Generating Sports News from Live Commentary: A Chinese Dataset for Sports Game Summarization

Kuan-Hao Huang[1] **Chen Li**[2] **Kai-Wei Chang**[1]
[1]University of California, Los Angeles
[2]Tencent AI Lab
[1]{khhuang, kwchang}@cs.ucla.edu
[2]lichen.lc18@gmail.com

Abstract

Sports game summarization focuses on generating news articles from live commentaries. Unlike traditional summarization tasks, the source documents and the target summaries for sports game summarization tasks are written in quite different writing styles. In addition, live commentaries usually contain many named entities, which makes summarizing sports games precisely very challenging. To deeply study this task, we present SPORTSSUM[1], a Chinese sports game summarization dataset which contains 5,428 soccer games of live commentaries and the corresponding news articles. Additionally, we propose a two-step summarization model consisting of a *selector* and a *rewriter* for SPORTSSUM. To evaluate the correctness of generated sports summaries, we design two novel score metrics: *name matching score* and *event matching score*. Experimental results show that our model performs better than other summarization baselines on ROUGE scores as well as the two designed scores.

1 Introduction

There are a large number of sports games playing every day. Apparently, manually writing sports news articles to summarize every game is labor-intensive and infeasible. How to automatically generate sports summaries, therefore, becomes a popular and demanding task. Recently, generating news from live commentaries has gradually attracted attention in the academic community (Zhang et al., 2016; Yao et al., 2017). At the same time, several trials have been done in the industry such as sports news from Toutiao's Xiaoming Bot[2], Sohu Ruibao[3] and AI football news[4].

[1]The dataset is available at https://github.com/ej0cl6/SportsSum

[2]http://www.nbd.com.cn/columns/803

[3]https://mp.sohu.com/profile?xpt=c29odW1wMzZpdDlzQHNvaHUuY29t

[4]https://www.51zhanbao.com

Live Commentary		
Time	Scores	Commentary Sentence
66'	0-0	多特蒙德球员格策拼抢犯规,对手获得控球权. Dortmund's player Götze fouled, and the opponent got the possession of the ball.
66'	0-0	施魏因斯泰格为拜仁慕尼黑赢得一个任意球. Schweinsteiger got a free kick for Bayern Munich.
67'	1-0	进球啦！！！拜仁慕尼黑球员克罗斯大禁区外左脚射门,球从右下角飞进球门,球进了!助攻的是穆勒. 拜仁慕尼黑1-0 多特蒙德. Goal!!! Bayern Munich's player Kroos shot with his left foot from the outside of the penalty area. The ball flew into the goal through the lower right corner. The ball went in! Muller gave the assist. Bayern Munich 1-0 Dortmund.
71'	1-0	拜仁慕尼黑球员里贝里大禁区左侧尝试右脚射门,可惜皮球高出球门.给他传球的是拉姆. Bayern Munich's player Ribery tried to shoot with his right foot from the penalty area's left side, but the ball was higher than the crossbar. Lahm passed the ball to him.

Sports News Ariticle
开场3分钟，克罗斯左侧任意球被顶到后点，里贝里禁区边缘抽射偏出近门柱。第8分钟，穆勒右路与曼朱基奇打出踢墙配合，在门前12米处推射被苏博蒂奇铲出底线。第13分钟，里贝里右路塞球，克罗斯在门前27米处抽射偏出近门柱。(...) In the 3rd minutes, Kroos's free kick on the left was tipped to the back, and Ribery's shot from the penalty area missed. In the 8th minute, Muller and Mandzukic had teamwork, and Muller's shot from the 12 meters ahead the goal line was touched out by Subotić. In the 13th minute, Ribery passed the ball from the right, and Kroos's shot near the 27 meters ahead the goal line missed. (...)

Table 1: An example of SPORTSSUM dataset.

Unlike traditional text summarization tasks (Hermann et al., 2015; Rush et al., 2015), the source documents and the target summaries for sports game summarization tasks are written in quite different styles. Live commentaries are the real-time transcripts of the commentators. Accordingly, commentary sentences are more colloquial and informal. In contrast, news summaries are usually more narrative and well-organized since they are written after the games. In addition, commentaries contain a large number of player names. One player can be referred to multiple times in the whole game, and one commentary sentence may mention multiple player names simultaneously. Those properties

Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, pages 609–615

make sports games summarization tasks very challenging.

In this paper, we present SPORTSSUM, a Chinese dataset for studying sports game summarization tasks. We collect 5,428 pairs of live commentaries and news articles from seven famous soccer leagues. To the best of our knowledge, SPORTSSUM is the largest Chinese sports game summarization dataset. In addition, we propose a two-step summarization model for SPORTSSUM, which learns a *selector* to extract important commentary sentences and trains a *rewriter* to convert the selected sentences to a news article. To encourage the model to capture the relations between players and actions better, we replace all the player names in the training sentences with a special token and train the proposed model on the modified template-like sentences.

The proposed model performs better than existing extractive and abstractive summarization baseline models in ROUGE scores (Lin, 2004). However, we observe that ROUGE scores cannot evaluate the correctness of generated summaries very well. Therefore, we design two new scores, *name matching score* and *event matching score*, as the auxiliary metrics for SPORTSSUM. Our experimental results demonstrate that the proposed model is superior to the baseline models in all the metrics.

Summarizing documents between two articles written in different styles and involving many named entities is not limited to the sports game summarization tasks. There are many possible applications, such as summarizing events from tweets and summarizing trends from forum comments. We hope that SPORTSSUM provides a potential research platform to develop advanced techniques for this type of summarization tasks.

2 Dataset

We present SPORTSSUM, a sports game summarization dataset in Chinese.

Data collection. We crawl the records of soccer games from Sina Sports Live[5]. The collected records contain soccer games in seven different leagues (Bundesliga, CSL, Europa, La Liga, Ligue 1, PL, Series A, UCL) from 2012 to 2018. For each game, we have a live commentary document C and a news article R, as illustrated in Table 1. The live commentary document C consists of a series of tuples (t_i, s_i, c_i), where t_i is the timeline information, s_i represents the current scores, and c_i denotes the commentary sentence. The news article R consists of several news sentences r_i. In addition to commentaries and news reports, we also include some metadata, such as rosters, starting lineups, and player positions, which is potentially helpful for sports game summarization tasks.

[5] https://match.sports.sina.com.cn/

League	# of games
Bundesliga	453
CSL	1371
Europa	143
La Liga	713
Ligue 1	161
Premier League	1220
Serie A	890
UCL	477
All	5428

Table 2: The number of games in different leagues.

Source	Avg. # chars	Avg. # words	Avg. # sent.	Total # vocab
Commentary	3459.97	1825.63	193.77	43482
News	801.11	427.98	23.80	21294

Table 3: Statistics of SPORTSSUM dataset.

Data cleaning. The crawled live commentary documents and news articles are quite noisy. Therefore, we apply multiple steps of data cleaning to improve the quality of the dataset. We first remove all the HTML tags from the commentary documents and the news articles. Then, we observe that there are usually some descriptions that cannot be directly inferred from the commentaries at the beginning of news articles, such as matching history. Hence, we design a heuristic rule to remove those descriptions. We identify several *starting keywords* which can indicate the start of a game, such as "一开场(at the beginning of the game)" and "开场后(after the game started)". The full list of starting keywords can be found in Appendix A. Once we see a starting keyword appearing in a news report, we remove all the sentences before the starting keyword. Finally, we discard those games with the number of news sentences being less than 5 and the number of commentary sentences being less than 20. After data cleaning, we have 5,428 games remaining (detailed numbers of games are shown in Table 2).

Notice that SPORTSSUM (5,428 games) is much larger than the only public sports game summariza-

tion dataset (150 games) (Zhang et al., 2016).

Statistics and properties. Table 3 shows the statistics of SPORTSSUM. On average, there are 193.77 sentences per commentary document and 23.80 sentences per news article. After applying word segmentation by *pyltp* tool[6], the average numbers of words for commentary documents and news reports are 1825.63 and 427.98, respectively.

As mentioned in Section 1, commentary sentences and news sentences are in quite different writing styles. Commentary sentences are more colloquial and informal, while news sentences are more narrative and well-organized. Also, commentaries contain a large number of player names, which makes the model easy to generate news reports with incorrect facts, as shown in Section 3.

3 Sports Game Summarization

The goal of sports game summarization is to generate a sports news report $\tilde{R} = \{\tilde{r}_1, \tilde{r}_2, .., \tilde{r}_n\}$ from a given live commentary document $C = \{(t_1, s_1, c_1), ..., (t_m, s_m, c_m)\}$. The generated news report $\tilde{R}$ is expected to cover most of the important events in the games and describe those events correctly. In this paper, we propose a two-step model for SPORTSSUM. The proposed model first learns a *selector* to extract important commentary sentences and then utilizes a *rewriter* to convert the selected sentences to a news article.

Sentence mapping. To train the selector and rewriter, we need some labels to indicate the importance of commentary sentences and the corresponding news sentences. To obtain the labels, we consider the timeline information and BERTScore (Zhang et al., 2020), a metric to measure the sentence similarity, and map each news sentence to a commentary sentence. Although we have no explicit timeline information for news sentences, we observe that many news sentences start with "in the n-th minute" and thus we can extract the timeline information for some news sentences.

We map sentences by the following steps: **1)** For each news sentence r_i, we extract the timeline information h_i if possible. Otherwise, we do not map this news sentence. **2)** We consider those commentary sentences c_j with t_j being close to h_i. More specifically, we consider $C^{(i)} = \{c_k, c_{k+1}, ...c_{k+l}\}$, where c_j is the commentary sentence with timeline information $t_j \in [h_i, h_i+3]$ for $k \leq j \leq k+l$. **3)** We compute BERTScore of the news sentence r_i and all the commentary sentences in $C^{(i)}$. The commentary sentence $c_j \in C^{(i)}$ with the highest score is considered to be mapped with the news sentences r_i.

With the above mapping process, we obtain a set of mapped commentary sentences and news sentences $\mathcal{D} = \{(\bar{c}_1, \bar{r}_1), (\bar{c}_2, \bar{r}_2), ..., (\bar{c}_s, \bar{r}_s)\}$, which can be used for training our selector and rewriter.

Selector. There are many commentary sentences in a live commentary document, but only few of them contain valuable information and should be reported in the news article. Therefore, we learn a *selector* to pick up those important sentences. More specifically, Given a commentary document $C = \{(t_1, s_1, c_1), ..., (t_m, s_m, c_m)\}$, the selector outputs a set $C_{select} = \{\tilde{c}_1, \tilde{c}_2, ..., \tilde{c}_n\}$ which contains only important commentary sentences.

We train a binary classifier as the selector to choose important commentary sentences. When training, for each commentary sentence c_i in C, we assign a positive label if c_i can be mapped with a news sentence by the aforementioned mapping process. Otherwise, we give a negative label.

Rewriter. The rewriter converts the selected commentary sentences $C_{select} = \{\tilde{c}_1, \tilde{c}_2, ..., \tilde{c}_n\}$ to a news report $\tilde{R} = \{\tilde{r}_1, \tilde{r}_2, .., \tilde{r}_n\}$. We focus on the sentence-level rewriter. That is, we convert each selected commentary sentence $\tilde{c}_i$ to a news sentence $\tilde{r}_i$. An intuitive way to learn the sentence-level rewriter is training a sequence-to-sequence (seq2seq) model, such as LSTM (Hochreiter and Schmidhuber, 1997) and Transformer (Vaswani et al., 2017), on the mapped sentences $\mathcal{D}$. However, as illustrated in Table 4, we observe that the seq2seq model tends to generate high-frequency player names rather than the correct player names even though the high-frequency player names do not appear in the commentary sentences. We call this situation *name mismatch problem*.

To solve the name mismatch problem, we train the rewriter in a *template-to-template* (tem2tem) way instead of in a seq2seq way. We first build a dictionary of player names from the lineup data (metadata). Next, for each $(\bar{c}_i, \bar{r}_i)$ in $\mathcal{D}$, we replace all the player names in $\bar{c}_i$ and $\bar{r}_i$ with a special token "`[player]`" so that the new sentence is like a template. If there are multiple player names in a sentence, we append a number to the special token to distinguish them, as shown in Table 5.

[6] https://github.com/HIT-SCIR/pyltp

Live Commentary Sentence	里贝里禁区左侧尝试右脚射门,皮球高出球门.给他传球的是拉姆. **Ribery** tried to shoot with his right foot from the left side of the penalty area, but the ball was higher than the crossbar. **Lahm** passed the ball to him.
Gound Truth News Sentence	拉姆转移到左侧，里贝里突入禁区左侧距门12米处抽射高出。 **Lahm** passed the ball to the left, and **Ribery** cut in the left penalty area and shot from 12 meters ahead the goal line. The shot was too high.
News Sentence Generated by Seq2seq Model	里贝里传球，曼朱基奇禁区左侧射门偏出远门柱。 **Ribery** passed the ball and **Mandzukic**'s shot from the left side of the penalty area was out of the goalpost.

Table 4: An example of the name mismatch problem. Seq2seq model tends to generate high-frequency player names rather than the correct names.

	Input Sentence	Output Sentence
Seq2seq	射门!!!里贝里球门线跟前右脚射门, 被阿德勒横身扑出. 给他传球的是拉姆. Shoot!!! Ribery's right foot shot in front of the goal line was saved by Adler. Lahm passed the ball to him.	拉姆右路低传，里贝里前点铲射被阿德勒封出。 Lahm made a low pass on the right and Ribery's shot from the front was blocked by Adler.
Tem2tem	射门!!!**[player1]** 球门线跟前右脚射门, 被**[player2]**横身扑出. 给他传球的是**[player3]**. Shoot!!! **[player1]**'s right foot shot in front of the goal line was saved by **[player2]**. **[player3]** passed the ball to him.	**[player3]** 右路低传，**[player1]**前点铲射被**[player2]**封出。 **[player3]** made a low pass on the right and **[player1]**'s shot from the front was blocked by **[player2]**.

Table 5: Training models by seq2seq versus training models by tem2tem.

After converting $\bar{c}_i$ and $\bar{r}_i$ to the template sentences, we train a seq2seq model on the template sentences. By training models in a tem2tem way, the model focuses more on the relations between players and actions and is less influenced by the high-frequency player names.

When predicting, for each commentary sentence $\tilde{c}_i$ in C_{select}, we use the aforementioned way to convert $\tilde{c}_i$ to a commentary template sentence. Then, we generate a news template sentence by the rewriter and replace all the special tokens in the sentence with the original player names.

4 Experiments

SPORTSSUM contains 5,428 games and we split them into three sets: training (4,828 games), validation (300 games), and testing (300 games) sets.

Evaluation. We consider ROUGE scores (Lin, 2004), which are standard metrics for summarization tasks. More precisely, we focus on ROUGE-1, ROUGE-2, and ROUGE-L. However, we observe that ROUGE scores cannot accurately evaluate the correctness of summaries. Some summaries may get high ROUGE scores but contain many incorrect facts. Therefore, we design two metrics: *name matching score* (NMS) and *event matching score* (EMS).

The name matching score evaluates the closeness of the player names in the ground truth news article R and the generated summaries $\tilde{R}$. Let N_g and N_p denote the set of the player names appearing in R and $\tilde{R}$, respectively. We define the name matching score as

$$\text{NMS}(R, \tilde{R}) = \text{F-score}(N_g, N_p).$$

Similarly, the event matching score evaluates the closeness of the events in R and $\tilde{R}$. We define an event as a pair (subject, verb) in the sentence. Two pairs $(\text{subject}_1, \text{verb}_1)$ and $(\text{subject}_2, \text{verb}_2)$ are viewed as equivalent if and only if **1)** subject_1 is the same as subject_2 and **2)** verb_1 and verb_2 are synonym[7] to each other. Let E_g and E_p represent the set of events in R and $\tilde{R}$, respectively, the event matching score is defined as

$$\text{EMS}(R, \tilde{R}) = \text{F-score}(E_g, E_p).$$

Implementations and Models. We consider the convolutional neural network (Kim, 2014) as the selector. For the rewriter, we consider the following: (1) **LSTM**: a bidirectional LSTM with attention mechanism (Bahdanau et al., 2015). (2) **Transformer**. (Vaswani et al., 2017) (3) **PGNet**: pointer-generator network, an encoder-decoder model with copy mechanism (See et al., 2017).

[7]Details to decide synonyms can be found in Appendix B.

Method	Model	ROUGE-1	ROUGE-2	ROUGE-L	NMS	EMS
Extractive Models	RawSent	26.52	7.64	25.42	57.33	36.17
	LTR	24.44	6.39	23.19	51.63	29.03
Abstractive Models	Abs-LSTM	30.54	10.16	29.78	10.87	14.03
	Abs-PGNet	34.02	11.09	33.13	17.87	19.76
Selector + Rewriter (Seq2seq)	LSTM	41.39	16.99	40.53	28.48	25.19
	Transformer	41.71	18.10	40.96	35.63	30.94
	PGNet	**43.17**	**18.66**	**42.27**	48.18	36.94
Selector + Rewriter (Tem2tem)	LSTM	41.71	17.08	40.82	**59.54**	40.34
	Transformer	41.47	17.18	40.54	58.26	39.33
	PGNet	41.95	17.09	41.01	59.35	**40.46**

Table 6: Evaluation results. NMS and EMS represent the name matching score and the event matching score.

For comparison, we consider two extractive summarization baselines: (1) **RawSent**: the raw sentences selected by the selector without rewriting. (2) **LTR**: the learning-to-rank approach for sports game summarization proposed by the previous work (Zhang et al., 2016).

In addition, we train a bidirectional LSTM with attention mechanism (**Abs-LSTM**) and a pointer-generator network (**Abs-PGNet**) on the paired commentaries and news articles as two simple abstractive summarization baselines. More implementation details can be found in Appendix C.

Results. Table 6 shows the experimental results. We observe that the extractive models (RawSent and LTR) get low ROUGE scores but high NMS and EMS. That means the extractive models can generate summaries with correct information, but the writing style is different from the ground truth. On the contrary, the abstractive models get higher ROUGE scores but lower NMS and EMS. That implies the summaries generated by the abstractive models usually contain incorrect facts.

Our proposed two-step model performs better than the extractive models and the abstractive models on ROUGE scores, NMS, and EMS. This verifies our design of the selector and the rewriter. In addition, we observe that when training the model in a tem2tem way, we can get better NMS and EMS, which implies that training by tem2tem can improve the correctness of summaries.

5 Related Work

Text summarization. Existing approaches can be grouped into two families: extractive models and abstractive models. Extractive models select a part of sentences from the source document as the summary. Traditional approaches (Carbonell and Goldstein, 1998; Erkan and Radev, 2004; McDonald, 2007) utilize graph or optimization techniques. Recently, neural models achieve good performance (Cheng and Lapata, 2016; Nallapati et al., 2017; Jadhav and Rajan, 2018). Abstractive summarization models aim to rephrase the source document. Most work applies neural models for this task. (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016; Zeng et al., 2016; See et al., 2017; Gehrmann et al., 2018).

Factual correctness of summaries. There is a lot of work focusing on evaluation and improvement of the factual correctness of summaries (Falke et al., 2019; Kryscinski et al., 2019; Wang et al., 2020; Maynez et al., 2020; Zhu et al., 2020).

Data-to-Text generation. Recently, generating news articles from different kinds of data-records becomes a popular research direction. Wiseman et al. (2017); Puduppully et al. (2019) focus on generating news from boxed-data. Zhang et al. (2016) and Yao et al. (2017) study generating sports news from live commentaries, but their methods are based on hand-crafted features.

6 Conclusion

We present SPORTSSUM, a Chinese dataset for sports game summarization, as well as a model that consists of a selector and a rewriter. To improve the quality of generated news, we train the model in a tem2tem way. We design two metrics to evaluate the correctness of generated summaries. The experimental results demonstrate that the proposed model performs well on ROUGE scores and the two designed scores.

Acknowledgments

We thank Tecent AI Lab, UCLA-NLP group, and anonymous reviewers for their feedback.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Jaime G. Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *ACL*.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *NAACL*.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Tobias Falke, Leonardo F. R. Ribeiro, Prasetya Ajie Utama, Ido Dagan, and Iryna Gurevych. 2019. Ranking generated summaries by correctness: An interesting but challenging application for natural language inference. In *ACL*.

Sebastian Gehrmann, Yuntian Deng, and Alexander M. Rush. 2018. Bottom-up abstractive summarization. In *EMNLP*.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Aishwarya Jadhav and Vaibhav Rajan. 2018. Extractive summarization with SWAP-NET: sentences and words from alternating pointer networks. In *ACL*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Evaluating the factual consistency of abstractive text summarization. *Preprint arXiv:1910.12840*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan T. McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *ACL*.

Ryan T. McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *ECIR*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *CoNLL*.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. In *AAAI*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Alex Wang, Kyunghyun Cho, and Mike Lewis. 2020. Asking and answering questions to evaluate the factual consistency of summaries. In *ACL*.

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2017. Challenges in data-to-document generation. In *EMNLP*.

Jin-ge Yao, Jianmin Zhang, Xiaojun Wan, and Jianguo Xiao. 2017. Content selection for real-time sports news construction from commentary texts. In *INLG*.

Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2016. Efficient summarization with read-again and copy mechanism. *Preprint arXiv:1611.03382*.

Jianmin Zhang, Jin-ge Yao, and Xiaojun Wan. 2016. Towards constructing sports news from live text commentary. In *ACL*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating text generation with BERT. In *ICLR*.

Chenguang Zhu, William Hinthorn, Ruochen Xu, Qingkai Zeng, Michael Zeng, Xuedong Huang, and Meng Jiang. 2020. Boosting factual correctness of abstractive summarization with knowledge graph. *Preprint arXiv:2003.08612*.

A Starting Keywords

We consider the following regular expressions as the starting keywords:

- 一开场
- 开场后
- 开场[\d]+分钟
- 开始[\d]+分钟
- 开场[仅][\d]+秒
- [\d]+秒
- 第[\d]+分钟
- [\d]+分钟
- [\d]+[米码]

B Event Matching Score

We pick up the top 300 most frequent verbs and ask human to annotate if the verb is an important verb for soccer games or not. Then, we ask human to cluster those important verbs based on their meanings. When calculating the event matching score, we only consider those verbs. Two verbs are viewed as the synonym to each other if they are in the same group. The groups of verbs are as follows:

- Shooting: 射门, 打门, 攻门, 抽射, 推射, 劲射, 远射, 低射, 补射, 扫射, 斜射, 捅射, 射, 怒射, 起脚, 铲射, 垫射, 吊射, 挑射, 弹射, 勾射, 爆射, 头球, 甩头
- Missed Shot: 偏出, 高出, 打偏, 弹出, 打高, 弹回, 打飞, 顶高, 顶偏, 超出, 射偏, 蹭偏, 蹭出, 滑出
- Passing: 传中, 传球, 斜传, 送出, 头球摆渡, 直塞, 横传, 挑传, 直传, 低传, 横敲, 给到, 传入, 传, 传到, 妙传, 斜塞, 长传, 短传, 回传, 回敲, 回点, 分球
- Blocking: 扑出, 挡出, 没收, 封堵, 得到, 封出, 托出, 扑住, 救下, 抱住, 救出
- Defense: 解围, 破坏, 铲出, 化解
- Foul: 犯规, 吃到, 警告, 判罚, 被判, 领到, 罚下, 出示, 被罚

C Implementation Details

For the selector, we consider CNN with the same architecture in (Kim, 2014) and set the learning rate to 10^{-3}.

For the rewriter, the implementation details are as follows:

- **LSTM**: we use a bidirectional LSTM with the attention mechanism (Bahdanau et al., 2015). The size of hidden state is set to 300. We set the learning rate to 10^{-3}.
- **Transformer**: we use the Transformer with the same architecture in the original paper (Vaswani et al., 2017). We set the learning rate to 10^{-4}.
- **PGNet**: we implement the pointer-generator network (See et al., 2017) and set the size of hidden state to 300. We set the learning rate to 10^{-3}.
- **LSTM-abs**: we use a bidirectional LSTM with the attention mechanism (Bahdanau et al., 2015). The size of hidden state is set to 300. We set the learning rate to 10^{-3}.
- **PGNet-abs**: we implement the pointer-generator network (See et al., 2017) and set the size of hidden state to 300. We set the learning rate to 10^{-3}.

For all the models, we use the 200-dimensional pre-trained Chinese word embedding from Tecent AI Lab[8].

[8] https://ai.tencent.com/ailab/nlp/embedding.html