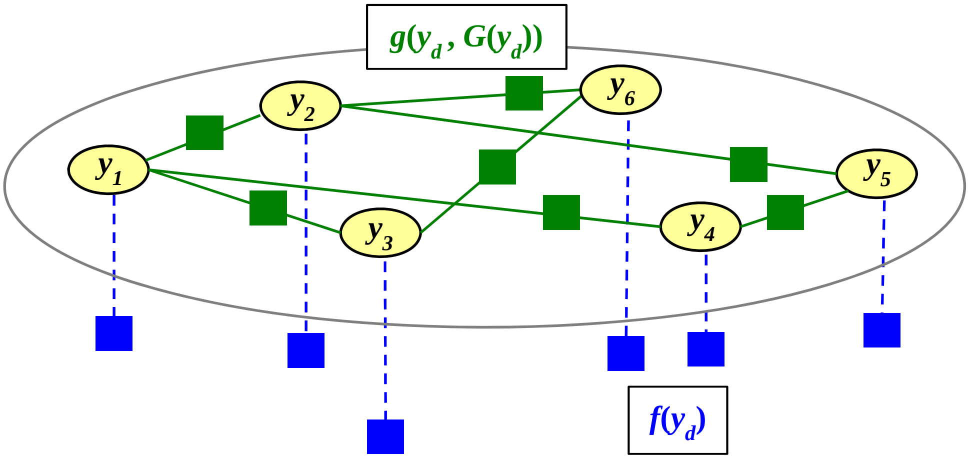

Figure 1. The proposed factor graph model.

Two kinds of potential functions are defined in the proposed factor graph model.

Local potential function *f(y_d)*. This potential function captures the local attributes of a random variable y_d. Suppose that the random variable y_d corresponds to a document *d*. The local potential function $f(y_d = 1)$ should take a larger value relative to $f(y_d = 0)$ if *d* is likely to contribute more significantly to the language model $\widetilde{P_u}$. The local potential function $f(y_d)$ is parameterized in a log-linear form:

$$f(y_d) = exp\{\alpha^T \mathbf{f}(y_d)\} \tag{3}$$

where $\mathbf{f} = \langle f_j \rangle^T$ is a vector of predefined feature functions and α is the parameter vector to be learnt. We assume that all feature functions $f_j(y_d)$ take a value of zero if $y_d = 0$. So, the larger the value of $\alpha^T \mathbf{f}(y_d = 1)$ is, the higher is the value $f(y_d = 1)$ relative to $f(y_d = 0)$. In other words, λ_d is (locally) believed to be higher.

In our experiment, we define the vector of feature functions as $\mathbf{f} = \langle f_{sim}, f_{oov}, f_{pop}, f_{cmf}, f_{af} \rangle^T$:

- *Similarity function f_{sim}.* The similarity between language models of the target user and a document should play an important role. We use cosine similarity between two unigram models in our experiments.
- *Document quality function f_{oov}.* The out-of-vocabulary (OOV) ratio is used to measure the quality of a document. It is defined as:

$$f_{oov} = 1 - \frac{|\{w: w \in d, w \notin V\}|}{|d|} \tag{4}$$

where *V* is the vocabulary set of the entire corpus, with stop words excluded.

- *Document popularity function f_{pop}.* To model the popularity of a document, this function is defined as the number of times the document *d* is shared.
- *Common friend function f_{cmf}.* It is defined as the number of common friends between the target user and the author of *d*.
- *Author friendship function f_{af}.* Assuming that documents posted by a user with more friends are more influential, this function is defined as the number of friends of the author of document *d*.

Pairwise potential function $g(y_d, G(y_d))$. For any two documents d_i and d_j, let the corresponding variables in the factor graph be y_i and y_j, respectively. The pairwise potential function defines the correlation of a variable y_i with another variable y_j. Similar to the local potential function, this function is parameterized as:

$$g(y_i, y_j) = exp\{\beta^T \mathbf{g}(y_i, y_j)\} \tag{5}$$

where **g** is a vector of feature functions indicating whether two variables are correlated. We assume that the two variables are not connected in the factor graph if $g(y_i, y_j) = 0$.

If we further denote the set of all variables linked to y_d as $G(y_d)$, then, for any variable y_d, we obtain the following result:

$$\begin{aligned} g(y_d, G(y_d)) &= \prod_{y \in G(y_d)} g(y_d, y) \\ &= exp\{\sum_{y \in G(y_d)} \beta^T \mathbf{g}(y_d, y)\} \end{aligned} \tag{6}$$

which is a function of y_d only. This expression will be used in the following equations.

We define the vector of feature functions $\mathbf{g} = \langle g_{rel}, g_{cat} \rangle^T$ as follows.

- *User relationship function g_{rel}.* We assume that two variables y_i and y_j are higher correlated if d_i and d_j are of the same author or the two authors are friends. The correlation should be even greater if the two documents are similar. Letting $a(d)$ denote the author of a document *d* and $\mathcal{N}[u]$ denote the closed neighborhood of a user *u*, we define

$$g_{rel} = \begin{cases} sim(d_i, d_j) & \text{if } a(d_j) \in \mathcal{N}[a(d_i)] \\ 0 & \text{if } a(d_j) \notin \mathcal{N}[a(d_i)] \end{cases}. \tag{7}$$

The similarity between documents $sim(d_i, d_j)$ is measured by the cosine similarity between two unigram language models.

- *Co-category function g_{cat}.* For any two variables y_i and y_j, it is intuitive that the two variables would have a higher correlation if d_i and d_j are of the same category. Letting $c(d)$ denote the category of document *d*, we define

$$g_{cat} = \begin{cases} sim(d_i, d_j) & \text{if } c(d_i) = c(d_j) \\ 0 & \text{if } c(d_i) \neq c(d_j) \end{cases}. \tag{8}$$

The flexibility of the proposed framework lies in the following aspects.

- The factor graph model is adaptable. The feature functions are not restricted to the ones we have used and can be freely added or redesigned in order to properly model different datasets.
- The set of relevant documents can be changed. In our experiment, we used the documents posted by friends to enrich the language model. Nevertheless, this is not a requirement. Whenever appropriate, documents posted by friends of friends, or any arbitrary set of documents can be adapted to tackle this problem.
- As we mentioned at the end of Section 2.1, the "base" user language model can be already smoothed by any technique. Furthermore, the language models need not be unigram models. If a higher order n-gram model is more suitable, it can be used in our framework. For our particular dataset, however, we find that it gives no advantage to use higher order n-gram models.

2.3 Model Inference and Optimization

Let Y be the set of all random variables. The joint distribution encoded by the factor graph model is given by multiplying all potential functions:

$$P(Y) = \frac{1}{Z}\prod_{d\in\mathcal{D}_{rel}} f(y_d)\, g\big(y_d, G(y_d)\big) \tag{9}$$

where Z is a normalization term to ensure that the probability sums to one.

The desired marginal distribution $P(y_d)$ can be obtained by marginalizing all other variables. Under most circumstances, however, the factor graph is densely connected. This makes the exact inference intractable, and approximate inference is required. After obtaining the marginal probabilities $P(y_d)$ with the approximate inference algorithm, the mixture weights λ_d in Eq. 2 are estimated by normalizing the corresponding marginal probabilities $P(y_d)$ to satisfy the constraint $\lambda_u + \sum \lambda_d = 1$. The normalization can be written as:

$$\lambda_d = (1-\lambda_u)\frac{P(y_d)}{\sum_{d\in\mathcal{D}_{rel}} P(y_d)}. \tag{10}$$

It can be verified that the above equation leads to a valid probability distribution for our mixture model.

The proposed factor graph model has $|\alpha| + |\beta|$ parameters, where $|\beta|$ means the dimensionality of the vector β. Combining Equation 2 and Equation 10, it can be observed that the total number of parameters in the mixture model is reduced from $1 + |\mathcal{D}_{rel}|$ to $1 + |\alpha| + |\beta|$, lowering the risk of overfitting.

A factor graph is often optimized by gradient-based methods. Unfortunately, since the ground truth values of the mixture weights λ_d are not available, we are prohibited from using these approaches. Here, we propose a two-step iterative procedure to optimize our model with respect to the model perplexity on held-out data.

At first, all of the model parameters (*i.e.* α, β, λ_u) are initialized randomly. Then, we infer the marginal probabilities of the random variables. Given these marginal probabilities, we can evaluate the perplexity of the user language model on a held-out dataset and search for better parameters. This procedure is repeated until convergence. We have also tried to train the model by optimizing the accuracy of the authorship attribution task. Nevertheless, we find that models trained by optimizing the perplexity give better performance.

3. Experiment

In this section, we evaluate the performance of language model enrichment with both intrinsic (perplexity) and extrinsic (authorship attribution) metrics. We compare the enriched language model with the original base language model and three intuitive enrichment methods.

3.1 Dataset and Experiment Setup

We performed experiments on the *Twitter* dataset collected by Galuba *et al.* (2010). Twitter data have been used to verify models with different purposes (Lin *et al.*, 2011; Tan *et al.*, 2011). To emphasize the cold start scenario, we randomly selected 15 users with about 35 tweets and 70 friends as candidates for an authorship attribution task. Our corpus consists of 4322 tweets. All words with less than 5 occurrences were removed. Stop words and URLs also were removed, and all words were stemmed. We identified the 100 most frequent terms as categories. The size of the vocabulary set is 1377.

We randomly partitioned the tweets of each user into training, validation, and testing sets. The reported result is the average of 20 random splits. In all experiments, we varied the size of training data from 1% to 15%, and held the same number of tweets from each user as validation and testing data. The statistics of our dataset, given 15% training data, are shown in Table 1.

Loopy belief propagation (Murphy *et al.*, 1999) was used to obtain the marginal probabilities. Parameters were searched with the pattern search algorithm (Audet & Dennis, 2002). To not lose generality, we used the default configuration in all experiments.

Table 1. Dataset statistics.

# of	Max.	Min.	Avg.
Tweets	70	19	35.4
Friends	139	24	68.9
Variables	467	97	252.7
Edges	9216	231	3427.1

3.2 Baseline Methods

We compared our framework with three baseline methods. The first ("*Cosine*") is a straightforward implementation that sets all mixture weights λ_d to the cosine similarity between the probability mass vectors of the document and user unigram language models. The second ("*PS*") uses the pattern search algorithm to perform constrained optimization over the mixture weights. Satisfying the constraint $\lambda_u + \sum \lambda_d = 1$, the algorithm iteratively searches for the optimal mixture weights λ_u and λ_d to lower the perplexity on the validation data. As mentioned in Section 2.3, the main difference between this method and ours ("*FGM*") is that we reduce the search space of the parameters by the factor graph model. Furthermore, social network information is exploited in our framework, while the PS method performs a direct search over mixture weights, discarding valuable knowledge. The third ("*LR*") models the probability $P(y_d)$ with the logistic function, where the local feature functions of each node are regarded as the independent variables, *i.e.* $P(y_d) = 1/(1 + e^{-\alpha^T \mathbf{f}(y_d)})$. By comparing our model to this baseline method, we want to show that the pairwise connections between nodes are useful.

Different from other smoothing methods, which usually are mutually exclusive, any other smoothing methods can be easily merged into our framework. In Equation 2, the *base language model* $P_u(w)$ can be already smoothed by any technique before being plugged into our framework. Our framework then enriches the user language model with social network information. We selected four popular smoothing methods to demonstrate such an effect, namely additive smoothing, absolute smoothing (Ney *et al.*, 1995), Jelinek-Mercer (JM) smoothing (Jelinek & Mercer, 1980), and Dirichlet smoothing (MacKay & Peto, 1994). Except for additive smoothing, the other three smoothing methods were all based on interpolation with the background corpus. The results of using only the base model (*i.e.* setting $\lambda_d = 0$ in Eq. 2) are denoted as "*Base*" in the following tables.

3.3 Perplexity

As an intrinsic evaluation, for each tweet in the testing set, we computed the perplexity of it under the author's own language model. The idea is that a better personalized language model should assign higher probability to a tweet if and only if it is written by the user himself.

The perplexity of a single sentence is defined as:

$$PPL(w_1 \cdots w_N) = 2^{-\frac{1}{N}\sum_{i=1}^{N} \log_2 P(w_i)} \tag{11}$$

where $w_1 \cdots w_N$ is an unseen testing sentence. The overall perplexity on a collection of sentences is simply computed by concatenating them. A smaller value signifies better performance. The results are shown in Table 2, where the asterisks indicate a significant difference between the best score and the second best score by t-test at a significance level of

0.05.

Table 2. Testing set perplexity.

Train %	**Additive**					**Absolute**				
	Base	Cosine	PS	LR	FGM	Base	Cosine	PS	LR	FGM
1%	903.1	711.6	701.3	843.9	**534.8***	864.7	695.3	696.7	814.7	**540.4***
5%	818.6	622.9	681.0	784.1	**504.3***	776.2	608.1	672.9	748.9	**509.6***
10%	749.9	559.8	658.1	726.4	**479.1***	703.2	543.7	638.9	681.3	**483.6***
15%	696.3	517.2	633.7	683.9	**471.7***	647.0	498.7	613.8	630.7	**469.4***
Train %	**Jelinek-Mercer**					**Dirichlet**				
	Base	Cosine	PS	LR	FGM	Base	Cosine	PS	LR	FGM
1%	631.7	571.3	640.6	615.7	**533.1***	632.6	571.6	635.8	616.2	**532.3***
5%	588.9	524.5	602.2	574.9	**505.3***	589.8	525.0	602.1	575.6	**503.0***
10%	553.0	489.9	570.2	545.4	**478.7***	554.0	490.7	581.4	546.2	**477.4***
15%	529.0	469.3	561.4	523.5	**465.6***	529.8	469.8	567.1	524.5	**466.2**

Our method significantly outperforms all of the methods in almost all settings. Furthermore, all methods gradually improve as more data are used to train the model. As expected, the advantage of our method is more apparent when data is sparse. Also, our method works much better than the "*LR*" method, which demonstrates the usefulness of the pairwise connections between documents in the factor graph.

We observe that the "*PS*" method takes a long time to converge and is prone to overfitting, likely because it has to search a few hundred parameters on average. It can also be observed from the table that, if the base language model is already smoothed by JM or Dirichlet smoothing, the "*PS*" method will only worsen, instead of enrich, the language model.

In terms of testing set perplexity, the "*Cosine*" method is the second best method to enrich a user's language model. The gap between the "*Cosine*" method and our method becomes smaller as more training data is available. When the base model is already smoothed by JM or Dirichlet smoothing and the data is less sparse (for example, the "15%" row), the "*Cosine*" method performs almost as good as our method. Nevertheless, when the base user language model $P_u(w)$ is sparse (*i.e.* the "1%" or "5%" rows), the similarity scores are not reliable and the performance of this method is restricted due to the bias coming from the base model.

3.4 Authorship Attribution

The authorship attribution task is chosen as the extrinsic evaluation metric. Given a sentence of unknown authorship, the task is to identify its author from a finite set of candidate users. For a thorough survey of recent works on this topic, see (Stamatatos, 2009).

Here, the goal is not about comparing with the state-of-the-art approaches in authorship attribution, but showing that typical applications of language model techniques can benefit from our framework.

To apply a personalized language model on this task, a naïve Bayes classifier is implemented (Peng *et al.*, 2004). The most probable author of a document d is the one whose personalized language model yields the highest probability. It is determined by the following equation:

$$u^* = argmax_u\{\prod_{w \in d} \widetilde{P_u}(w)\} \quad (12)$$

where we assume uniformity in the candidate users, and $\widetilde{P_u}(w)$ is as defined in Eq. 2.

Notice that we have used the unigram probability in Eq. 12, as in all following experiments. In fact, we also have conducted experiments with bigram models. Although the bigram model achieves lower perplexity as expected, we have observed a 15 to 20 percent decrease of the accuracy on the authorship attribution task, compared to the unigram model. This signifies that higher order n-gram models may not be suitable for the sparse data scenario. Similar arguments also have been given by Peng (2004). The results are shown in Table 3, where the asterisks indicate a significant difference between the best score and the second best score, by t-test at a significance level of 0.05.

Comparing the four "*Base*" columns, we find that additive smoothing performs about as well as the other three smoothing methods. That is, blindly interpolating with the entire background corpus does not fix the sparse data problem.

Similar to the result in Section 3.3, the "*Cosine*" method gets better (even better than our method, if the JM or Dirichlet smoothing is applied) when more data is available. A possible explanation is that the cosine similarity between the base user language model $P_u(w)$ and a document language model $P_d(w)$ is not reliable when $P_u(w)$ is estimated from a small corpus. In these cases, using the cosine similarity alone is not enough and it is better to rely less on this information and include other types of features. Our factor graph model can alleviate this problem by bringing more information into the language model; hence, it performs better than this baseline method under such circumstances. Conclusions that can be drawn from the results of the "*PS*" and "*LR*" methods are similar to those in the previous section.

Table 3. Accuracy (%) of authorship attribution.

Train %	Additive					Absolute				
	Base	Cosine	PS	LR	FGM	Base	Cosine	PS	LR	FGM
1%	0.5307	0.5853	0.6040	0.5700	**0.6587***	0.5120	0.5787	0.5947	0.5640	**0.6413***
5%	0.5967	0.6313	0.6267	0.6073	**0.6760***	0.5753	0.6233	0.6080	0.5947	**0.6687***
10%	0.6307	0.6640	0.6280	0.6433	**0.7053***	0.6153	0.6733	0.6220	0.6400	**0.6920**
15%	0.6513	0.6867	0.6400	0.6560	**0.7113***	0.6487	0.6967	0.6260	0.6560	**0.7200***
Train %	**Jelinek-Mercer**					**Dirichlet**				
	Base	Cosine	PS	LR	FGM	Base	Cosine	PS	LR	FGM
1%	0.5400	0.6127	0.6107	0.5967	**0.6560***	0.5233	0.6127	0.6093	0.5860	**0.6527***
5%	0.6147	0.6607	0.6420	0.6173	**0.6787***	0.6020	0.6567	0.6360	0.6187	**0.6787***
10%	0.6373	**0.6967**	0.6633	0.6513	0.6860	0.6293	0.6927	0.6507	0.6433	**0.6947**
15%	0.6553	**0.7094**	0.6520	0.6593	0.6987	0.6533	**0.7100***	0.6467	0.6567	0.6907

3.5 Feature Effectiveness

In this section, we evaluate the effectiveness of the feature functions mentioned in Section 2.2. There are five local feature functions $(f_{sim}, f_{oov}, f_{pop}, f_{cmf}, f_{af})$ and two pairwise feature functions (g_{cat}, g_{rel}) in our model. Including λ_u, there are a total of eight parameters for each user language model $\widetilde{P_u}(w)$. Since all features are standardized before training, we can inspect the feature weights learnt from data to determine the relative importance of them. To be focused on the cold start setting, we choose to analyze the case where only 1% of the data is used during model training. We will refer to the feature weights as, say w_{sim}, for simplicity.

First, we inspect the mixture weight λ_u for the base user language model. The higher this value is, the better is the original base user language model $P_u(w)$. We plot this value for all 15 users (the x-axis) in Figure 2. The different colors of the bars represent the different smoothing methods that were applied to the base model $P_u(w)$.

As can be observed from Figure 2, the users can be roughly categorized into three types:

- λ_u is low for all smoothing methods, *e.g*. Users 1, 2, and 4.
- λ_u is high for all smoothing methods, *e.g*. Users 5, 7, 11, and 13.
- λ_u is higher for the JM and Dirichlet smoothing, but is lower for the additive and absolute smoothing, *e.g*. Users 3, 6, and 9.

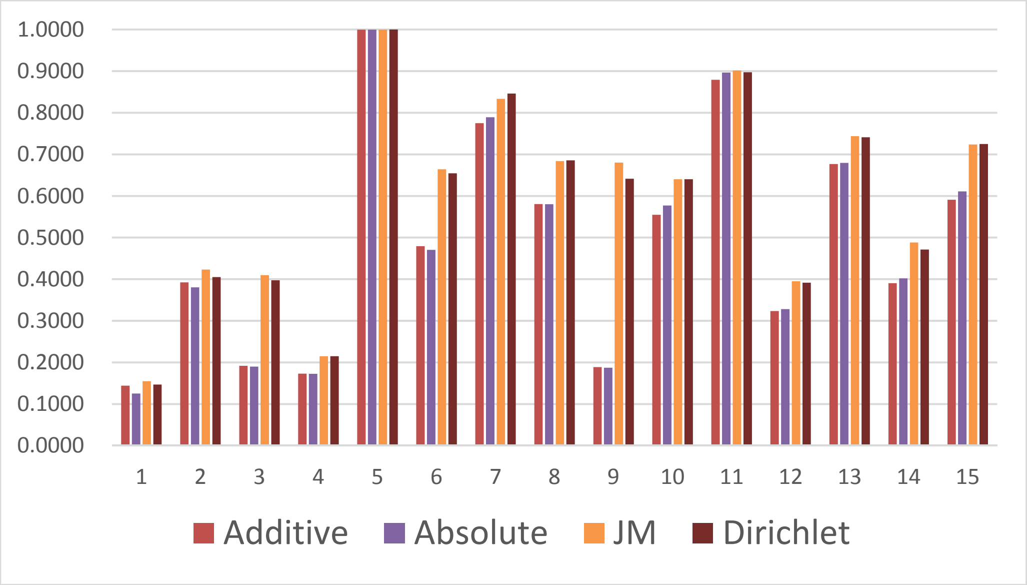

Figure 2. The mixture weight λ_u for each user.

We take Users 1, 5, and 6 as examples of the three types, respectively. Their feature weights are plotted in Figures 3, 4, and 5. It is worthwhile to mention that there is no user whose λ_u is lower for the JM and Dirichlet smoothing but is higher for the additive and absolute smoothing. This is consistent with the general idea that the JM and Dirichlet smoothing are better techniques than the other two.

Among all feature functions, f_{sim} generally has a higher feature weight w_{sim}. This is consistent with the experiments in Sections 3.3 and 3.4, where it has been shown that the cosine similarity may lead to an improved model performance. The weight w_{sim} has the highest value when λ_u is also high, as in Figure 4. This is reasonable because a higher λ_u signifies that the best model selected (with respect to the validation set perplexity, as described in Section 2.3) relies more on the base user language model $P_u(w)$. The similarity score computed from $P_u(w)$ should also be reliable.

By regarding each user as a sample, the correlation between λ_u and the feature weights can be computed. The results are shown in Table 4. There exists a relatively high positive correlation between λ_u and w_{sim}, as expected from the figures above. Also, for the pairwise features, the correlation coefficients are negative. This makes sense because, if the best model selected favors the base user LM more (*i.e.* a higher λ_u), then there should be no need to include the complicated pairwise features to enrich the language model. The negative correlation coefficients for w_{cmf} and w_{af} can be explained in the same way.

For some of the users whose λ_u is higher for JM and Dirichlet smoothing but lower for the other two, the feature weights exhibit a similar grouping. For User 6 (as shown in Figure 5), $w_{oov}, w_{pop}, w_{af}, w_{cat}$, and w_{rel} are lower if JM or Dirichlet smoothing is applied and higher if either of the other two is applied. This also explains the negative correlation between λ_u and w_{rel} (or w_{cat}).

Table 4. Correlation coefficient between λ_u and the feature weights.

Feature type	Local					Pairwise	
Feature weight	w_{sim}	w_{oov}	w_{pop}	w_{cmf}	w_{af}	w_{cat}	w_{rel}
ρ	0.3429	-0.0682	0.0112	-0.2316	-0.2166	-0.1583	-0.4920

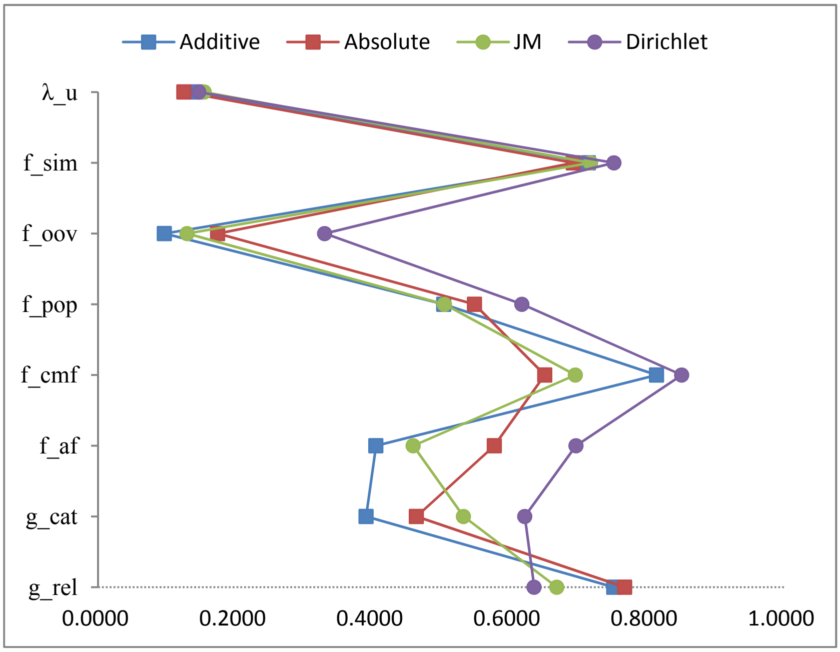

Figure 3. The feature weights for User 1. ($\lambda_u \approx 0.15$)

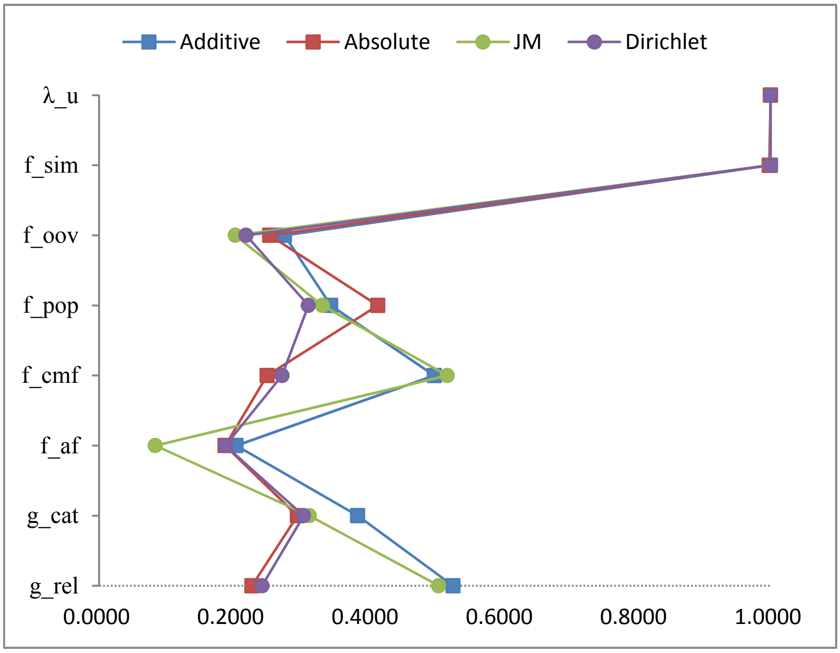

Figure 4. The feature weights for User 5. ($\lambda_u \approx 1$)

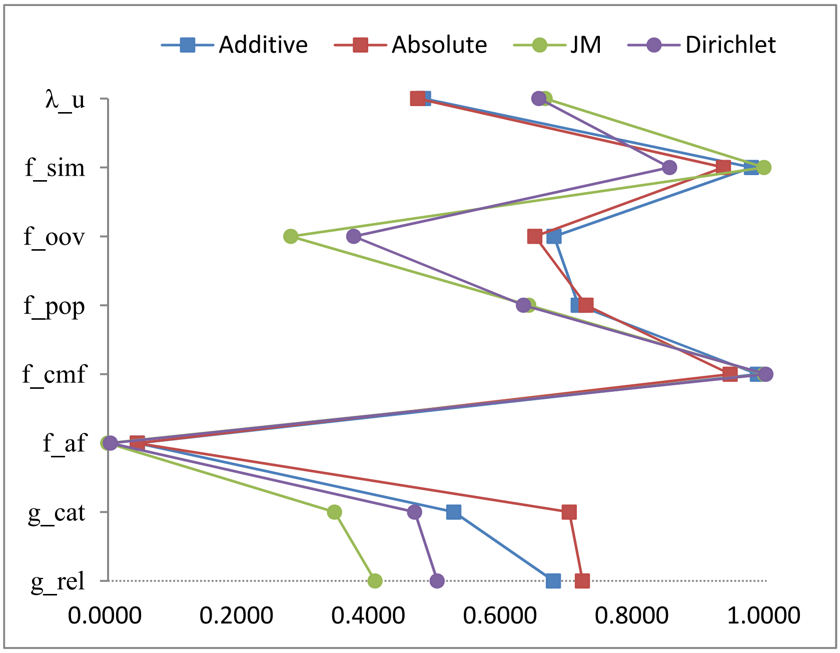

Figure 5. The feature weights for User 6. ($\lambda_u \approx 0.47$ for additive and absolute smoothing, and $\lambda_u \approx 0.65$ for JM and Dirichlet smoothing)

As has been mentioned, we want to demonstrate the usefulness of the pairwise feature functions (g_{rel} and g_{cat}) by comparing our model to the "*LR*" baseline. In fact, the usefulness of these feature functions can also be verified by observing that both w_{rel} and w_{cat} are greater than zero. By t-test at a significance level of 0.05, w_{cat} is significantly greater than zero for all users and all smoothing methods. Similarly, w_{rel} is significantly greater than zero for all except three of the ⟨user,smoothing method⟩ pairs (out of 60).

4. Related Work

Personalization has been studied for a long time in various textual related tasks. Personalized search is established by modeling user behavior when using search engines (Shen *et al*., 2005; Xue *et al*., 2009). A query language model could be also expanded based on personalized user modeling (Chirita *et al*., 2007). Personalization has also been modeled in many NLP tasks, such as summarization (Yan *et al*., 2011) and recommendation (Yan *et al*., 2012). Different from our purpose, these models do not aim at exploiting social media content to enrich a language model. Wen *et al*. (2012, 2013) combines user-level language models from a social network, but instead of focusing on the cold start problem, they try to improve the speech recognition performance using a mass amount of texts on a social network. On the other hand, our work explicitly models the more sophisticated document-level relationships using a probabilistic graphical model.

5. Conclusion

The advantage of our model is threefold. First, prior knowledge and heuristics about the social network can be adapted in a structured way through the use of a factor graph model. Second, by exploiting a well-studied graphical model, mature inference techniques can be applied in the optimization procedure, making it much more effective and efficient. Finally, different from most smoothing methods, which are mutually exclusive, any other smoothing method can be incorporated into our framework to be further enhanced. Using only 1% of the training corpus, our model can improve the perplexity of base models by as much as 40% and the accuracy of authorship attribution by at most 13%.

Reference

Audet, C., & Dennis Jr, J. E. (2002). Analysis of generalized pattern searches. *SIAM Journal on Optimization*, *13*(3), 889-903.

Chen, S. F., & Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics,* 310-318. Association for Computational Linguistics.

Chirita, P. A., Firan, C. S., & Nejdl, W. (2007). Personalized query expansion for the web. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval,* 7-14. ACM.

Clements, M. (2007). Personalization of Social Media. In *Proceedings of the BCS IRSG Symposium: Future Directions in Information Access 2007*.

Galuba, W., Aberer, K., Chakraborty, D., Despotovic, Z., & Kellerer, W. (2010). Outtweeting the twitterers - predicting information cascades in microblogs. In *Proceedings of the 3rd conference on Online social networks,* 3-3. USENIX Association.

Jelinek, F. (1980). Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, 381-397.

Kschischang, F. R., Frey, B. J., & Loeliger, H. A. (2001). Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, *47*(2), 498-519.

Lin, J., Snow, R., & Morgan, W. (2011). Smoothing techniques for adaptive online language models: topic tracking in tweet streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining,* 422-429. ACM.

MacKay, D. J., & Peto, L. C. B. (1995). A hierarchical Dirichlet language model. *Natural language engineering*, *1*(03), 289-308.

Murphy, K. P., Weiss, Y., & Jordan, M. I. (1999). Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, 467-475. Morgan Kaufmann Publishers Inc..

Ney, H., Essen, U., & Kneser, R. (1995). On the estimation ofsmall'probabilities by leaving-one-out. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *17*(12), 1202-1212.

Peng, F., Schuurmans, D., & Wang, S. (2004). Augmenting naive bayes classifiers with statistical language models. *Information Retrieval*, *7*(3-4), 317-345.

Shen, X., Tan, B., & Zhai, C. (2005). Implicit user modeling for personalized search. In *Proceedings of the 14th ACM international conference on Information and knowledge management,* 824-831. ACM.

Stamatatos, E. (2009). A survey of modern authorship attribution methods. *Journal of the American Society for information Science and Technology*, *60*(3), 538-556.

Tan, C., Lee, L., Tang, J., Jiang, L., Zhou, M., & Li, P. (2011). User-level sentiment analysis incorporating social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining,* 1397-1405. ACM.

Wang, Z., Li, J., Wang, Z., & Tang, J. (2012). Cross-lingual knowledge linking across wiki knowledge bases. In *Proceedings of the 21st international conference on World Wide Web,* 459-468. ACM.

Wen, T. H., Lee, H. Y., Chen, T. Y., & Lee, L. S. (2012). Personalized language modeling by crowd sourcing with social network data for voice access of cloud applications. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, 188-193.

Wen, T. H., Heidel, A., Lee, H. Y., Tsao, Y., & Lee, L. S. (2013). Recurrent neural network based language model personalization by social network crowdsourcing. In *INTERSPEECH*, 2703-2707.

Xue, G. R., Han, J., Yu, Y., & Yang, Q. (2009). User language model for collaborative personalized search. *ACM Transactions on Information Systems (TOIS)*, *27*(2), 11.

Yan, R., Nie, J. Y., & Li, X. (2011). Summarize what you are interested in: an optimization framework for interactive personalized summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 1342-1351. Association for Computational Linguistics.

Yan, R., Lapata, M., & Li, X. (2012). Tweet recommendation with graph co-ranking. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers*-Volume 1, 516-525. Association for Computational Linguistics.

Zhai, C. (2008). Statistical language models for information retrieval. *Synthesis Lectures on Human Language Technologies*, *1*(1), 1-141.

Zhai, C., & Lafferty, J. (2001). A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, 334-342. ACM.