A Case Study of In-House Competition for Ranking Constructive Comments in a News Service

Hayato Kobayashi¹, Hiroaki Taguchi¹, Yoshimune Tabuchi¹, Chahine Koleejan¹, Ken Kobayashi¹, Soichiro Fujita², Kazuma Murao³, Takeshi Masuyama¹, Taichi Yatsuka¹, Manabu Okumura², Satoshi Sekine⁴

¹Yahoo Japan Corporation, ²Tokyo Institute of Technology, ³VISITS Technologies Inc., ⁴RIKEN
Background

- Ranking user comments is important for online news services because comment visibility directly affects the user experience.
- There have been many studies on comment ranking by user feedback.
  - (Hsu+ 2009, Das Sarma + 2010; Brand&V. D. Merwe 2014; Wei+ 2016)
- However, user feedback does not always represent comment quality.

![Figure 1: Comments on Yahoo! JAPAN News for article “Lifting the ban on drinking/smoking at 18.”](image)

(e.g., by position bias)
Fujita et al. (2019) introduced the concept of constructiveness in argument analysis for ranking comments without biased user feedback.

Constructiveness has no correlation with user feedback (Like/Dislikes).

**Table 1: Conditions for constructive comments.**

| Pre                  | Maintain decency and relevance |
|----------------------|-------------------------------|
| Related to article and not libelous |                               |

| Main                  | Represent typical cases of being constructive |
|-----------------------|-----------------------------------------------|
| Intended to stimulate discussions |                               |
| Objective and supported by fact  |                               |
| New idea, solution, or insight  |                               |
| User’s unique experience    |                               |
This Work

Approach

- Take Fujita et al.’s study one step further towards practical application.
  - Key aspect: Performance improvement by in-house competition.

Contributions

- Report the details of the in-house competition in Yahoo! JAPAN News.
  - 2.73% improvement in performance (NDCG) against the baseline.
- Consider several ensembles of the submitted various models.
  - 0.62% improvement in NDCG against the best single model.
In-House Competition

Task

- Ranking comments based on their constructiveness scores (C-scores).
  - C-score = a graded numeric score representing the level of constructiveness.

Dataset

- 59,120 comments (9,845 articles with about 6 comments).
  - Including 995 long comments (with 126-400 characters).

Evaluation

- NDCG: \( \frac{1}{K} \sum_{k=1}^{K} \text{NDCG}@k \)
  \[
  \text{NDCG}@k = Z_k \sum_{i=1}^{k} \frac{2^{r_i} - 1}{\log_2(i+1)}
  \]
- NDCG-L: NDCG only for the long comments (sub measure).
  - To avoid sloppy methods that determine long comments to be constructive.

| Pre | Related to article and not libelous |
|-----|------------------------------------|
| Main | Intended to stimulate discussions  |
|      | Objective and supported by fact    |
|      | New idea, solution, or insight     |
|      | User’s unique experience           |

Table 1: Conditions for constructive comments.
Submission Trend

- Number of submissions increased at the beginning of work (where time is more available) and on the day of the deadline.
- 8 individuals submitted:
  - 14 models during the competition period (before the deadline).
  - +4 models after the deadline.
- Total 18 models for research.

Figure 2: Cumulative number of submissions over the competition period.
Performance Increase (%) Compared to Baseline

- Many models performed better than Baseline.
- Highest performance increase was 2.73% by Model-17 for NDCG.
- Use of the leaderboard had a positive effect for participants submitting high-performance models for both measures in the latter half of the competition.

Figure 3: Increase (%) in NDCG (top) and NDCG-L (bottom) for each model compared to Baseline.

Baseline: A linear rankSVM model with features based on term-frequency vectors.
High-performance Models

- **Model-4**: Highest NDCG (before the deadline).
  - A gradient boosting model with features based on pretrained word embeddings.

- **Model-11**: Highest sum of NDCG and NDCG-L.
  - A linear rankSVM model with features based on C-score prediction (= stacking) and the distance between an article and its comment.

- **Model-14**: Highest NDCG-L.
  - A gradient boosting model with features based on maximal substrings and words.

- **Model-17**: Highest NDCG (after the deadline).
  - A variant of the RankNet model (BiLSTM+GCNN) with features based on subwords.
- Prepared 4 simple and 2 recent ensemble methods.
  - Simple methods: ScoreAve, NormAve (2011), RankAve, TopkAve (2009)
  - Recent methods: PostEval (2018), WeightEval (2020)

- **NormAve**: Use the average of the predicted scores of all models after normalizing the scores (Burges+ 2011).

- **WeightEval**: Use the weighted average of the top-k promising predictions (Fujita+ 2020), which is a hybrid of (continuous) majority voting and averaging. (The other methods are omitted due to time constraint.)
Results of Ensemble Models

- WeightEval performed the best for the main measure NDCG.
  - 0.62% improvement against the best single model.

- NormAve is the most promising for practical use (no parameter tuning).

|                | NDCG | NDCG-L | NDCG@3 | Prec@3 |
|----------------|------|--------|--------|--------|
| Baseline       | 81.63| 86.74  | 81.09  | 73.30  |
| Model-4        | 83.60| 82.15  | 82.79  | 73.98  |
| Model-11       | 83.35| 88.34  | 82.93  | 73.20  |
| Model-14       | 82.53| **88.77** | 81.83  | 72.86  |
| Model-17       | 83.86| 88.24  | 83.27  | 72.01  |
| ScoreAve       | 83.85| 86.66  | 83.20  | 73.40  |
| NormAve        | 84.33| 88.41  | 84.01  | **74.11** |
| RankAve        | 83.46| 88.25  | 82.92  | 73.30  |
| TopkAve        | 84.35| 88.35  | 83.31  | 73.54  |
| PostEval       | 84.32| 88.64  | 83.88  | 73.91  |
| WeightEval     | **84.38** | 88.30  | **84.18** | 74.04  |

Table 2: NDCG variants (%) and precision (%) for (a part of) the submitted models and their ensembles.
- Qualitative evaluation from the perspective of service.
  - 3 service experts ranked the comment lists created by candidate models.
  - Criterion: Which list should be provided as a service?

- Two cases:
  - Baseline vs. naive methods.
  - Baseline vs. submitted models.
    - Service preferred not to use ensemble models because it would be unreasonable to maintain different models.
Baseline vs. Naive Methods

- **Feedback**: Descending/ascending order of number of Likes/Dislikes.
- **Latest**: Descending order of comment date.
- **Length**: Descending order of comment length.

- Baseline (C-score) clearly performed better than the other methods.
- Constructiveness is useful even in human evaluation, while the previous study (Fujita+ 2019) used NDCG only.

|                  | Average Rank |
|------------------|--------------|
| Feedback         | 2.61         |
| Latest           | 3.42         |
| Length           | 2.20         |
| Baseline (C-score)| **1.77**    |

Table 3: Qualitative evaluation results of Baseline and naive methods (lower ranks are better).
Baseline vs. Submitted Models

- Prepared the four high-performance single models.
  - Model-4 (GBM with word embeddings), Model-11 (rankSVM with stacking), Model-14 (GBM with maximal substrings), Model-17 (RankNet with subwords).

- Best single model (Model-17) also had the best average rank.

- Competition format is effective even in a service-level judgment.

| Model        | Average Rank |
|--------------|--------------|
| Baseline     | 3.86         |
| Model-4      | 3.64         |
| Model-11     | 3.63         |
| Model-14     | 3.41         |
| Model-17     | **3.11**     |

Table 4: Qualitative evaluation results of submitted models and Baseline (lower ranks are better).
Conclusion

Summary

- Reported the details of the in-house competition in Yahoo! JAPAN News.
  - 2.73% improvement in performance (NDCG) against the baseline.

Discussion

- Service decision suggests that while an ensemble of different models is promising in an academic sense, it still has challenges in an industrial sense.
  - Model unification/distillation for improving maintainability and latency?
Thank you!