Using GAN to Generate Sport News from Live Game Stats

Changliang Li¹(✉), Yixin Su², Ji Qi³, and Min Xiao⁴

¹ Kingsoft AI Lab, Beijing, China

lichangliang@kingsoft.com

² University of Melbourne, Parkville, Australia

yixins1@student.unimelb.edu.au

³ Kingsoft WPS AI, Wuhu, Anhui, China

qiji1@wps.cn

⁴ Beijing University of Posts and Telecommunications, Beijing, China

xiaomincloud@gmail.com

Abstract. One goal in artificial intelligence field is to create well-formed and human-like natural language text given data input and a specific goal. Some data-to-text solutions have been proposed and successfully used in real applied domains. Our work focuses on a new domain, Automatic Sport News Generating, which aims to produce sport news immediately after each match is over so that both time and labor can be saved on writing the news articles. We propose to use Generative Adversarial Networks (GAN) architecture for generating sport news based on game stats. Our model can automatically determine what is worth reporting and generate various appropriate descriptions about the game. We apply our approach to generate NBA (National Basketball Association) game news. Especially, This paper focuses on reporting the summary of game result and performance of players. Our model achieves good results on both tasks. To our best knowledge, this is the first work based on GAN to generate sports news using game statistics.

Keywords: GANs · Text generation · Natural language processing

1 Introduction

Natural Language Generation (NLG) and data-to-text approaches have gained increasing attention in recent years due to the applicability and challenge. By automatically generating high quality texts based on extracted relevant information from the source data, these approaches can eliminate the gap between raw data and human users.

There are some data-to-text solutions have been proposed for specific domains. There are some examples, generation of weather reports from meteorological data in several languages [23, 24], the creation of custom letters which answer customers' questions [25], the generation of reports about the state of

neonatal babies from intensive care data [26], and the generation of project management [27] and air quality reports [28].

With the development of sports (such as football and basketball), there are thousands of games concerned by billions of people every year. In this paper, we propose an Automatic Sport News Generating System, which aims to produce sport news immediately after each match is over so that saves both time and labor on writing the news articles. However, there are few researches in this area by now, especially using machine learning models, which leaves space in generating articles with better quality. One of the latest research in this area simplifies it into learning to rank problem. This method just pieces together past news sentences based on live comments instead of actually generating sentences using the match data [3]. In this paper, our system generates National Basketball Association (NBA) game news directly from stats. NBA games news usually consists of three parts: the first part summarizes the overall game result; the second part describes player's performance; the last part describes the situation and events occurred during each of four sections. This paper focuses on first two parts, which are closely related to the game stats.

Recently, Generative Adversarial Networks (GAN) [1] have been introduced as a novel way to train a generative model. It has significantly improved several machine learning research fields, such as image generation [13], video prediction [7] and other domains [6, 8, 14]. The newly proposed WGAN, which solved the convergence problem, has enable the GAN to be utilized on fields based on discrete data, such as Natural Language Procession (NLP).

In this work, our Automatic Sport News Generating System is based on WGAN. There are two models that fulfil different generating tasks in the system. First we leverage a WGAN model to generate most important and variable phrases that can accurately describe the match or player's performances. Then, by putting these phrases together with other constant information into template-based sentence generator, fluent sentences can be generated. This procedure ensures that the generated sentences are well-written and brings clarity to the reader, which is important in sports news. Concatenating outputs from Summary Sentence Generating Model (SSGM), and Player Performance Generating Model (PPGM), the system finally generates the most important parts in NBA sport news.

The main contribution of our system is that it can generate proper phrases to describe different matches, without the intervene of experts (news writers in sport news) or rules. For example, the system can judge which player plays good in a match and generates corresponding phrases to describe him. This process is automatically performed by WGAN according to real news and stats. Furthermore, we define a series of rules for sentence generator so that the generated sentences will be various, and are more close to idiomatic expression. As our best knowledge, this is the first framework that successfully generates sport news with the participating of GAN model based on game stats. It proves the potential of GAN model on NLP applications.

The rest of our paper is structured as follows: Sect. 2 describes related works; Sect. 3 gives a detailed description of our model; Sect. 4 analyzed experiment results and Sect. 5 summarizes this work and the future direction.

2 Related Work

Automatic Sports News Generating is a promising task that aims to generate sports news short after each game finishes. Then readers can receive the news quickly and saves human labor. Currently, there are not many researches on this area. A research recently [3] attempt to generate football news from live comments and use learning to rank method to extract most probable sentence appeared previously to fit a new live comment. Basically, the author reduced the problem into a ranking problem.

GAN has drawn significant attention as a new machine learning model recently [1]. The task of GAN is to produce a generator by a minimax game. In last few years, many variations have been invented to improve the original GAN, such as f-GAN [15], Energy based GAN [20], info GAN [17], and WGAN [21], which proves that GAN is a promising model.

Currently, A work that combines deep convolutional neural networks and GAN (DCGAN) [8] has been proposed, which proves that GAN has great potential in image generating area. Inspired by DCGAN, researchers have made much progress on image generating aspect. A model that utilize DCGAN within Laplacian pyramid framework [6] has been proved that it can produce higher quality images than original DCGAN. Meanwhile, DCGAN has been used to generate photo-realistic natural images by mainly updating the loss function cite.

Besides image generating, NLP is another important task in machine learning area. However, GAN is considered to be difficult to apply on NLP [1] since the update of GAN is based on continued space, but languages are discrete.

Though application of GAN on NLP is tough, there are several attempts on it. An effort that tries to learn words representation from documents [9] has bypass the problem of handling natural language sentences directly. In addition, researchers try to solve the discrete problem by modelling the generator as Reinforcement Learning (RL) called SeqGAN [18]. This model avoids the differentiate problem caused by gradient policy in generator. On the other hand, the discrete problem can be solved by mapping discrete natural languages into continuous vector spaces as well. Combining LSTM and CNN with generated vectors to do text generating [19], researchers successfully generated some realistic sentences. Another work, moreover, established a model that directly researches generating sentences from a continuous space using GAN [5], which is also a progress for GAN on NLP.

Traditional GAN and its applications only receive noise as input and produce output we expected in generator. We cannot decide which class to output if the expected output contains many categories. As solution, Convolutional Generative Adversarial Networks (CGAN) [4], are designed. It adds conditions as part of input both in generator and discriminator. Therefore, GAN can map conditions with expected output, and we are able to generate more accurate output

when we feed conditions while testing. [13] is a typical application of CGAN that takes descriptions as conditions and corresponding images as output. Then generator can generate figures according to the descriptive words we input.

WGAN, and its refined version WGAN with gradient policy, is improved forms of GAN that aim to solve the problem that traditional GAN is hard to train. The author analyzed the reason why GAN usually not converge while training, and makes some specific modifies. The modification is very successful. WGAN even works on discrete embedding such as one-hot vectors.

In our research, we attempt to utilize WGAN in sports news generating tasks. Due to the limited usage of GAN on NLP, we reduced the aim of WGAN from generating a whole sentence to generate several phrases whose expression may be different between news. This idea is applicable and reasonable. In sports news, there are many similar sentences. Therefore, we can safely ignore these repeated words and just aiming at the variational words, which are mostly the judgemental words that contain mostly concerned information. Through combining the output words and other constant information, such as team name and player name, our model can generate summary sentences and player performance sentences very similar to what will appear in a real sports news.

3 Method

In this section, we will illustrate the method to generate NBA news. There are two models to fulfil different tasks. One is Summary Sentence Generating Model (SSGM), and another is Player Performance Generating Model (PPGM).

3.1 Summary Sentence Generating Model (SSGM)

SSGM is designed to summarize the overall game situation. It tends to get overall basic information (date, teams and score) as input, and output the summary sentence for the whole match.

The summary sentences contain the most abstract but important information in a match. The structure of summary sentence is relatively stable and simple. However, the phrases are various according to how one team beats another. For example, if Houston Rockets beats Dallas Mavericks with 131-102, the summary sentence in news may likely be “Rockets in home curt whup Mavericks with 131-102”. “whup” is used here that is better than normal word “beat” so as to show that Rockets played an excellent game. These differences among words are often decided by news writers, or precisely, experts who can judge the expression by experience.

In our work, we would like SSGM to learn these expressions automatically, which will generate appropriate sentences without importing experts. Therefore, instead of producing the whole sentence using algorithm directly, which will be inefficient and unnecessary, we focus on generating the subjective phrases, such as “whup” in this instance. Then, combining the phrases and other constant information together into template, the SSMG finally output the generated summary sentence.

To fulfil this task, we need to design a generator that can generate phrases suitable for different situations. GAN, a powerful model that can train such a generator through confronting with a discriminator, is very popular in recent researches and produces impressive results. To overcome the problem that GAN is hard to converge while training, a conditional WGAN with gradient policy is used in SSGM.

Comparing to traditional GAN, which trains the discriminator to classify whether an input is real or fake, WGAN uses Wasserstein distance in discriminator to measure the difference between two kinds of inputs, which ensure the model will converge to a plausible result even for one-hot word embedding data. In addition, gradient policy method is utilized to stable the training process so that we can train the model without devoting energy on tuning the model for converging.

In SSGM, the most important factor that will influence the generated phrase is the score of each team. Therefore, a neural network is designed, with scores and noise as input and the phrase embedding as output. In discriminator, another neural network is designed to take scores and corresponding phrase vector as input and output a single value that refers how likely the input phrase and scores combination to be true.

Given:

- p_g : the probability distribution of generated phrase vector, calculated from generator function G , in SSGM, the function is represented as $G(s+z)$, where s is the score vector and z is the noise vector.
- p_r : the probability distribution of real phrase vector.
- $p_{\hat{x}}$: the probability distribution of randomly interpolation calculated by $\hat{x} = \epsilon x_r + (1 - \epsilon)x_g$, where $\hat{x} \sim p_{\hat{x}}$, $x_r \sim p_r$, $x_g \sim p_g$ and $\epsilon \sim Uniform[0, 1]$.

The loss function of WGAN in SSGM is:

$$L_G = -E_{x \sim p_g}[D(x + s)] \quad (1)$$

$$L_D = -E_{x \sim p_r}[D(x + s)] + E_{x \sim p_g}[D(x + s)] + \lambda E_{x \sim p_{\hat{x}}}[\|\nabla_x [D(x + s)]\|_p]^2 \quad (2)$$

Where L_G and L_D is the loss of generator and discriminator in WGAN with gradient policy, $D(x)$ is the function in discriminator.

While training WGAN, we try to minimize the two loss values alternatively until it reaches the equilibrium. Then the generator is expected to produce plausible result. After getting the phrases, we input them into Sentence Generator together with constant information to generate the final summary sentences. The constant information includes constant words include team name, score and home or away.

The Sentence Generator in SSGM is designed to combine constant information and generated sentences together according to some templates. For example, the template can be:

On [Date], [Team(A)] made a [Score(A)-Score[B] [SSGM phrase] [Team(B)].

Where $[Team(K)]$ is the name of team K. $[Score(K)]$ is the score that got by team K in the match, the $[Date]$ is the date that the match happened and $[Phrase]$ is the phrase that generated by WGAN. Figure 1 shows the flow chart of the SSGM.

Fig. 1. Flow chart of SSGM

In the flow chart, Generator takes team scores as input and generate score vectors, which is called condition. Then, concatenating with noise, they form the input for neural networks. The output of generator is phrase representation as embeddings that try to imitate what subjective phrase that an expert would like to employ for given scores. In discriminator, the neural network combines the phrase vector (real or fake) and score vector together as input, then output whether the discriminator believes the phrase is real or generated from generator.

3.2 Player Performance Generating Model (PPGM)

Unlike the overall description of match that the sentence structure is relative stable and simple, players' performance is more variable. For instance, there are ten starting players and several alternates in each basketball game. However, not all players will be shown in the news. The news writer tends to report the players who are more eye-catching, such as star players and who acts surprisingly well in that game. In addition, which data of one player is going to be reported (scores, rebounds, assists, etc.) is also worth concerning.

In PPGM, as shown in Fig. 2, we considered all aspects above. The basic structure of PPGM is also a WGAN with gradient policy. In short, the goal of WGAN is to determine which data of a player in this match is worth reporting, and which phrases or words should be used to describe these data.

Firstly, each player will be represented as a vector called player embedding. Therefore, the model can learn the phrase distribution for different players with different data. For example, if an up-and-coming youngster hit 20 scores in his first match, we should report it as “delivered a sensational”. However, if LeBron

Fig. 2. Flow chart of PPGM

James (a super star in NBA) got 20 scores, it is a normal data for him, we would more likely to use “scored” rather than “delivered a sensational”.

Furthermore, players’ performance will be represented as performance matrix, which combines player embedding and corresponding performance data as follow:

$$PM_{ij} = py_i^T \times pf_{ij} \quad (3)$$

where PM_{ij} the Performance Matrix of the i^{th} player in the j^{th} match; py_i is the Player Vector of i^{th} player and pf_{ij} is the Performance Vector of the i^{th} player in the j^{th} match.

Next, performance matrix will be used as condition in both generator and discriminator. The generator takes performance matrix and noise as input and outputs word/phrase embedding for each data. These phrases describe how the players perform in term of each data.

Additionally, there is a special embedding that represent “no report” for each data. If the model outputs this embedding with the highest probability, it means that this data of the player is not worth reporting in the news. Therefore, if all data of one player are outputted as “no report”, this player’s performance will not show in the news at all.

In discriminator, the input consists of performance matrix with real data or generated data together. The output of discriminator decides whether these phrases are in real news or just generated by the generator.

WGAN in PPGM is similar to that in SSGM, but with different conditions and generator output. The generator function is modified into $G(z + v_p^T v_s)$, where v_p is the player vector and v_s is the score vector got by corresponding players. Discriminator function is modified into $D([x + v_p^T v_s])$, where x is either the output of generator or real phrase vector.

The loss function of WGAN in PPGM is:

$$L_G = -E_{x \sim p_g}[D(x + v_p^T v_s)] \quad (4)$$

$$L_D = -E_{x \sim p_r}[D(x + v_p^T v_s)] + E_{x \sim p_g}[D(x + v_p^T v_s)] + \lambda E_{x \sim p_{\hat{x}}}[\|\nabla_x D(x + v_p^T v_s)\|_p]^2 \quad (5)$$

Since each player’s performance phrases are generated separately, the function of Sentence Generator in PPGM is to generate the whole paragraph for all corresponding players combining some constant information. Specifically, constant information contains all constant information such as player names, which team they belongs to and monotonous presentation. The Sentence Generator in PPGM is designed to combine constant information and generated sentences together according to specific rules. Different rules map to different templates so that the expressions are diversified and more similar to a real news. For example:

If one player got outstanding performance on two indexes such as scores and assists, it should be reported as ([Player(i)] [PPGM phrase] points and [PPGM phrase] assists). The connection word “and” is utilized to make sentence smooth.

As a result, the outputs of SSGM and PPGM constitute the paragraphs of a NBA news, which is accomplished automatically.

4 Experiments

In this section, we tested the two models based on real game stats.

4.1 Dataset

The data is collected from NetEase NBA website, which is one of the most popular NBA website in China. It contains full data in each match, including reports and the detailed statistic data for the match and each player. For example, the detailed statistic data and corresponding new report details provided in supplementary material appendix A.

For example, there is a match between Cleveland Cavaliers and Oklahoma City Thunder, the detailed statistic data is as shown in Fig. 3. The corresponding news report is also available as shown in Fig. 4. The summary sentence is shown in green box and players’ performance sentences are shown in red box.

The translation:

The Cavaliers scored 115-92 victory over Oklahoma City Thunder. Love scored 29 points, grabbed 11 rebounds and 4 assists, LeBron had 25 points, added 11 times Assists, 7 rebounds, 3 steals but had 5 turnovers, Jefferson got 15 points and 6 rebounds, Smith got 15 points, Thompson got 14 points and 14 rebounds, Mozgov got 11 points and 5 rebounds 15, rebounds and 3 assists, Westbrook gets 20 points, 11 assists and 9 rebounds, Durant finished 26 points, had 5 rebounds and 3 assists, Westbrook had 20 points, added 11 assists and 9 rebounds, Ibaka added 12 points.

The data will then be further processed for both two tasks in subsequent parts.

In the following subsections, to make non-Chinese readers easier to understand the news reports, we translate them into English for illustration.

克里夫兰骑士球员详细												
首发球员	时间	投篮	三分	罚球	前场篮板	总篮板	助攻	犯规	抢断	失误	封盖	得分
勒布朗-詹姆斯	40:02	9-20	2-6	8-12	3	15	8	4	0	8	2	28
JR-史密斯	28:12	1-4	1-2	0-0	0	0	0	1	0	2	0	3
凯文-勒夫	34:13	4-13	3-6	4-4	5	21	1	4	0	1	3	15
凯瑞-厄文	34:53	10-22	3-4	1-1	3	3	2	3	0	4	1	24
特里斯坦-汤普森	22:25	0-3	0-0	0-0	3	4	2	2	0	1	0	0
替补球员	时间	投篮	三分	罚球	前场篮板	总篮板	助攻	犯规	抢断	失误	封盖	得分
理查德-杰弗森	14:48	3-6	0-2	3-4	1	4	0	5	0	2	0	9
Dahntay Jones	04:06	1-2	1-2	4-4	0	0	0	0	0	0	0	7
凯尔-科沃尔	19:43	0-3	0-3	0-0	0	4	1	0	0	0	0	0
詹姆斯-琼斯	02:16	0-1	0-1	0-0	0	0	0	0	0	0	0	0
钱宁-弗莱	00:00	0-0	0-0	0-0	0	0	0	0	0	0	0	0
德隆-威廉姆斯	18:30	0-4	0-2	0-0	0	3	1	3	0	2	0	0
艾曼-舒伯特	16:46	2-6	1-3	0-0	0	5	0	1	0	0	0	5
德里克-威廉姆斯	04:06	0-2	0-0	0-0	0	0	0	0	0	0	0	0
凯-费尔德	00:00	0-0	0-0	0-0	0	0	0	0	0	0	0	0
沃尔特-塔瓦雷斯	00:00	0-0	0-0	0-0	0	0	0	0	0	0	0	0
总计	-	30-66	11-31	20-25	15	59	15	23	0	20	6	91
命中率	-	34.9%	35.5%	80%	-	-	-	-	-	-	-	-

Fig. 3. Match stats

4.2 Experiment Settings and Results

In this subsection, we will describe experiment settings and analyze the results.

SSGM Settings. In SSGM, WGAN model takes scores as input, and generates the phrases that describe how one team beat another team. We extract the summary sentences from news reports and related data. Then we process and organize them into form as shown in Table 1.

In this experiment, two scores are both duplicated 10 times into 10 dimensionalities and normalized to guarantee that WGAN will fully use the score information. Then the scores vector are concatenated together to form the conditions. In generator, conditions are concatenated with 5 dimensionality noise vector as the input. Meanwhile, the output phrases are embedded using one-hot embedding method.

阿杜38+8勇士屠骑士1-0 库里28+10詹皇空砍28+15
 来源: 网易体育 作者: 小柳 有 145472 人参与 手机看赛事

网易体育6月2日报道:

金州勇士队在总决赛中赢得开门红。虽然勒布朗得到28分、15个篮板和8次助攻,但杜兰特更为抢眼拿到38分、8个篮板和8次助攻,库里得到28分和10次助攻,他们至队第三节进攻发力确立大比分优势,勇士队在总决赛首场主场以113-91击败克里夫兰骑士。勇士队以1-0领先,总决赛的第二场将继续留在奥克兰进行。

17/98 勒布朗防守库里。

勇士队的杜兰特得到38分、8个篮板和8次助攻,库里得到28分、10次助攻、6个篮板和3次抢断,格林得到9分和11个篮板,汤普森16投3中,得到6分和4次助攻。骑士队的勒布朗得到28分、15个篮板、8次助攻和8次失误,厄文得到24分和4次失误,勒夫得到15分、21个篮板和3次盖帽。

Fig. 4. News report (Color figure online)

In discriminator, conditions vector are concatenated the phrase vector together to form the input, and get one value in $[0, 1]$ to indicate whether the discriminator believe the condition and phrase combination comes from real news.

After training, the generator can generate phrases given two scores. SSGM then put it together with other information into templates. Then we can get the summary sentence.

Table 1. SSGM training dataset

ID	Teams	Score	Phrase	Date
1	Boston-Los Angeles	119-113	beat	2.20
2	Miami-Oklahoma City	115-92	whup	2.22
3	Houston-San Antonio	114-117	lose by a neck	2.24
4	Golden State-Cleveland	121-118	edged out	2.28
...	...	...	...	...

SSGM Result. Given the data from NBA game in 2016-2-22, Cavaliers vs Thunder, Table 2 lists several results generated from our approach, compared to the news on website.

Table 2. SSGM generated result compared with real report

Score	Category	Sentence
115-92	Generated	Cleveland Cavaliers made a 115-92 beat the Golden State on February 22
		Cleveland Cavaliers made a 115-92 whup the Golden State on February 22
	Reported	Cleveland Cavaliers made a 115-92 beat the Golden State on February 22

From the result, it should be noticed that the generated summary sentence is amazingly similar as the real news. It is scarcely possible for people to distinguish whether the report is generated automatically or written by sports-writers. It shows that WGAN in SSGM learned the pattern that reports the summary result for different score combination, without an expert getting involved.

PPGM Settings. In PPGM, situations are more complicated than SSGM. The data is processed as shown in Table 3. First, we set each players' vector as length 50 with random value within $[-1, 1]$. Next, through statistic, only 5 kinds data are shown in the player performance sentence. They are "scores", "rebounds", "assists", "mistakes", "first start". Among the 5 data, the first 4 are all integers and will be normalized. On the other hand, the "first start" data is Boolean, which 1 represent first start and 0 otherwise. Therefore, the performance information is a vector of length 5.

For each kind of data, we employ one-hot embedding to represent corresponding frequently used phrases. We embedded each data output as a vector of length 5 with the last dimension representing "no report". Therefore, the output of generator in WGAN is a matrix of 5×5 , which phrases of all data generated at once. The discriminator form the input as the concatenation of Performance Matrix

Table 3. PPGM training dataset

Player ID	Rebound	Assist	Turnover	Points	Starter
1	11 (grabbed)	4	0	29 (scored)	1
2	7 (had)	11 (had)	5 (but had)	25 (had)	1
3	1	7 (added)	3	2	0
4	6 (got)	0	3	15 (got)	0 (sub)
5	5 (had)	3 (had)	3	26 (finished)	1
...	...	...	...	...	...

and phrase embedding matrix then output a single value representing whether it believes the Performance Matrix and phrase embedding matrix comes from real news.

After training, in each game, each player will be evaluated whether his performance is worth reporting and which phrases should be reported. Finally, adding all the results from WGAN together with corresponding name into template. PPGM will generate the player performance sentences using the rules for all reported players.

PPGM Result. Given the data from NBA game in 2016-2-22, Cavaliers vs Thunder, we finally got the generated paragraph shown in Table 2.

From the results, we can see that the player performance sentences are clear and plausible. Furthermore, the sentence generator leveraged various templates and rules to generate the whole paragraph. It fully utilizes the input data and prevent monotonous description for all players. This brings even better effect than the original news.

Among all the input data, PPGM chose what kind of data to report automatically. All the chosen data are prominent and worth reporting. For example, Tristan Thompson got 14 rebounds, which is amazing in NBA match. our model reported it as “Tristan Thompson grabbed 14 rebounds”. For another example, Kyrie Irving only got 2 scores in the match. However, PPGM still reports him because our model can learn that he is NBA star concerned by many fans, based on his history performance as represented as Performance Matrix. And the corresponding report, “Kyrie Irving performed poorly, 5 hit 1, and only got 2 scores”, gives the vivid description about his performance (Table 4).

In sum, PPGM takes all players’ data in a match and capable of deciding which data worthy reporting and generating diverse description of players’ performance.

Based on the both results of SSGM and PPGM, we are confident to extend our model to replace this part from human writer and save the time to generate NBA news.

Table 4. PPGM generated results compared with real reports

Category	Result
Generated	Cleveland Cavaliers blasted enemy in the game. Love and James Combined 54 points, Thompson grabbed 14 rebounds and scored 4 of 6 shooting. Smith finished with 15 points, with 5 of 12 shooting 5-12. Dellavedova was substituted and added 7 assists, shotted 1-3. Jefferson was substituted and scored 15 points on 4 of 5 shooting. Irving performed murky and only had 2 points on 1 of 5 shooting among the playing time of 9:16. Meanwhile, Oklahoma City Thunder fails to defend on their home curt. Durant and Westbrook combined 46 points. Ibaka scored 12 points on 6 of 14 shootings. Adams grabbed 9 rebounds and hit 4 out of 5. Cavaliers had blossom in the game and finished with six players in double figures. On the other hand, Thunder lose their hit rate from behind the arc and only had 22.7%
	Cleveland Cavaliers played a wonderful match. Headed by LeBron-James scored 25 points, 11 assists, 7 rebounds, but also 5 turnovers, Love got the team's highest 29 points and contributed 11 rebounds and 4 Assists, Tristan-Thompson grabbed 14 rebounds. Richard-Jefferson was Substituted and performed outstanding of having 15 points and 6 rebounds. Matthew-De La Vitoria off the bench sent 7 assists. Kerry-Erwin started but only 2 points. Oklahoma City Thunder fails to defend on their home court. Kevin-Durant finished a game-high of 26 points and sent 5 assists, Russell-Westbrook almost got three double and add 20 points, 11 assists and 9 rebounds. Cavaliers finished with six players in double figures
Reported	The Cleveland cavaliers won the game, Kevin love had 29 points and 11 rebounds, and lebron James scored 25 points, 11 assists 7 rebounds and 3 steals, but there are 5 turnovers, Tristan Thompson grabbed 14 rebounds. JR. Smith scored 15 points. Substitute Matthew delevedo sent out seven assists. Substitute Richard Jefferson scored 15 points. Kerry Ervin played nine minutes and 16 s and scored 1 of 5, just 2 points. The Oklahoma city thunder team lost home, Durant had a team-high 26 points and five rebounds and three assists, Westbrooke had 20 points and 11 assists and 9 rebounds, Serge Ibaka had 12 points. Stephen Adams picked up nine rebounds. The Cleveland cavaliers scored double with many people

5 Conclusion and Future Work

In this work, we propose an Automatic Sport News Generating System, which aims to produce sport news immediately after each match is over. We utilize WGAN combining with template to generate the summary sentences and player performance sentences in NBA match news. This is the first work that applied GAN on automatic sport news generation field. The system not only fulfil the task of generating sports news based on WGAN model, but also gives new angle of leverage GAN on NLP area.

As to future work, we will continue generating the remaining of NBA match news. For the abstract description of the whole match part, we will train WGAN model on sequence of data, which is a new attempt that combines GAN and sequence model together on NLP tasks.

References

1. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems (2014)
2. Mikolov, T., et al.: Efficient estimation of word representations in vector space. arXiv preprint [arXiv:1301.3781](#) (2013)
3. Zhang, J., Yao, J.-g., Wan, X.: Towards constructing sports news from live text commentary. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), vol. 1 (2016)
4. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint [arXiv:1411.1784](#) (2014)
5. Bowman, S.R., et al.: Generating sentences from a continuous space. arXiv preprint [arXiv:1511.06349](#) (2015)
6. Denton, E.L., Chintala, S., Fergus, R.: Deep generative image models using a Laplacian pyramid of adversarial networks. In: Advances in Neural Information Processing Systems (2015)
7. Mathieu, M., Couprie, C., LeCun, Y.: Deep multi-scale video prediction beyond mean square error. arXiv preprint [arXiv:1511.05440](#) (2015)
8. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint [arXiv:1511.06434](#) (2015)
9. Glover, J.: Modeling documents with generative adversarial networks. arXiv preprint [arXiv:1612.09122](#) (2016)
10. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax. arXiv preprint [arXiv:1611.01144](#) (2016)
11. Ledig, C., et al.: Photo-realistic single image super-resolution using a generative adversarial network. In: CVPR, vol. 2, no. 3 (2017)
12. Maddison, C.J., Mnih, A., Teh, Y.W.: The concrete distribution: a continuous relaxation of discrete random variables. arXiv preprint [arXiv:1611.00712](#) (2016)
13. Reed, S., et al.: Generative adversarial text to image synthesis. arXiv preprint [arXiv:1605.05396](#) (2016)
14. Salimans, T., et al.: Improved techniques for training GANs. In: Advances in Neural Information Processing Systems (2016)
15. Nowozin, S., Cseke, B., Tomioka, R.: f-GAN: training generative neural samplers using variational divergence minimization. In: Advances in Neural Information Processing Systems (2016)
16. Tang, D., Qin, B., Liu, T.: Aspect level sentiment classification with deep memory network. arXiv preprint [arXiv:1605.08900](#) (2016)
17. Chen, X., et al.: InfoGAN: interpretable representation learning by information maximizing generative adversarial nets. In: Advances in Neural Information Processing Systems (2016)
18. Yu, L., et al.: SeqGAN: sequence generative adversarial nets with policy gradient. In: AAAI (2017)

19. Zhang, Y., Gan, Z., Carin, L.: Generating text via adversarial training. In: NIPS Workshop on Adversarial Training, vol. 21 (2016)
20. Zhao, J., Mathieu, M., LeCun, Y.: Energy-based generative adversarial network. arXiv preprint [arXiv:1609.03126](https://arxiv.org/abs/1609.03126) (2016)
21. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein GAN. arXiv preprint [arXiv:1701.07875](https://arxiv.org/abs/1701.07875) (2017)
22. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* **8**(3–4), 229–256 (1992)
23. Goldberg, E., Driedger, N., Kittredge, R.I.: Using natural-language processing to produce weather forecasts. *IEEE Intell. Syst.* **2**, 45–53 (1994)
24. Coch, J.: System demonstration interactive generation and knowledge administration in MultiMeteo. *Natural Language Generation* (1998)
25. Coch, J., David, R., Magnoler, J.: Quality tests for a mail generation system. In: IA 95. Journées internationales (1995)
26. Portet, F., et al.: Automatic generation of textual summaries from neonatal intensive care data. *Artif. Intell.* **173**(7–8), 789–816 (2009)
27. White, M., Caldwell, T.: EXEMPLARS: a practical, extensible framework for dynamic text generation. *Natural Language Generation* (1998)
28. Busemann, S., Horacek, H.: Generating air quality reports from environmental data. In: *Proceedings of the DFKI Workshop on Natural Language Generation* (1997)