Statistics items	Weibo dataset	Twitter dataset
# of posts	43,250	48,563
# of positive posts	32,060	34,624
# of negative posts	11,190	13,939
Size of vocabulary	30,171	23,181
# of sentiment words	4,495	2,457
# of topic words	22,758	17,899
# of syntactic units	314,712	213,590
# of sentiment units	40,775	42,650
# of topic units	164,529	98,987

Table 1: The statistics of the microblog datasets we used

they are assumed suitable personalized sentiment labels [Go *et al.*, 2009; Pak and Paroubek, 2010]. We selected 33 frequently used positive emoticons and 57 negative ones to derive ground-truth polarities. We crawled the microblog posts of 281 Sina Weibo users and 674 Twitter users using Weibo API[2] and Twitter API[3], respectively. We obtained 43,250 Weibo posts and 48,563 tweets, each containing one positive or negative emoticon. For Weibo corpus, we used an effective Chinese tokenizer[4] for word segmentation. For both datasets, Stanford POS tagger and neural network dependency parser were employed for POS tagging and dependency parsing, respectively (see Section 4.1). The Chinese sentimental words ontology bank[5] and NRC's EmoLex and MaxDiff Twitter Sentiment Lexicon[6] were used as sentiment lexicons for Weibo posts and tweets, respectively. The statistics about the two datasets are shown in Table 1.

We used 10-fold cross validation for evaluation, where 8 folds were for training, 1 for development and 1 for test. We implemented our models based on the generic factorization tool SVDFeature[7]. A common issue with microblog data is the imbalanced sentiment class distribution [Li *et al.*, 2011b; Liu *et al.*, 2013]. We re-sampled the training instances *for each user* to balance the proportion of positive and negative posts while keeping the development and test data are intact. We used geometric mean [Kubat and Matwin, 1997] as the metric for evaluation considering imbalanced nature of our dataset, which was defined as $G\text{-}mean = \sqrt{Sensitivity \times Specificity}$, where *Sensitivity* is the true positive rate and *Specificity* is the true negative rate.

5.2 Experiments and Results

Fixed parameters setting

We optimized the f parameter via validation on the development set by performing a grid search on all values of $10 * x$ with $x \in \{1, 2, ..., 10\}$. Basically the performance was not sensitive with respect to f, and we fixed $f = 60$ which is slightly better than other choices. We tuned λ using the development data and fixed it as $1.0e$-4. Since λ_U and λ_T are calculated by $\frac{\delta^2}{\delta_U^2}$ and $\frac{\delta^2}{\delta_T^2}$ (see formula 4), we set the two ratios to fixed values as $1.0e$-3 as we found that varying them just influenced the results slightly.

[2] open.weibo.com/

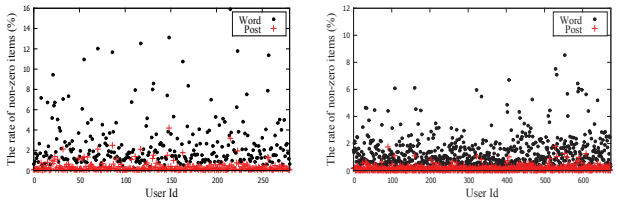
[3] https://dev.twitter.com/overview/api

[4] ictclas.nlpir.org/

[5] ir.dlut.edu.cn/EmotionOntologyDownload.aspx

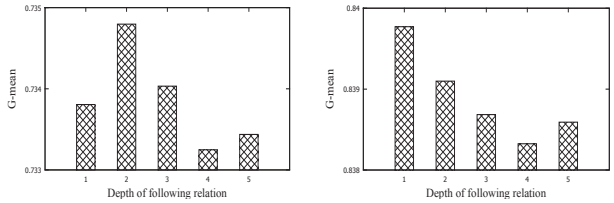
[6] www.saifmohammad.com/Lexicons/

[7] svdfeature.apexlab.org/wiki/Main_Page



(a) Weibo dataset (b) Twitter dataset

Figure 3: Decomposition alleviated sparsity of user data



(a) Weibo dataset (b) Twitter dataset

Figure 4: Influence of the depth of followers considered

Effects of decomposition and following relation

We examined the effect of word-level decomposition for dealing with feature sparsity. We compare the ratio of non-zero features of *each user* before decomposition, i.e., $\frac{\text{observed posts \#}}{\text{total posts \#}}$, and that after it, i.e., $\frac{\text{observed words \#}}{\text{total words \#}}$. The ratios are displayed in Figure 3. Overall, the non-zero rate of the personal data raised considerably from 0.36% to 2.9% for Weibo data and from 0.15% to 1.26% for Twitter data. This implies that the decomposition is helpful to alleviate the sparsity.

We also studied how the depth of following relation considered can influence the performance. Here we use the formula $\mathbf{C} = \sum_{i=1}^{n} \frac{1}{i}\mathbf{C}^i$ to calculate the entry values of connection matrix up to n level of depth in the following relation. The strength of connection is decayed by a factor of $\frac{1}{i}$ for the i-th level relation. Figure 4 shows that our model achieved best result when n=2 on Weibo and n=1 on Twitter. This is because the user relation on our Twitter data is much denser. The result seems to indicate that the first-level connection is sufficient for Twitter, but followers one more step deeper is helpful for Weibo, and more depth may bring too much noise. But overall the impact of depth does not appear very strong.

Comparison of different configurations

We compared the performance of five different settings: (1) **Basic:** direct application of LFM using the original user-post matrix; (2) **BOW:** our bag-of-words LFM without considering dependency relation and user following relation, which is equivalent to modifying the reviewer-product-review model [Li *et al.*, 2011a] to remove the product dimension not needed; (3) **Follow**: our model that considers following connection and uses only bag of words; (3) **Depend**: our model that considers dependency-based syntactic units but not using following relation; (5) **Full:** our fully configured model.

As shown in Table 2, **Basic** performs the worst on G-mean due to sparsity of the user-post matrix. Other models with the decomposition appear much better. This indicates that allevi-