Topic-specific Retweet Count Ranking for Weibo

Hangyu Mao, Yang Xiao, Yuan Wang, Jiakang Wang, and Zhen Xiao

School of EECS, Peking University

Outline

- ➢ Background
- ➢ Consideration and Design
- ➢ Evaluation
- ➢ Conclusion

Weibo

- **Weibo** is the biggest micro-blogging service in China
  - The counterpart of Twitter
  - https://weibo.com

- The user, tweet, and topic are **three major entities** in Weibo.
  - Users can generate tweets to express opinions and share experiences.

Weibo

- **Topic** is the group of all tweets sharing the same #topic name#
  - Topic has it own properties, e.g., topic category (society, sports, etc.), and topic information
  - Topics are ranked according to their popularity in the **Hot Topic List** as shown in the figure
    - If we click these topics, we can see the corresponding tweets.

Weibo

- **Tweets** are divided into Recommended Tweets and Common Tweets
  - There are usually 3-10 Recommended Tweets for each topic
  - Recommended Tweets have a large retweet count

Weibo

- **Users** are encouraged to read the representative tweets in the Hot Topic List rather than the scattered tweets in their timelines.

Related tweets

Unrelated tweets

Weibo

- Topic is becoming the core unit to organize tweets and users in Weibo

Fig. 1. The organization of topic, user and tweet in Weibo.

Topic is beneficial

- For users
  - Know the detailed information more easily
- For advertisers
  - Advertisements are more effective for proactive users
- For Weibo
  - Hot topic is the main source of page view (PV)
  - The PV of #Running Man# increases from 13.23 to 42.81 billion after Weibo introduces the "Super Topic" service last year
- For the Government
  - Better regulate public opinions about hot topics

The problem

To attract more PV and

to further get the above benefits of topic:

Find out *popular tweets* and

make them as the recommended tweets.

The problem

- Measure the popularity of a tweet by its retweet count.
- A larger retweet count usually means that more users have seen, and will see, the corresponding tweet and topic, and that we will further get more benefits [15, 25, 27, 28].
  - Researchers often use popular level as the synonym of retweet count [12, 14].

Necessity 1

- It is necessary to find out the popular tweets and put them in the first few pages of a topic.
  - There are too many tweets in a topic
  - Most users only look through the recommended tweets in the first few pages

Necessity 2

- It is necessary to automate the process of finding out popular tweets
  - Currently, the process can only be done manually
  - It is easy to miss the most popular tweets

Our work

- **Topic-specific** retweet prediction problem using **a ranking perspective**
  - Predict the retweet count ranking order for all tweets **belonging to the same topic**
  - Recommend the higher-ranked tweets **to each topic** rather than to each user

- As far as we know, many researchers have studied retweet prediction problem for personalized recommendation, but recommendation for topics **has not been** widely studied.
  - This work advances the study of topic-specific retweet prediction problem, which has not been well studied like traditional retweet prediction tasks as pointed out by [12, 13].

Outline

- Background
- Consideration and Design
- Evaluation
- Conclusion

Consideration 1

- How to deal with the large number of tweets
  - Hundreds of thousands of new tweets in one minute
- Candidate Tweet Filter
  - As we only care about the popular tweets when we do recommendation
  - Random forest with a dynamic filtering threshold
    - Tree-based methods have better interpretability

Table 1: Features used to build Candidate Tweet Generator. Those features are proved useful by (Cui et al. 2011; Jenders, Kasneci, and Naumann 2013; Luo et al. 2013).

Fea. Name	Fea. Meaning	Fea. Value
is_retweet	post or retweet	{0, 1}
post_hour	post hour	[0, 23]
at_count	how many @	[0, +inf)
tag_count	how many ##	[0, +inf)
followee_count	out-degree	[0, +inf)
follower_count	in-degree	[0, +inf)
tweet_count	tweets number	[0, +inf)

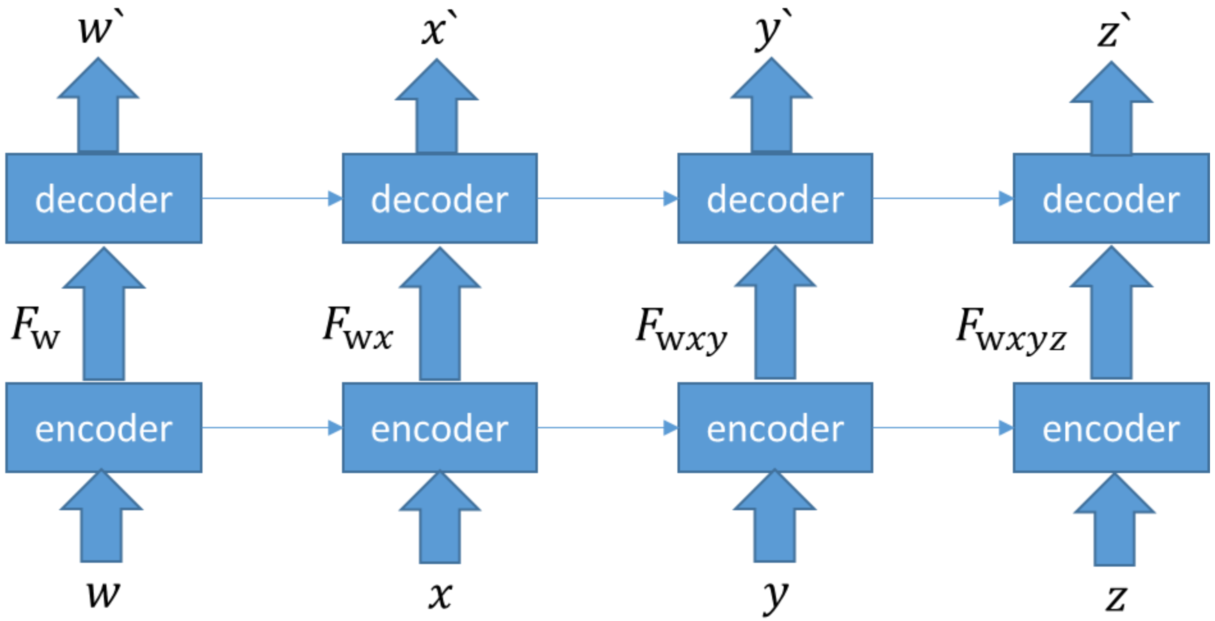
Consideration 2

- How to derive **effective features for tweets**. There are two challenges:
  - For a single tweet
    - The tweet text is short and text-length is random.
    - Researchers have shown that traditional BOW methods and Topic Model methods suffer from either sparseness or inefficiency for short texts [23].
  - For multiple tweets belonging to the same topic
    - Most of them share many words in topic-specific setting.
    - It is difficult to distinguish them.

- RNN is a better choice to deal with this kind of task
  - Summarize sequences with different length.
  - Distinguish sequences that have same words but in different orders [8].

Consideration 2

- We propose a LSTM-embedded autoencoder (LSTM-AE) for tweet feature generation
  - The input is the embedding of each word in the tweet text.
  - The output is the reconstruction of the input.
  - During training, we try to minimize mean square error between the inputs and outputs.

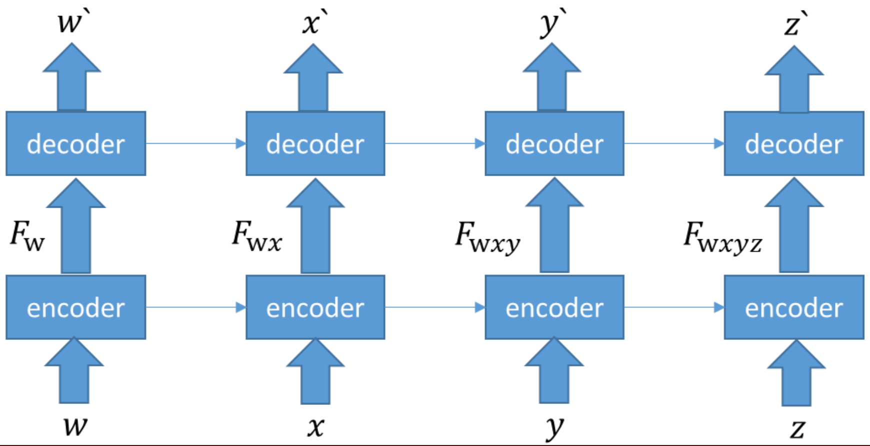


$$\min_{\theta} \frac{1}{N * T} \sum_{n=1}^{N} \sum_{t=1}^{T} (y_n^t - x_n^t)^2$$

tweet text “WXYZ”

Consideration 2

- The insight of this loss function: any different prefixes of the tweet text is a possible distinctive feature in our model.
  - The hidden state of LSTM can memory the history input information, so the output of the **encoder** can represent the features F_w, F_{wx}, F_{wxy}, and F_{wxyz} in some extent.
  - Based on these features, the **decoder** tries to reconstruct the inputs.
  - A well-trained LSTM-AE can not only distinguish the whole tweet text "WXYZ", but also distinguish any prefixes w, wx, wxy, wxyz of tweet text "WXYZ". This is why we say "any different prefixes of the tweet text is a possible distinctive feature in our model".



tweet text "WXYZ"

Consideration 2

- The above model is very suitable for topic-specific applications.
- In contrast, the existing models may not be suitable for this task.
  - The most popular encoder-decoder models try to maximize the following log-likelihood
    - Not autoencoder; Need Y for supervision

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} log P_{\theta}(y_n|x_n)$$

  - RNN-based **autoencoder**
    - Reconstruct the input sequence only based on the **final** embedding F_{WXYZ}.
    - F_{WXYZ} and F_{WAYZ} will be too similar to distinguish, especially in the topic-specific applications where tweets usually share many similar words.

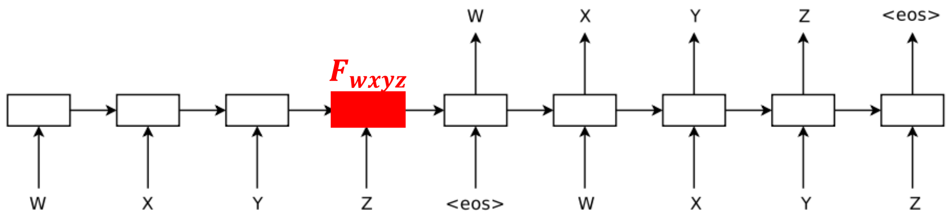


Figure 1: The sequence autoencoder for the sequence "WXYZ". The sequence autoencoder uses a recurrent network to read the input sequence in to the hidden state, which can then be used to reconstruct the original sequence.

Dai, Andrew M., and Quoc V. Le. "Semi-supervised sequence learning." NIPS 2015.

Consideration 3

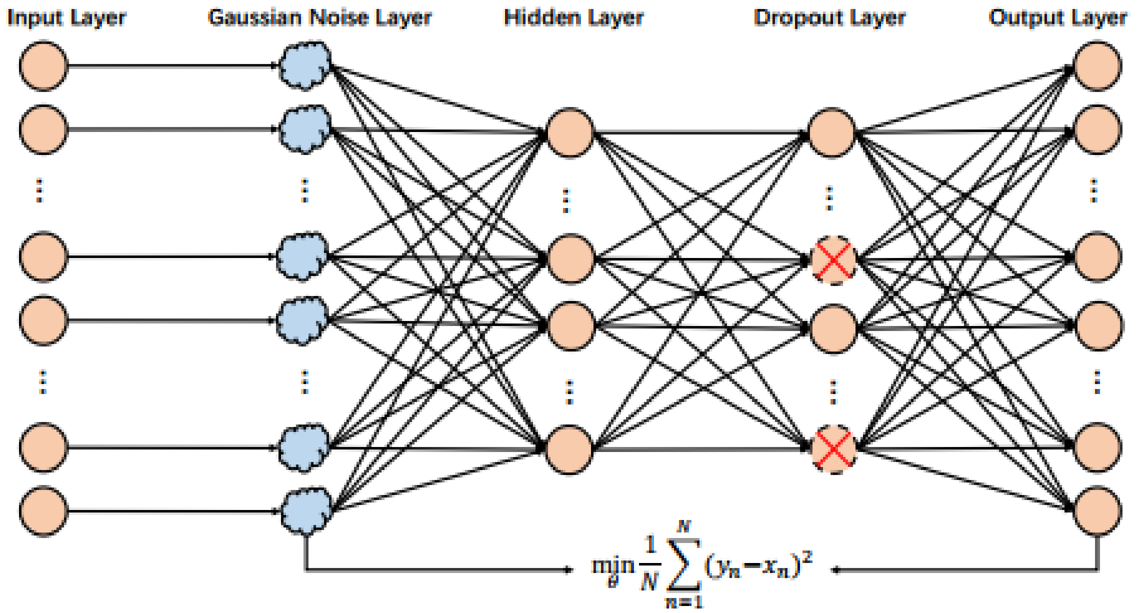
- How to fully catch the **meaning of topics**
  - In topic-specific setting, it is crucial to understand what the topics really talk about.
  - However, Weibo itself can provide little information for topics.
    - There are only a few words related to the topic information.

Consideration 3

- Leverage real-time news information from Toutiao, the most popular news recommendation platform in China, to enrich the meaning of topic
  - [16, 20] point out that micro-blogging service is more than social network but news media
  - Over 85% topics are headline news in the real world
  - Specifically, we use topic as keyword to search Toutiao, and only the returned news headlines are processed
    - News titles frame the interpretation of the article content and provide the most important information for readers [2, 10]

Consideration 3

- Denoising autoencoder (DAE) to translate news titles into topic features
  - On one hand, DAE can learn embedding features for topics so that both topic features and tweet features have a similar semantic space.
  - One the other hand, the number of topics is much less than that of tweets, so a Gaussian Noise Layer with different noise variances can create more training data for topics.



Framework

- LSTM-AE and DAE can generate features for tweets and topics, respectively.
- As for **user features**, we can crawl them from user database directly.
- After collecting all of the features about tweets, topics and users, we use LambdaMART [5, 7] to learn the desired ranking function.
- **T**opic-**S**pecific re**T**weet **R**anking (TSTR) framework summarizes our designs

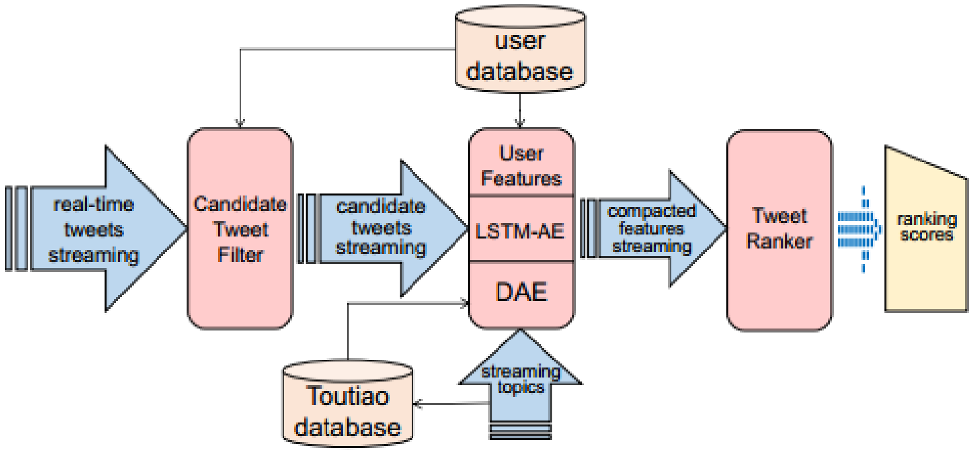


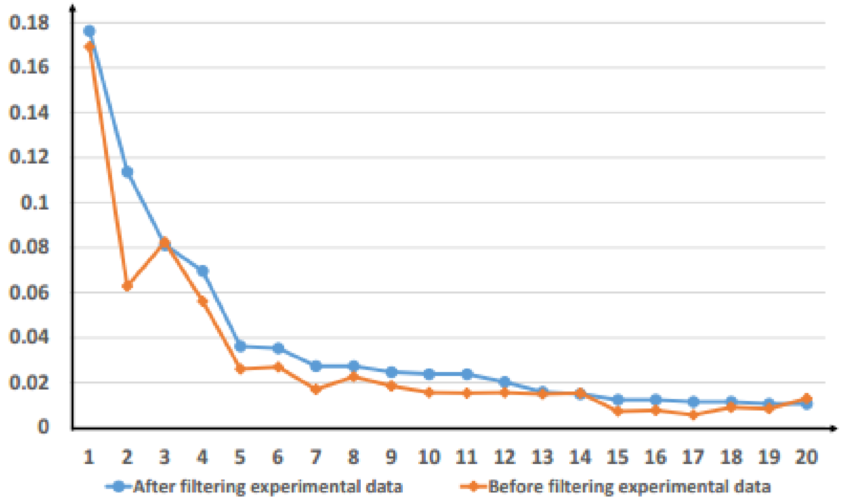
Fig. 2. An illustration of TSTR framework.

Outline

- Background
- Consideration and Design
- Evaluation
- Conclusion

Data

- 5K topics; 200K users; 900K tweets
- The data has been preprocessed to remove the noise: abandon topics which have less than 50 tweets and 20 unique users
- The topic distributions before and after the preprocessing are similar
  - the experimental data does not have too many biases to limit the applicability of our model at the system level.



Metrics

- We adopt five ranking metrics for evaluation
  - Bigger values represent better results

Reciprocal Rank (RR). RR only considers the first relevant tweet position. If this tweet is ranked at position p, then the RR value is $1/p$.

Precision at k (P@k). P@k is the percentage of relevant tweets in the returned top k tweets.

Average Precision (AP). AP considers all relevant tweets. If some predictions rank all 3 relevant tweets in the positions of $[1, 4, 7]$, then the AP value is $(1/3) \times (P@1 + P@4 + P@7)$.

Spearman's Rank Correlation Coefficient (Spearman's ρ). Spearman's ρ considers relative ranking difference between true relevant score S^t and predicted relevant score S^p. For a sample of size n, the i-th score pair (S_i^t, S_i^p) is converted to ranking pair (R_i^t, R_i^p), and the coefficient ρ is computed as $\rho = 1 - \frac{6\sum_1^n d_i^2}{n(n^2-1)}$, where $d_i = R_i^t - R_i^p$.

Normalized Discounted Cumulative Gain at k (NDCG@k). While the above metrics only consider the relative positions (i.e., relative ranking indexes) of relevant tweets, NDCG@k cares about relevant scores (i.e., absolute retweet count) of all the returned top k tweets. The metrics are formally defined as follows:

$$DCG@k = \sum_{i=1}^{k} \frac{log(score(i)+1)}{log(i+1)} \quad (12)$$

$$maxDCG@k = \sum_{i=1}^{k} \frac{log(score^*(i)+1)}{log(i+1)} \quad (13)$$

$$NDCG@k = \frac{DCG@k}{maxDCG@k} \quad (14)$$

where $score(i)$ denotes the retweet count of the tweet ranked at i-th position and $score^*$ denotes the retweet count list of the ideal ranking system.

Baselines

- **V2S**: a recently proposed **topic-specific** model [12, 13]

$$R_{uvm} \approx \sum_{k=1}^{K} [T_k(m) \cdot v_U^k(u) \cdot v_T^k \cdot s^k(v)]$$

Given the approximation in Equation 4.7, topic-specific user virality and susceptibility, and topic virality can be learnt by solving the following regularized tensor factorization problem.

$$(4.8) \qquad (\mathcal{V}_T^*, \mathcal{V}_U^*, \mathcal{S}^*) = \underset{\mathcal{V}_T, \mathcal{V}_U, \mathcal{S}}{arg.min} \mathcal{L}(\mathcal{V}_T, \mathcal{V}_U, \mathcal{S})$$

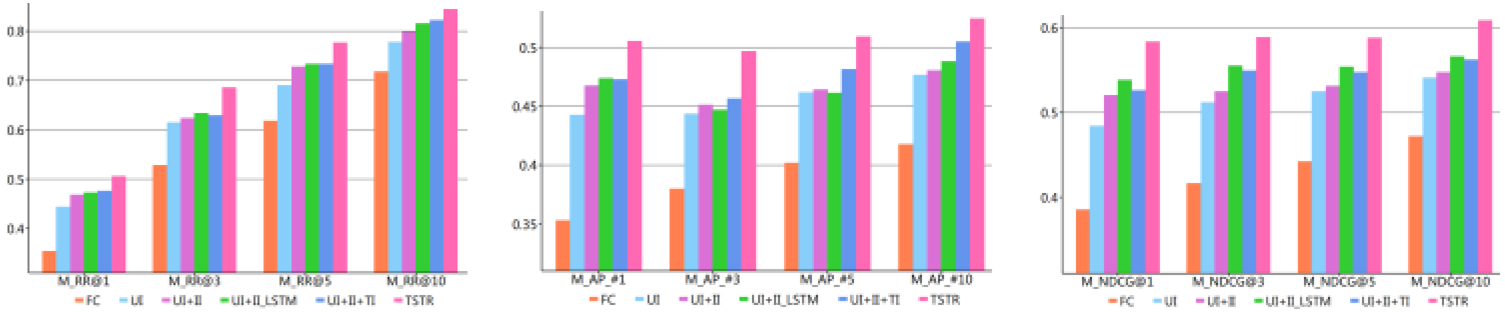
where $\mathcal{L}$ is the regularized sum-of-squares error function which is defined as follows.

$$(4.10) \quad \mathcal{L}(\mathcal{V}_T, \mathcal{V}_U, \mathcal{S}) =$$

$$= \sum_{(u,v,m)\in\mathcal{K}} \left[R_{uvm} - \sum_{k=1}^{K} (T_k(m) \cdot v_U^k(u) \cdot v_T^k \cdot s^k(v))\right]^2$$

- The following feature sets and their combinations are used to train ablation models
  - FC, follower count as feature
  - UI (User Info), user features as feature
  - II (Tweet Info), original tweet embeddings as feature
  - TI (Topic Info), original topic embeddings as feature
  - II_LSTM, tweet embeddings generated by LSTM-AE as feature
  - TI_DAE, topic embeddings generated by DAE as feature

Results



(a) Reciprocal Rank. (b) Average Precision. (c) NDCG.

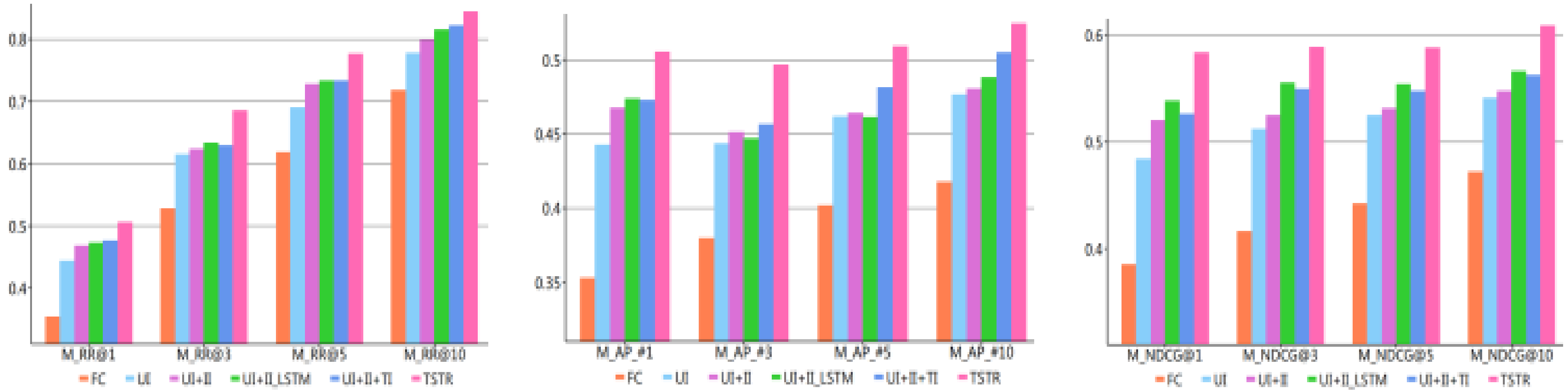
Fig. 4. Results of Reciprocal Rank, Average Precision and NDCG.

Table 1. Results of Precision and Spearman's ρ.

	FC	**UI**	**UI+II**	**UI+II_LSTM**	**UI+II+TI**	**TSTR**
M_P@1_#1	0.25568	0.35795	0.40341	0.40120	0.39205	**0.43713**
M_P@1_#3	0.42614	0.52272	0.55114	0.55689	0.53977	**0.61677**
M_P@1_#5	0.50568	0.59091	0.65909	0.66467	0.65909	**0.71856**
M_P@1_#10	0.61364	0.68750	0.73864	0.76647	0.75000	**0.79042**
M_P@3_#3	0.32765	0.39962	0.39773	0.41517	0.41098	**0.44711**
M_P@3_#5	0.43750	0.53409	0.52083	0.52894	0.55682	**0.58084**
M_P@3_#10	0.55114	0.66856	0.64205	0.65269	0.68561	**0.70259**
M_P@5_#5	0.37273	0.42386	0.41364	0.40838	0.42273	**0.45269**
M_P@5_#10	0.50568	0.59318	0.58068	0.56048	0.59773	**0.62874**
M_P@10_#10	0.38295	0.43750	0.42727	0.41617	0.45114	**0.49042**
Spearman's ρ	0.32757	0.33011	0.37825	0.38225	0.37660	**0.38359**

(1) Ablation models

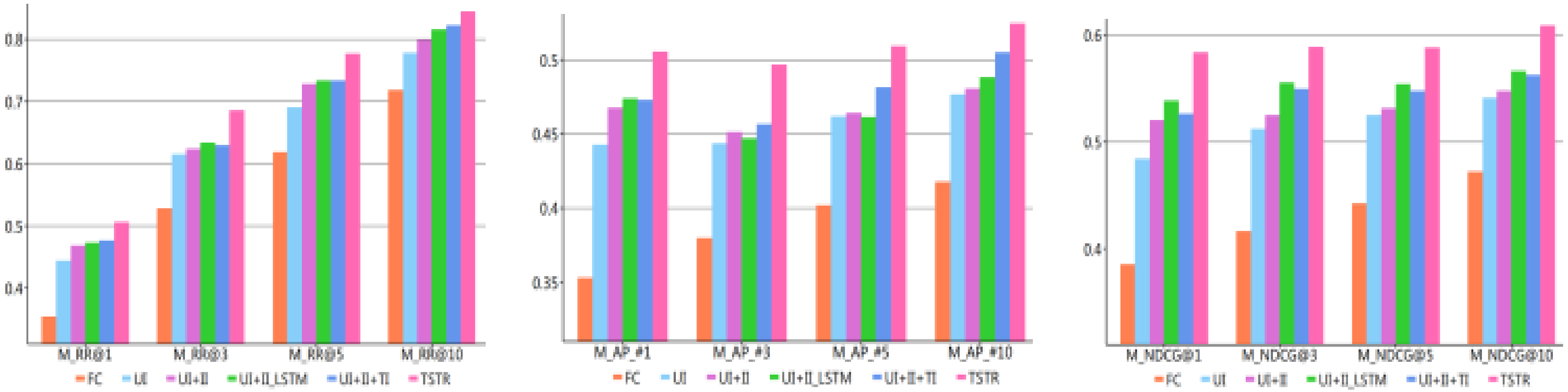
- UI: user features is important [6, 9, 17, 22].
- FC: the number of follower is important [6, 14, 15].
- **II: due to the topic-specific setting, the results of II are even worse than that of FC. The reason is that the tweets are too similar to distinguish in topic-specific setting.**
- **UI+II:** has better performance than the above because of feature interactions.



(a) Reciprocal Rank. (b) Average Precision. (c) NDCG.

(2) LSTM-AE

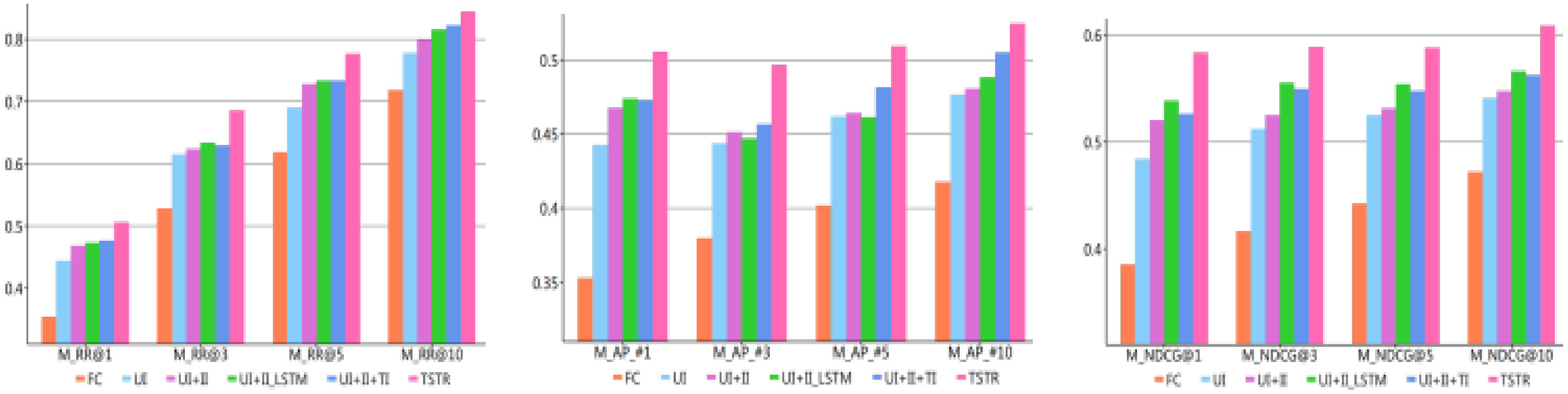
- LSTM-AE can generate effective features for short tweets with random-length, even though these tweets have similar contents in topic-specific setting
  - Most results of UI+II_LSTM are better than that of UI+II
  - UI+II_LSTM even performs better than UI+II+TI for metrics such as NDCG
  - Improvements are significant at the level of 0.05 in terms of Student`s t-test



(a) Reciprocal Rank. (b) Average Precision. (c) NDCG.

(3) Hypothesis testing

- Real-time topic information (i.e., TI) from Toutiao is able to boost the retweet count ranking task indeed.
  - All results of UI+II+TI are better than that of UI+II.
  - A few improvements are marginal, but please note that the model is only used to test our hypothesis.



(a) Reciprocal Rank. (b) Average Precision. (c) NDCG.

(4) TSTR

- Flexible framework for all metrics
  - All the entries are positive
- TSTR improves UI+II_LSTM / UI+II+TI
  - The average improvements are bigger than 6% for all metrics
  - T-test is 0.01 (some 0.05)
  - DAE can deal with news title text
  - LSTM-AE can deal with tweet text

Table 2. The improvements compared to other models.

	UI+II_LSTM	UI+II+TI
M_P@1_#1	8.96%	11.50%
M_P@1_#3	10.75%	14.27%
M_P@1_#5	8.11%	9.02%
M_P@1_#10	3.12%	5.39%
M_P@3_#3	7.69%	8.79%
M_P@3_#5	9.81%	4.31%
M_P@3_#10	7.65%	2.48%
M_P@5_#5	10.85%	7.09%
M_P@5_#10	12.18%	5.19%
M_P@10_#10	17.84%	14.24%
Ave. Improv.	**9.70%**	**8.23%**
M_AP_#1	6.72%	7.00%
M_AP_#3	11.17%	8.86%
M_AP_#5	10.47%	5.76%
M_AP_#10	7.53%	3.95%
Ave. Improv.	**8.97%**	**6.39%**
M_RR@1	7.02%	6.47%
M_RR@3	8.13%	9.00%
M_RR@5	5.95%	5.96%
M_RR@10	3.55%	2.66%
Ave. Improv.	**6.16%**	**6.02%**
M_NDCG@1	8.51%	10.95%
M_NDCG@3	6.09%	7.18%
M_NDCG@5	6.14%	7.43%
M_NDCG@10	7.55%	8.33%
Ave. Improv.	**7.07%**	**8.47%**
Spearman's ρ	0.35%	1.86%

(4) TSTR

- The improvements for Spearman's ρ are much smaller than the improvements for other metrics
  - Spearman's ρ cares more about **all tweets**
  - Other metrics care more about the **popular tweets**
    - Possible reason: too many unpopular tweets with small retweet count, which may have a lot of noise.
    - **TSTR is suitable for applications caring more about the higher ranked tweets, but not the ranking of all tweets.**
    - **Representative applications are recommendation and hot events detection.**

	UI+II_LSTM	UI+II+TI
M_P@1_#1	8.96%	11.50%
M_P@1_#3	10.75%	14.27%
M_P@1_#5	8.11%	9.02%
M_P@1_#10	3.12%	5.39%
M_P@3_#3	7.69%	8.79%
M_P@3_#5	9.81%	4.31%
M_P@3_#10	7.65%	2.48%
M_P@5_#5	10.85%	7.09%
M_P@5_#10	12.18%	5.19%
M_P@10_#10	17.84%	14.24%
Ave. Improv.	**9.70%**	**8.23%**
M_AP_#1	6.72%	7.00%
M_AP_#3	11.17%	8.86%
M_AP_#5	10.47%	5.76%
M_AP_#10	7.53%	3.95%
Ave. Improv.	**8.97%**	**6.39%**
M_RR@1	7.02%	6.47%
M_RR@3	8.13%	9.00%
M_RR@5	5.95%	5.96%
M_RR@10	3.55%	2.66%
Ave. Improv.	**6.16%**	**6.02%**
M_NDCG@1	8.51%	10.95%
M_NDCG@3	6.09%	7.18%
M_NDCG@5	6.14%	7.43%
M_NDCG@10	7.55%	8.33%
Ave. Improv.	**7.07%**	**8.47%**
Spearman's ρ	0.35%	1.86%

(5) V2S

- V2S performs worse than our TSTR model
  - values in the red boxes are **positive**
- V2S performs better than other ablation models
  - values in the red boxes are **smaller** than values in the blue boxes
- V2S cannot perform well for metrics such as P@**1**, P@**3**, AP#**1** and RR@**1** (as pointed out by the blue arrows)
  - V2S is not suitable for applications caring more about the higher ranked tweets, such as recommendation and hot events detection.
  - Our TSTR model is a better choice as analyzed before.

Table 2. The improvements compared to other models.

	UI+II_LSTM	UI+II+TI	V2S
M_P@1_#1	8.96%	11.50%	18.23%
M_P@1_#3	10.75%	14.27%	8.87%
M_P@1_#5	8.11%	9.02%	5.02%
M_P@1_#10	3.12%	5.39%	3.49%
M_P@3_#3	7.69%	8.79%	9.73%
M_P@3_#5	9.81%	4.31%	6.86%
M_P@3_#10	7.65%	2.48%	3.95%
M_P@5_#5	10.85%	7.09%	6.24%
M_P@5_#10	12.18%	5.19%	3.69%
M_P@10_#10	17.84%	14.24%	3.59%
Ave. Improv.	**9.70%**	**8.23%**	**6.97%**
M_AP_#1	6.72%	7.00%	7.70%
M_AP_#3	11.17%	8.86%	6.11%
M_AP_#5	10.47%	5.76%	4.87%
M_AP_#10	7.53%	3.95%	3.40%
Ave. Improv.	**8.97%**	**6.39%**	**5.52%**
M_RR@1	7.02%	6.47%	7.70%
M_RR@3	8.13%	9.00%	4.26%
M_RR@5	5.95%	5.96%	2.90%
M_RR@10	3.55%	2.66%	1.92%
Ave. Improv.	**6.16%**	**6.02%**	**4.20%**
M_NDCG@1	8.51%	10.95%	5.98%
M_NDCG@3	6.09%	7.18%	5.88%
M_NDCG@5	6.14%	7.43%	5.56%
M_NDCG@10	7.55%	8.33%	4.56%
Ave. Improv.	**7.07%**	**8.47%**	**5.50%**
Spearman's ρ	0.35%	1.86%	0.66%

Outline

- Background
- Consideration and Design
- Evaluation
- Conclusion

Our work

- A TSTR model is proposed to address the ***topic-specific*** retweet count ranking task in Weibo. Extensive experiments on real Weibo data show the effectiveness and flexibility of TSTR model.
- We leverage real-time news information from Toutiao to enrich the topic information, which is a general idea for other applications. A DAE is used to translate news information into topic features.
- A LSTM-AE extends traditional RNN-based encoder-decoder models for generating tweet features. The insight is that any different prefixes of tweet text is a possible distinctive feature!
- The experiments show that TSTR model is suitable for applications (e.g., hot events detection, recommendation) caring more about the higher ranked tweets (popular tweets).

Observations

- User features are more suitable for this topic-specific ranking task than tweet features

- Real-time topic information from Toutiao is potential to boost applications in Weibo

- TSTR framework is suitable for applications such as recommendation, hot events detection. In these applications, we care more about the *higher* ranked tweets, but not the ranking of *all* tweets.

Further improvements

- How to use historical information and network structure information properly in topic-specific setting
- Other novel methods for topic meaning enrichment and other novel models for topic feature extraction
- Perhaps, it is worth trying the end-to-end models

Thank you!