Textual Summarization of Time Series using Case-based Reasoning: A Case Study

Neha Dubey, Sutanu Chakraborti, Deepak Khemani

Department of Computer Science and Engineering
Indian Institute of Technology Madras, Chennai - 600036
{nehamay,sutanuc}@cse.iitm.ac,{khemani}@iitm.ac.in

Abstract. In this paper, we propose an end-to-end Case-Based Reasoning (CBR) approach for generating the textual summaries of time series data in the weather domain. Our approach is intended to assist the user in decision making to generate the summary of a given time series. For that, the approach first generates appropriate abstractions of a time series and then generates the textual summary for each of the abstractions. Empirical results show the effectiveness of the approach in generating suggestions very close to human-authored summaries in the majority of the cases.

Keywords: Time Series, Natural Language Generation, CBR.

1 Introduction

Data-to-text generation is a subfield of Natural Language Generation(NLG), which deals with generating the textual descriptions of non-linguistic data sources such as time series, and event logs. The subtasks in an NLG system involves selecting the relevant information from input data and organizing it coherently (content selection), making relevant choices to express the relevant information (micro planning), and rendering the information in the output text (surface realization) [6]. In this work, we focus on summarization of time series using text in weather domain. Textual summarization of a time series requires an understanding of the time series in the context of the underlying process that generates the time series as well as the end user's requirements. For example, an expert might identify significant patterns in a time series based on her experience and provide an explanation for the same, which a non-expert might find hard to arrive at, by visual inspection alone. Furthermore, depending on the end user, the same time series can have different summaries, for example, summary for a sports news channel is different from one meant for farmers, and both are different from the forecast meant for trekkers in mountains.

Content selection in time series summarization involve finding the appropriate level of abstraction for a time series, which includes selecting change points in a time series. These change points along with the trend information can summarize the time series data. Once this is done, the later stages of an NLG system can choose appropriate linguistic realization to describe the information. One

significant issue in doing so is that of lexical choice. For example, an increasing trend in a time series can be described using *increasing*, *gradually increasing*, or *rising*. This, in particular, makes the evaluation of an NLG system hard as the there can be multiple correct textual summaries for a given time series.

In this paper, we propose an end-to-end Case-Based Reasoning (CBR) approach to assist the user in generating the textual summary of a time series in the weather domain. CBR works by recalling past experiences and is based on the premise that similar problems recur and have similar solutions. In our case, the experiences in the case library are the time series and their corresponding textual summaries. The incoming new time series is matched against the cases in the case library to retrieve relevant similar cases and the textual summary of the relevant cases is reused to generate the summary for the new time series.

In the past, researchers have proposed other approaches for generating weather forecast text summary for a time series. These systems can be mainly categorized in two ways: knowledge-rich (top-down) [11], and knowledge-light (bottom-up or data-driven) [3, 4]. While it is expensive and time-consuming to acquire knowledge in top-down system, knowledge-light systems need large and reliable parallel corpora of input and output text. The system by Sripada et al. [11] is a top-down system that generates the forecast text by following the pipeline architecture of an NLG system, which includes all the subtasks in NLG. Most of the other systems that generate weather forecast text are not complete NLG systems as these systems focus only on micro-planning and realization and not on content selection [3, 4].

To the best of our knowledge, ours is the first attempt to propose an end-to-end CBR system that performs all subtasks in an NLG pipeline from content selection to text realization. In addition to that, the proposed system generates multiple summaries rather than just one to account for the fact that there can be multiple correct summaries for a given time series. This is consistent with our goal of assisting a human user, who can select one of the outputs and, if necessary post-edit to produce the final summary.

2 Our Approach

The architecture of our system follows the typical pipeline architecture of an NLG system: Content Generation (CBR_1) and Text Generation (CBR_2). The first component decides the abstraction level of a wind time series, which is used to generate the intermediate representation of that time series and the second component generates the textual description of the given intermediate representation. Figure 1 shows the system's architecture. Two components each of which is a CBR system are illustrated in Figures 2 and 3, respectively.

Time Series Summarization Weather data consists of day-wise wind time series and the corresponding textual summaries. The time series is a multivariate time series as it involves two channels, namely, wind speed and wind direction. A sample time series in weather domain is shown as input in Figure 1. The human-written summary for the given time series is "S 02-06 increasing 16-20", which

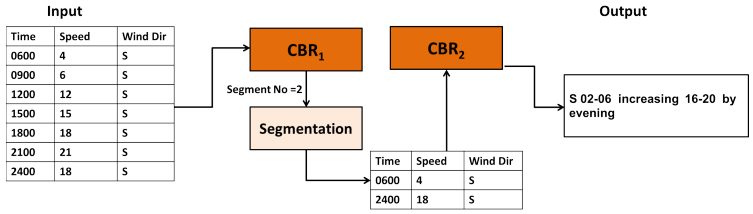

Fig. 1: Architecture for text generation system

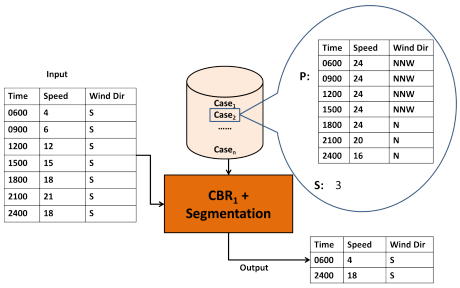

Fig. 2: CBR_1 Case representation
Problem: Wind time series; Solution: Number of segments

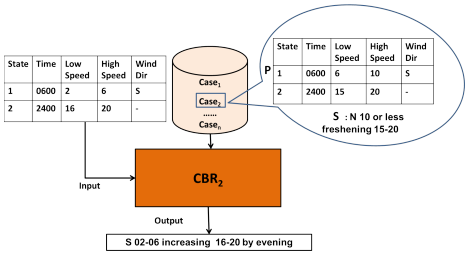

Fig. 3: CBR_2 Case representation
Problem: Intermediate representation of the wind time series; Solution:Weather forecast text

describes the change in wind speed as increasing from 4 (02-06) knots to 18 (16-20) knots. These weather forecasts are produced to assist offshore oil company staff in making right decisions for the tasks that depend upon the weather, for example, to carry out the supply boat operation.

Content generation from time series involves a trade-off between minimizing the number of change points reported and maximizing the faithfulness of its approximation. The error in approximation can be reduced by increasing the length of the summary (by increasing the number of change points reported). In the proposed system, we predict the number of change points in a time series using CBR_1 and then generate the approximation to the time series that minimizes the approximation error. To approximate a given time series, we use optimal segmentation algorithm [2]. Segmentation is the process of approximating a time series with straight line segments. For example, Figure 4 shows the segmentation of a time series with five segments. Given the number of segments, the optimal segmentation algorithm globally minimizes the error of approximation.

In the weather domain, a calm day with fewer fluctuations may need a lower number of segments than a bad weather day. This is because the more the weather is volatile, the more is the number of segments that will be abstracted out. For example, the text in Figure 1 has two wind states (S, 02-06) and (-, 16-20). The count of wind states (as the number of segments) is used as input to the segmentation algorithm to generate the intermediate representation.

2.1 Generating Intermediate Representations

In the weather domain, based on the observation that days with similar weather conditions have similar forecast text and the similar level of abstraction in time

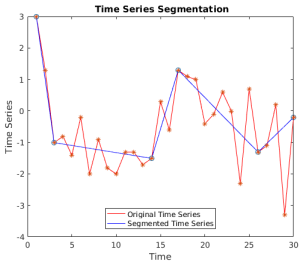

Fig. 4: Segmentation example

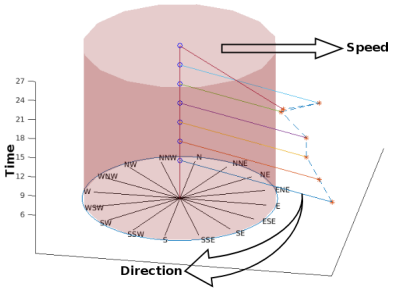

Fig. 5: Wind vector representation of wind time series

series, we hypothesize that *similar wind time series have the similar number of segments in the forecast text*. Thus, the case used in CBR_1 has wind time series as the problem component and number of segments as the solution component. The representation of wind time series is the wind vector (wind speed as magnitude and wind direction as direction of the wind vector) representation as shown in Figure 5. This choice of representation of time series allows us to consider two univariate time series: wind speed and wind direction into a single time series with a wind vector at each time stamp.

Retrieval In an NLG system, for a time series, the relevance of a case to a query depends upon the end user requirements. For example, a wind time series on a calm day can have similar behaviour in terms of patterns on a bad weather day, however, both have entirely different interpretations with respect to the end user. Thus, a case is relevant to a query if it has the patterns similar to that in the query time series and the similar error tolerated in approximating a case and the query. To retrieve the cases with patterns similar to the query, we use the dynamic time warping (DTW) [8] on wind vector time series. In order to retrieve the cases with the similar error of approximation, we define a distance measure called $Error_{distance}(query, case)$ between a query and a case.

DTW aligns two time series by scaling/shrinking on time axis. In our case, since the time series have wind vector representation, the equation of DTW similarity for two time series T_1 and T_2 can be modified as :

$$DTW(i,j) = \min\{DTW(i-1,j), DTW(i,j-1), DTW(i-1,j-1)\} + vectordist(i,j) \quad (1)$$

where $DTW(1,1)$=$vectordistance(1,1)$ is the distance between first point of both time series, and i and j are the time indices in time series T_1 and T_2, respectively. The local distance measure $vectordist$ in our case is defined as $vectordist(i,j) = \sqrt{s_{T_{1i}}^2 + s_{T_{2i}}^2 - 2 * cos(d_{T_{1i}} - d_{T_{2i}})}$, where $s_{T_{1i}}$ and $d_{T_{1i}}$ denote speed and direction at time i of time series T_1. Thus, the DTW distance between a query and a case is $dist_{dtw}(query, case) = DTW(n,n)$,where n is the length of time series.

$Error_{distance}(query, case)$ denotes that a case is relevant to a query if both case and query have the similar error of approximation for the given choice of segmentation. To define $Error_{distance}(query, case)$ we define two terms: the error of approximation of a query, i.e., $Error_{query}$, and error of approximation of a case, i.e., $Error_{case}$. $Error_{query}$ with respect to a case is the error between query time series and its segmented intermediate representation when the query is seg-

mented with the segment number as in the case. $Error_{case}$ is the error between the problem component of a case (wind time series) and its intermediate representation as used in generating the textual summary of the case. Thus, $Error_{case}$ gives the error as tolerated by a human forecaster while writing the textual summary of a time series. To get the intermediate representation for a time series corresponding to the forecast summary, we reconstruct a time series from the text by using text to time series mapping, where each entry in the text is mapped to some entry in the time series. For example, the forecast text in Figure 1 contains two entries (02-06, S), and (-,16-20) corresponding to the times 6:00, and 24:00 in the time series, respectively. The rest of the time series values between 6:00 and 24:00 are obtained by interpolating between these two values in the text. Now, we define the $Error_{distance}(query, case)$ as:

if $Error_{case} < Error_{query}$(Case is relevant to query) **then**
$Error_{distance}(query, case) = \alpha * |Error_{query} - Error_{case}|$
else
$Error_{distance}(query, case) = \exp(\beta * |Error_{query} - Error_{case}|)$
end if
The parameter α, β are set using cross validation.

Finally, the total distance between a case and query can be defined as

$$dist(query, case) = dist_{dtw}(query, case) + Error_{distance}(query, case) \quad (2)$$

The corresponding similarity can be defined as$1/(1 + dist(query, case))$. At the end, the cases with maximum similarity with the query are retrieved.

Multiple Representations of a Query Typically, an expert forms an "overall" view of a time series using her experiences in the process of generating content for it. These views are often tacit and can vary across experts. Further, each of the intermediate abstractions of a time series can be used to generate a textual summary. Therefore, in the proposed approach, the CBR_1 system generates multiple appropriate abstractions of a time series, each with a certain confidence value. Next, the CBR_2 generates the textual summary for each of these abstractions of a time series.

Confidence Scores Let the retrieved cases for a query be the set $C_1, C_2, C_3, .., C_m$ with the corresponding segment counts as $k_1, k_2, k_3, ..., k_m; \forall k_i < K$, where K is the maximum segment count possible for a time series. Let the similarity of retrieved cases in the set with be $s_1, s_2, s_3, ..., s_m$, then the confidence associated with each representation of query becomes the weighted mean of the segment counts in the retrieved cases, i.e., for $k = 1, 2, 3, ...K$

$$confidence(k) = \frac{\sum_{i=1}^{m} s_i * I\{k_i = k\}}{\sum_{i=1}^{m} s_i} \quad (3)$$

where $I\{k_i = k\}$ is an indicator function, which is 1 when k_i is equal to k, else 0. For each segment count k, for which the confidence score is greater than a threshold value, we generate the intermediate representation. For that, we use optimal segmentation algorithm by Bellman [2]. The algorithm takes the input as the number of segments and outputs the approximated (segmented) time series which has the least possible error of approximation. Let the set of such query

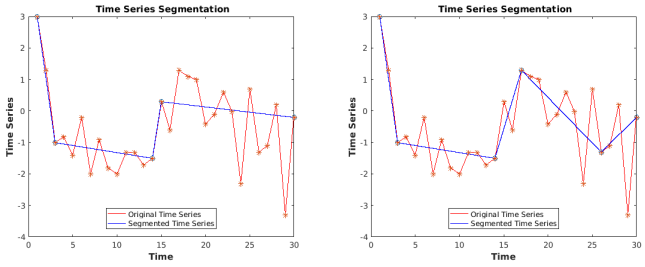

Fig. 6: Time series representations with multiple levels of abstractions

representations be $Q_{=}\{Q_k\}$, where $k < K$. For illustration, Figure 6 shows time series representations for a given time series, each with four and five segments.

2.2 CBR_2: Text Generation

Once we have the intermediate representations for a time series, we generate the forecast text corresponding to each representation using a similar approach as in [1]. The case library used in this step as shown in Figure 3, with the intermediate representation of time series as the problem side of a case and the textual summary as the solution side. Thus, for each of the query representations in $Q_{=}\{Q_k\}$, where $k < K$, the approach finds the cases which have exact patterns in speed and direction as in the text. The patterns for speed are *Increasing, Decreasing, and Stable*, and for direction *Backing(anticlockwise) and Veering(clockwise)*. For example, for query representation Q_2 with two segments, we retrieve those cases where the number of segments in the text is two. Next, among these cases, the cases are retrieved in which the patterns in speed and direction are same as the patterns in the query representation Q_2. Finally, among the retrieved cases with exact patterns, cases with the closest distance to the query are retrieved. The distance is the average vector distance between the query representation Q_k and a case C. This distance can be defined as $dist(Q_k, C) = \sum_{i=1}^{k} vectordist(Q_{ki}, C_i)/k$, where i an k are the time index and the length of the time series Q_k and C, respectively. Let the set of retrieved cases for each query representation in Q_k be the set $C_k = \{C_1, C_2, ..., C_m\}$ and the corresponding similarity with the query representations be $\{sim_1, sim_2, ..., sim_m\}$, where $sim_j = 1/(1 + dist(Q_k, C_j))$. Finally, all the cases retrieved for all query representations can be denoted as $Candidate\_Cases = \bigcup_{i=1}^{k} C_k$, where $k < K$.

Ranking of candidate cases The final ranking of the cases is done by using confidence score for the query representation for which the case is retrieved and the similarity of the case with that query representation. Thus, the final relevance score of a case $C_j \in Candidate\_Cases$ with respect to a query is $confidence(k) * sim_j$.

Text generation Among the top relevant cases in the ranked list of candidate cases $Candidate\_Cases$, the solution component of the cases, i.e., the textual summary is reused to generate the final textual summary for each query representation. Since the patterns in the case and query are same, we replace the

corresponding speed and directions in the retrieved text to generate the summary for the query representation. Figure 3 for CBR_2 shows the example of the text reuse as the text in the case *N 10 or less freshening 15-20 by evening* is reused to generate the summary of the incoming time series as *S 02-06 freshening 16-20 by evening* by replacing (N, 10 or less), (-,15-20) with (S, 02-06), and (-,16-20), respectively.

3 Experiments

We have used the SUMTIME-MAUSAM parallel corpus [11] of 1045 numerical weather data and human written forecasts where we take only wind time series. All results are obtained by performing 3-fold cross-validation with 90-10 split.
Evaluation Measure *1. Modified Edit Distance* Ideally, the evaluation of an NLG system is either human evaluation in the form of ratings given by humans for text quality or task-based evaluation(extrinsic) when the system is deployed in the real world. However, human-based evaluation is costly and not always feasible in absence of experts. Therefore, we measure the performance of our system by measuring how semantically close the generated text is to the actual human-authored summary by using modified edit distance measure. Metric-based evaluations (NIST, ROGUE) are not appropriate for our approach since we have only one reference text for a generated text.

The Levenshtein distance between two texts, $Text1$ and $Text2$ with lengths n and m, respectively can be defined as [5]:

$$L(i,j) = \min\{L(i-1,j)+1, L(i,j-1)+1, L(i-1,j-1)+c_I(a_i,b_j)\} \quad (4)$$

where $0 \leq i \leq n$, $0 \leq j \leq m$, and the Indicator function $c_I(a_i, b_j)$ can be defined as $c(a_i, b_j) = \{1 - sim(a_i, b_j)$ where $sim(a_i, b_j)$ is the similarity between words a_i and b_j in $Text1$ and $Text2$, respectively.

We modify the above measure in our case, instead of words in the text, we take semantic units in the text like speed, direction, patterns in speed and direction, time phrase. Here, we use the pre-processed parsed text available with the parallel corpus [11] to remove any other information present. Next, the similarity between these semantic units can be defined as follows:
Similarity between two values of wind speed and direction In weather forecasts, forecasters write ranges instead of numbers from time series, for example, "20-25" instead of 22 knots for a value of speed. Since our system selects points from time series, it generates the single number for the value of the speed. Therefore, we calculate the similarity between the single value of speed with the range in reference text as the $1 -$ distance from mid-value in the range. For direction, if it is similar to the ground truth text, the value of similarity is 1 or else it is 0.
Pattern Similarity Patterns in speed and direction can be summarized using similar words, for example, a rising pattern in a time series can be described using *Increasing, Rising, Freshening*. These choices are author dependent [7]. For illustration, we use synonym similarity values as shown in Table 1. These values, however, should be given by domain experts.

Time phrase Similarity For time phrase matching, we use the Jaccard similarity (common words divided by total words) between the generated time phrase and the time phrase present in ground truth text. We believe that this case can be further improved by knowing how authors use a time phrase, for example, in some cases, "0900" is used as "morning" while in some cases as "midday". Using the above similarities, the edit distance between the generated text and the ground truth text can be calculated using equation 4 where a_i and b_j are semantic units in generated text and the ground truth text, respectively. Next, the similarity between the generated text and ground truth text can be defined as $sim(text_{generated}, text_{groundtruth}) = 1 - L(i,j)/max(n,m)$ where n and m are numbers of semantic units in generated text and in the ground truth text, respectively and $0 \leq i \leq n$, $0 \leq j \leq m$.

2. Relevance Factor: Evaluation of generated multiple output texts We define the relevance of a generated text as follows:

if $sim(text_{generated}, text_{groundtruth}) > Threshold$ **then**
  $relevance(text_{generated}) = 1$
else
  $relevance(text_{generated} = 0$
end if

For a query Q, the system generates a ranked list of texts, If any of the generated texts in the ranked list is relevant with respect to the ground truth, we say the ranked list is relevant. Over a set of queries, the Relevance Factor can be defined as the average number of the queries for which the $Ranked_{list}$ is relevant.

3. Mean Average Precision (MAP): Evaluation of generated multiple output texts The average precision for a single query Q is the mean of the precision obtained in the $Ranked_{list}$. The mean average precision for a set of queries is the mean of the average precision scores of each query.

Experiment Design Within our knowledge, there is no existing work for direct comparison with our system. The earlier approach by Sripada et al. [10] evaluated using post edit data, where forecasters edited the generated text. The counts of the edits were used for measuring the performance of the system. Therefore, we compare our system with the following configurations: First, we change the similarity measure in the first module, i.e., CBR_1 to generate the intermediate representation of a query time series as shown in Figure 2. The evaluation measure for CBR_1 is the accuracy of correctly predicting the number of segments. Second, while keeping the configuration of CBR_1 fixed, we change the output configuration of second CBR, i.e., CBR_2: First, when the system generates one textual summary; Second, when the system generates a ranked list of summaries. To generate the single output for a query time series Q, we take the estimated segment count k by CBR_1 and segment the query time series, resulting in an intermediate representation Q_k. Now, input to the second system is Q_k, which outputs the textual summary for Q_k. To generate the multiple outputs of a query time series Q, we generate different segmented version of Q, $\{Q_k\}$, where $k < K$, and the confidence value for each representation is provided by CBR_1 using equation 3. Next, CBR_2 generates the textual summaries for each of the representations. The evaluation measures used here are

Table 1: Synonyms Similarity

Word1\Word 2	Increasing	Rising	Freshening
Increasing	1	0.8	0.8
Rising		1	0.8
Freshening			1

Table 2: Result for various configurations of the CBR system

Sl No.	CBR 1 Configurations	Accuracy of Segment	CBR 2 Configurations			
			Single Output Text		Multiple Output Text	
	Similarity	Prediction (%)	MAP Score	Relevance Factor	MAP Score	Relevance Factor
1	DTW	55.91	0.59	0.59	0.55	0.80
2	DTW +Error Distance	60.57	0.61	0.61	0.62	0.80

Mean Average Precision (MAP) and Relevance factor. The results of both the configurations with a relevance threshold of 0.4 are shown in Table 2.

4 Results and Discussion

We analyzed our generated textual summary with respect to the ground truth textual summary based on the level of abstraction for a time series, i.e., number of segments and the similarity with the ground truth summary.
Case 1 (System works as expected): When the estimated level of abstraction for a time series is same as of ground truth and the generated text is similar to the ground truth summary. Since near synonym choices used in the text such as *Backing* and *Becoming* are mostly author dependent, we accommodate this in our evaluation measure by taking synonyms similarity.
Case 2: When the estimated level of abstraction for a time series is same as of ground truth and the generated text is not similar to the ground truth summary. For example, if the actual text is *SE 25-30 backing SE-ESE 20-25* and the generated text is *SSE 25.0 easing later to 23.0*. In the summary, the verb type is determined by the change in wind speed, and direction, i.e., a speed verb is chosen if the change in speed is more significant than the change in direction and vice versa. We notice that humans elide verb information and their text is shorter. Therefore, the correct level of abstraction, i.e., number of segments does not guarantee the high similarity of generated text with the ground truth text.
Case 3: When the estimated level of abstraction for a time series is not same as of ground truth and the generated text is similar to the ground truth summary. This case is interesting, because, even though the intermediate content selection is not correct, the final text is similar to the actual text. For example, if the actual text is *S-SW 18-22* and the generated text is *SW 18.0 increasing SSW 21.0*, when we looked at the time series, we found that the change in speed, which is increasing from 18 to 21 in this case, is ignored by forecasters as they wrote only ranges (*18-22*,*S-SW*). However, in the multiple text configuration of our system, a forecaster can choose the summary, which she prefers.
Case 4: When the estimated level of abstraction for a time series is not same as of ground truth and the generated text is similar to the ground truth summary. In this case, our system does not perform as expected. For example, if the actual text is *SW 30-35 rising 38-42 by afternoon/evening and later veering W'LY 25-30* and the final text is *SW 31.0 veering W 26.0 in the evening*, we lose most of the trend information of a time series in the generated text. We suspect that these cases are harder cases and need more domain knowledge. In future, we plan to store such cases as the exceptional cases so that the system can flag a warning message for an external review if any of the exceptional case is reused.

Further, we made some general observations for time series summarization:
1. The data analysis techniques to summarize time series need to be adapted according to the domain and end-user requirements. For example, in the medical domain, artefacts, anomalous spikes are more important, while in the weather domain, trends are more important. This knowledge can be integrated into various forms: for example, we learn the number of segments using available data, or we can use distributional measures like word2vec and wordnet on a large parallel corpus to get the synonyms similarity to evaluate the system.
2. Adaptation of the CBR system can be further improved. For example, cases can be stored as segment wise and for a query time series, multiple cases can be retrieved for each segment in the query. Next, the retrieved segments of these cases can be reused to generate the textual summary.

5 Related Work

An earlier approach by Sripada [11] to generate the textual summary of time series in weather domain is a rule-based approach. The approach uses error thresholds provided by experts to find the appropriate abstraction of a time series (content selection). Next, to choose the appropriate words and phrases for the textual summary, the system uses rules induced from corpora and as provided by experts. The other approach by Sowdaboina [9] is a purely data-driven approach to select the content for a summary. It uses a neural network to select the representative points from a time series. The approach, however, considers only one channel to select the points and does not generate the final textual summary. The other text generation systems are MOUNTAIN [4] and (Probabilistic Context-Free Grammar) PCFG based system [3], and the CBR system by Adeyanju [1]. These systems do not perform content selection and focus on other NLG subtasks like microplanning and realization. The input to these systems is the time series reversed engineered from its textual summary, i.e., same as the problem component of a case in CBR_2.

6 Conclusion

In this paper, we presented an end-to-end CBR system for generating the textual summaries of time series in the weather domain. First, the system generates appropriate abstractions of a given time series with certain confidences for various abstractions. Later, it generates the textual descriptions of these abstractions. In future, we plan to analyze the CBR_2 in isolation with CBR_1 to specifically study the cases where a correct abstraction of a time series does not necessarily result in the generated textual summary similar to ground truth summary. Further, we plan to store some specific cases with poor alignment as exceptional cases in the case base. That means, in our case, a small perturbation in time series gives rise to an entirely different textual summary. This, in turn, has the cognitive appeal as humans also have few specific experiences combined with the abstract experiences formed over time.

References

1. Adeyanju, I.: Generating Weather Forecast Texts with Case based Reasoning. International Journal of Computer Applications **45**(10), 35–40 (2012)
2. Bellman, R.: On the Approximation of Curves by line segments using dynamic programming. Communications of the ACM **4**(6), 284 (1961)
3. Belz, A.: Automatic Generation of Weather Forecast Texts Using Comprehensive Probabilistic Generation-Space Models. Natural Language Engineering **14**(04), 431–455 (2008)
4. Langner, B., Black, A.W.: MOUNTAIN: A Translation-based Approach to Natural Language Generation for Dialog Systems
5. Miura, N., Takagi, T.: Wsl: Sentence similarity using semantic distance between words. In: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015). pp. 128–131 (2015)
6. Reiter, E.: An architecture for data-to-text systems. In: Proceedings of the Eleventh European Workshop on Natural Language Generation. pp. 97–104. Association for Computational Linguistics (2007)
7. Reiter, E., Sripada, S., Hunter, J., Yu, J., Davy, I.: Choosing words in computer-generated weather forecasts. Artificial Intelligence **167**(1-2), 137–169 (2005)
8. Sakoe, H., Chiba, S.: Dynamic Programming Algorithm Optimization for Spoken Word Recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing **26**(1), 43–49 (1978). https://doi.org/10.1109/TASSP.1978.1163055
9. Sowdaboina, P.K.V., Chakraborti, S., Sripada, S.: Learning to summarize time series data. In: International Conference on Intelligent Text Processing & Computational Linguistics. pp. 515–528 (2014)
10. Sripada, S., Reiter, E., Hunter, J., Yu, J.: Segmenting time series for weather forecasting. Applications and Innovations in Intelligent Systems X pp. 105–118 (2002)
11. Sripada, S., Reiter, E., Hunter, J., Yu, J.: Sumtime-meteo: Parallel corpus of naturally occurring forecast texts and weather data. Computing Science Department, University of Aberdeen, Aberdeen, Scotland, Tech. Rep. AUCS/TR0201 (2002)