Text2Chart: A Multi-staged Chart Generator from Natural Language Text

Md. Mahinur Rashid, Hasin Kawsar Jahan, Annysha Huzzat, Riyasaat Ahmed Rahul, Tamim Bin Zakir, Farhana Meem, Md. Saddam Hossain Mukta, and Swakkhar Shatabda(✉)

Department of Computer Science and Engineering, United International University, Dhaka, Bangladesh
{mrashid171045,hjahan171054,ahuzzat171034,rrahul171089,tzakir171032, fmeem171031}@bscse.uiu.ac.bd
{saddam,swakkhar}@cse.uiu.ac.bd

Abstract. Generation of scientific visualization from analytical natural language text is a challenging task. In this paper, we propose Text2Chart, a multi-staged chart generator method. Text2Chart takes natural language text as input and produces visualization as two-dimensional charts. Text2Chart approaches the problem in three stages. Firstly, it identifies the axis elements, known as x and y entities, of a chart from the given text. Next, it finds a mapping of x-entities with its corresponding y-entities. Subsequently, it generates a chart type among bar, line, or pie, which is suitable for the given text. Combination of these three stages is capable of generating visualization from the given statistical text. We have also constructed a dataset for this problem. Experiments show that Text2Chart performs best with BERT based encodings with LSTM models in the first stage to label x and y entities, Random Forest classifier in the mapping stage and fastText embedding with LSTM in the chart type prediction stage. In our experiments, all the stages show satisfactory results and effectiveness considering the formation of charts from analytical text, achieving a commendable overall performance.

Keywords: Chart generation · Natural Language Processing · Information retrieval · Neural network · Automated visualization

1 Introduction

In recent years, advances in Natural Language Processing (NLP) have made huge progress in extracting information from natural language texts. Among them, a few example tasks are: document summarization [1], title or caption generation from texts, generating textual descriptions of charts [2], named entity recognition [3], etc. There have been several attempts to generate graphs or structural elements from natural language texts or free texts [4–6]. Scientific charts (bar, line, pie, etc.) are visualizations that are often used in communication. However,

J. Gama et al. (Eds.): PAKDD 2022, LNAI 13281, pp. 3–16, 2022.
https://doi.org/10.1007/978-3-031-05936-0_1

automated generation of charts from natural language text has always been a challenging task.

There are very few works in the literature addressing the exact problem of scientific chart generation from natural language text [7,8]. In [7], the authors have presented an infographic generation technique from natural language statements. However, their method is limited to single entity generation only. Text2Chart extends it to multiple entity generation and thus can generate more complex charts. Nevertheless, Generative Pre-trained Transformer 3 (GPT-3)[8] has been a recent popular phenomenon in the field of deep learning. OpenAI has designed this third-generation language model that is trained using neural networks. To the best of our knowledge, there has been an attempt to make a simple chart building tool using GPT-3. As its implementation is not accessible yet, the field of information extraction regarding chart creation can still be considered unexplored to some extent. Moreover, the dataset used in GPT-3 is a very large one, and the training is too expensive.

In this paper, we propose Text2Chart, a multi-staged technique that generates charts from analytical natural language text. Text2Chart works in a combination of three stages. In the first stage, it recognizes x-axis and y-axis entities from the input text. In the second stage, it maps x-axis entities with their corresponding y-axis entities, and in the third stage, it predicts the best-suited chart type for the particular text input. Text2Chart is limited to three types of charts: bar charts, line charts and pie charts. Tasks in each stage are formulated as supervised learning problems. We have created our own dataset which is labeled for all three stages of Text2Chart. We have used a wide range of evaluation metrics for all the three stages and different combinations of word embeddings and classifiers. The experimental results shows that the best results in the first stage are obtained using BERT embedding and Bidirectional LSTM, achieving an $F1$-score of 0.83 for x-entity recognition and 0.97 for y-entity recognition in the test set. In the mapping stage, Random Forest achieves the best results of 0.917 of Area under Receiver Operating Characteristic Curve (auROC) in the test set. In the third stage, the model fastText with LSTM layers performs the best to predict the suitable chart type. Here, Text2Chart achieves the best results of auROC 0.64 for pie charts and auROC 0.91 for line charts. The experimental analysis of each stage and in combination shows the overall effective performance of Text2Chart for generating charts from given natural language charts.

2 Related Work

Recent developments in the field of NLP is advancing information extraction in general. One of the first and foremost steps in NLP is the proper vectorization of the input corpora. One of the breakthroughs in this area is word2vec proposed in [9]. Word2Vec maps words with similar meaning to adjacent points in a vector space. The embedding is learnt using a neural network on a continuous bag of words or skip-gram model. A character-level word embedding is proposed in [10]. Recently, Bidirectional Encoder Representations from Transformers (BERT) is

proposed in [11]. BERT is trained on a large corpora and enables pre-trained models to be applicable to transfer learning to a vast area of research. BERT has been successfully applied to solve problems like Named Entity Recognition (NER) [3], text summarization [1], etc.

Text based information processing has been a long quest in the field [12]. Kobayashi et al. [12] have presented a NLP based modeling for line charts. A Hidden Markov Model based chart (bar, line, etc.) recognition method is proposed in [13]. Graph neural networks have been employed in [4] to generate logical forms with entities from free text using BERT. In a very recent work [5], Obeid et al. have used transformer based models for text generation from charts. For this work, they have also constructed a large dataset extracting charts from Statista. However, their work focuses on chart summarizing and hence called 'Chart-to-Text'. In an earlier work [14], authors have proposed a method for generating ground truth for chart images. Both of the works are limited to bar charts and line charts only. A Generative Adversarial Network, AttnGAN is proposed in [15] that can generate images from text descriptions. Balaji et al. [2] has proposed an automatic chart description generator. CycleGT has been proposed recently that works in both directions: text to graphs and graphs to text [6]. Kim et al. [16] has proposed a pipeline to generate an automatic question answering system based on charts.

Automated visualization has always been a very fascinating area. A survey of Machine Learning based visualization methods has been presented in [17]. Deep Eye is proposed in [18] to identify best visualizations from pie chart, bar chart, line chart and scatter chart for a given data pattern. 'Text-to-Viz' is proposed in [7] that generates excellent infographics from given text. However, their method is limited to a single entity only. GPT-3 [8] has been a recent phenomenon in the field which has been reported to generate charts from natural language texts. However, GPT-3 implementation is not open yet. Moreover, it is trained on an extremely large corpora and an extremely large transformer based model which requires huge resources. In the light of the review of the existing methods, we believe there is a significant research gap to be addressed in this area.

3 Proposed Method

Text2Chart consists of three stages as shown in Fig. 1. It takes a free text as input containing the analytical information. Then it produces x and y axis entities followed by a mapping generation among these elements. In stage 3, the chart type is predicted. A combination of these three are then passed on to the chart generation module. This section presents the detailed procedure of these stages.

3.1 Stage 1: x-Axis and y-Axis Label Entity Recognition

In the first stage, we identify the potential candidate words for both x-axis and y-axis entities of a two dimensional chart. We have formulated the problem as a supervised machine learning task. Here, input to the problem is a paragraph

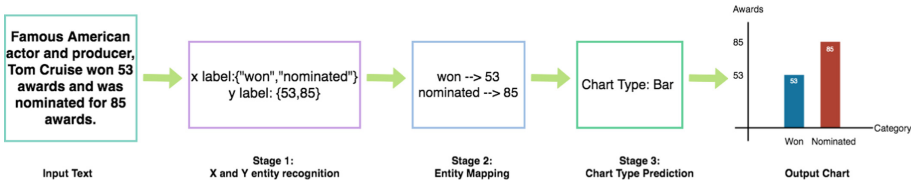

Fig. 1. The overall methodology of Text2Chart.

or natural language text and output is a list of words labeled as x-entity and y-entity.

To identify x-entity and y-entity, we build a neural network with different word embeddings and sequence representations. We have employed and experimented with two different strategies - i) detecting both types of entities at once and ii) using separate models for recognizing x and y entities. Detecting both x and y entities at once shows a drawback as there lies a possibility that a certain type of entity may outperform the loss function of the other types as observed in the experiments (Sect. 4.3).

We have experimented with both of the strategies using word embedding like Word2Vec [9], fastText [10] and the sequence output of the pre-trained model provided by BERT [11]. For each sample text in the dataset, we take the generated embedding and use it as an input to our model. Then we use layers of Bi-directional LSTM networks. On top of that, we use the time-distribution layer and dense layer to classify each word index that falls into a category of a respected entity or not.

3.2 Stage 2: Mapping of x and y Label Entities

After identifying the x and y entities in Stage 1, we map each of the identified x entity with its corresponding y entity. For example, if we have an x entity set for a text as $\{x_1, x_2, \cdots, x_M\}$ and y entity set of that text is $\{y_1, y_2, \cdots, y_N\}$ and their mapping is as follows $\{(x_1, \phi(x_1)), (x_2, \phi(x_2)), \cdots, (x_M, \phi(x_M)\}$. Please note, here x_i, y_j denotes their position in the sequence. Here the mapping function, $\phi(x_i)$ maps an entity x_i to another entity, y_k. However, there is often found that the entity set lengths are not same $M \neq N$ and often the sequential order is not maintained. For two x entities x_i, x_j if they maps to y_k, y_l, then a sequential mapping ϕ guarantees, $i \leq j, k \leq l$ whereas the non-sequential mapping will not guarantee that. However, in our observation, non-sequential mapping is not that frequent. In order to address these issues, we propose that the mapping is dependent on the distances between the corresponding entities. We call it our baseline model for this task. From the training dataset, we learn the probability distribution for positive and negative likelihood for distances between x and y entities which are $P(d(x_i, y_k)|\phi(x_i) = y_k)$ and $P(d(x_i, y_k)|\phi(x_i) \neq y_k)$ respectively. For the missing values in the range, nearest neighbor smoothing is used to estimate the likelihood values and then normalized to convert it to a probability distribution. The baseline model defines the mapping as in the following equation:

$$\phi(x_i) = \underset{k}{\text{argmax}} \frac{P(d(x_i, y_k)|\phi(x_i) = y_k)}{P(d(x_i, y_k)|\phi(x_i) = y_k) + P(d(x_i, y_k)|\phi(x_i) \neq y_k)} \quad (1)$$

For a particular entity x_i and a particular y_k entity, we take the two other entities, one immediately before (x_{i-1}, y_{k-1}) and the next one (x_{i+1}, y_{k+1}) to create the feature vector. For 6 such entity positions, we generate 15 possible pairs and take pairwise distances among them. Note that, for two similar type entities we take unsigned distance and for different entities signed distances are taken to encode their relative positions into the feature vector. With this feature vector, we train two models: SVM and Random Forests, where the latter works slightly better. As this is an argmax based calculation, the probability distribution of the Random Forest classifier was more consistent than that of SVM. The reason for the inconsistency of the distribution with the scores in SVC is that the 'argmax' of the scores may not be the argmax of the probabilities. Therefore we take the auROC as the primary evaluation matrix for this stage. We take the harmonic mean of auROC of both training and validation so that the measure is balanced and they do not outperform each other.

3.3 Stage 3: Chart Type Prediction

Generally, a bar chart is the most commonly accepted chart type for any statistical data. However, for better visualization and understanding, pie charts and line charts are also used. Pie charts are suitable if the entities conform to a collection/composition. Line charts are suitable for the cases where the entities themselves form a continuous domain. For this stage, we have applied fastText word embeddings to build two models with LSTM layers and dense layers. Each model performs binary classification; one is to predict if a pie chart is suited for the text or not, and the other is for the line chart. When neither of these two chart types are fitting, only the bar chart is assigned to the text.

4 Experimental Analysis

Text2Chart is implemented using Tensorflow version 2.3. All the experiments have run using Google Colab and the cloud GPU provided with it. The hardware environment of our work requires a CPU of 2.3 GHz, GPU 12 GB, RAM 12.72 GB and Disk of 107 GB. All the experiments have run at least 5 times with different random seeds and only the average results are reported in this section. Source codes and the dataset of Text2Chart will be made available via a public repository (at the time of publication).

4.1 Dataset Construction

While starting this work, no datasets were available for this particular task of automatic generation of a chart out of a natural language text. Text2Chart requires a specific dataset from which the text samples are suitable for recognizing the chart information. Here chart information refers to the x-axis entities

Table 1. Summary of datasets used in the experiments.

Dataset	Text	x, y Entity prediction				Mapping	Chart type	
	samples	x Tokens	y Tokens	x Labels	y Labels	pairs	Pie	Line
Training	464	3411	3614	1984	1909	1984	73	58
Validation	116	985	1058	548	529	548	20	11
Test	137	988	1075	574	561	574	20	15

and the corresponding y-axis values respectively. The text samples must contain all these entities to construct the particular chart. We have collected text samples from Wikipedia, other statistical websites and crowdsourcing. We have used crowdsourcing to label the data so that the texts are labeled for all three stages. All the labeled data are then cross checked by a team of volunteers and only the consensus labels are taken. In total, 717 text samples are taken in the final dataset with 30,027 words/tokens. The average length of the text samples is 53 words and the maximum length is 303 words in a single text. This final dataset is then split in the train, validation and test sets each containing 464, 116 and 137 samples respectively. A summary of the dataset is shown in Table 1. Please note that in the first stage the token number is higher than labels since a particular x or y entity/label might consist of two words or tokens. All the texts are labeled to be suitable for bar charts and only the statistics for pie and line charts are shown in the table.

4.2 Performance Evaluation

All the methods are trained using the training set and the performance are validated using the validation set. Only after the final model is selected, the model is tested on the test set. For the axis entity recognition task in the first stage, we adopt the $F1$-score and its variant the harmonic mean of f1-scores. We observe the Receiver Operating Characteristic (ROC) curve and the area under curve (auROC) in order to summarize and compare the performances of the classifiers in the second stage of entity mapping. Finally for chart type prediction, we adopt Matthews Correlation Coefficient (MCC) evaluation metric, as MCC being a more reliable statistical rate than F1-score and accuracy in binary classification evaluation for an imbalanced dataset.

4.3 Axis Label Recognition Task

The first stage of our work is x-axis and y-axis label entity recognition. Here we predict whether a given word from the text input can be an x-axis or y-axis entity. We have experimented with our neural architecture model of bidirectional LSTM combining several embeddings, such as fastText, Word2Vec and BERT in order to recognize these entities. For each of the embeddings, we have used two different approaches. In the first approach, x-entity and y entity prediction is considered as separate prediction tasks. Here we have the two models, one for each of the tasks. In the second approach, they are considered together as a combined prediction task.

Experiments with fastText Embedding. For both of the approaches using fastText (individual and combined), we have used a neural architecture with 4 hidden layers and a dense output layer. The first two hidden layers consist of bidirectional LSTM layers of 512 neurons and 128 neurons followed by a time-distributed dense layer of 64 neurons and a dense hidden layer with 1024 neurons. Epoch and batch size are kept fixed at 8 for all the models considered here. Experimental results of fastText experiments are given in the first four rows of Table 2. Note that we have reported precision, recall and $F1$-score for x and y entity predictions. Also the harmonic mean of $F1$-score is reported. Note that, the individual approach achieves $F1$-score for x and y entities of 0.66 and 0.85 respectively in the validation set which is improved in the combined approach being 0.66 and 0.89. It is clear that the prediction or recognition of x axis entities is a much more difficult task compared to y axis entity recognition. Here, we can conclude that both models perform almost similarly which is also reflected in the harmonic mean of $F1$-score respectively 0.74 and 0.76.

Table 2. Experimental results for the axis label prediction task in the frist stage of Text2Chart.

Model	Dataset	Precision (x)	Recall (x)	Precision (y)	Recall (y)	$F1$-score (x)	$F1$-score (y)	Harmonic $F1$-score
fastText	Training	0.81	0.80	0.93	0.88	0.80	0.90	0.84
Individual	Validation	0.68	0.64	0.89	0.81	0.66	0.85	0.74
fastText	Training	0.81	0.73	0.89	0.97	0.77	0.93	0.84
Combined	Validation	0.73	0.60	0.86	0.93	0.66	0.89	0.76
word2Vec	Training	0.90	0.88	1.00	1.00	0.89	1.00	0.94
Individual	Validation	0.72	0.62	0.79	0.77	0.67	0.78	0.72
word2Vec	Training	0.99	0.99	1.00	1.00	0.99	1.00	0.99
Combined	Validation	0.72	0.64	0.83	0.74	0.68	0.78	0.73
BERT	Training	0.99	0.99	.99	0.99	0.99	0.99	0.99
Individual	Validation	**0.89**	**0.86**	0.95	**0.98**	**0.87**	**0.97**	**0.92**
BERT	Training	0.99	1.00	0.99	1.00	0.99	0.99	0.99
Combined	Validation	0.86	0.78	**0.96**	0.97	0.82	**0.97**	0.89
Best	**Test**	0.85	0.82	0.96	0.98	0.84	0.97	0.89

Experiments with word2vec Embedding. The word2vec embedding represents the word tokens in the corpus by representing the words with common context in a close proximity in the vector space as well. Similar to the experiments of fastText we have two approaches employed here: individual and combined. For word2vec embedding, the network structure is kept the same as in the fastText experiments. However, for training we have used 16 epochs and a batch size of 8. The experimental results are shown in the second four rows of Table 2. From Table 2, we can see that this combined approach is giving $F1$-score of the x and y entity recognition task as 0.68 and 0.78 respectively which is almost similar to the performance of the individual approach (0.67 and 0.78 respectively). The performance only differs in the x entity recognition task which is also observed in the harmonic mean of $F1$-score. Note that the overall performance of word2vec

embedding is significantly worse compared to fastText embedding. Also note that the higher level of overfitting of the word2vec model has reflected in the high values of precision, recall and $F1$ score in all the tasks in the training dataset which is not repeated in validation.

Experiments with BERT Embedding. We have also experimented with BERT embeddings on the same architecture proposed in Sect. 3. However, in these experiments the network structure is different with the same number of layers. Here too we have used two approaches: individual and combined. In the individual approach, the first two hidden layers of the neural architecture are bidirectional LSTM with 1024 neurons in each followed by a time-distributed dense layer with 1024 neurons and a dense layer with 256 neurons. In the case of x entity recognition, we have used a batch size of 2 and 80 epochs for training. In the case of y entity recognition, the batch size was 8. In the combined approach, the architecture structure has differed only in the last hidden dense layer. Here the number of neurons is 1024. We have used an online training for this combined approach. The experimental results with BERT embedding is reported in the third four rows of Table 2. From the results shown there, we can notice that for BERT embedding, the performances in the individual approach outperform the combined approach in x entity prediction performance. The results in y entity recognition is almost similar for both of the approaches. Thus the both harmonic mean and $F1$-score of x entity recognition are superior in combined approach which are 0.87 and 0.92 respectively compared to those of 0.82 and 0.89 in the individual approach.

To summarize, we can note that the results in BERT embedding are superior to two other embeddings. The best achieved values are shown in boldfaced fonts in the Table. Thus, we take the BERT embedding individual x and y entity prediction approach with bidirectional LSTM as the best performing model among those used in the experiments. With the best model, we have also tested its performance on the test dataset. The results are shown in the last row of Table 2. Here, it is interesting to note that the learned model is not overfitting and the performances in the validation set and test set are not much different.

4.4 Mapping Task

After recognizing the x and y entities with high precision and recall in stage 1, the second stage sets the target to map them in an ordered way. We have first used a transfer model from the best performing model in the first stage to see if that helps. However, the very low $F1$-score of 0.41 and auROC of 0.64 have discouraged us from proceeding in this way. It is evident that the same architecture is not suitable for the different stages due to differences in the type of the task. Note that this task is highly imbalanced as the number of positive mappings are very small compared to negative mappings. Thus the model often gets biased towards the negative model and might show poor performance in the positive prediction.

Table 3. Experimental results for the mapping task in the second stage.

Model	Dataset	Class	Precision	Recall	$F1$-score	Harmonic $F1$-score	auROC
Baseline	Training	0 (−ve)	0.94	0.94	0.94	0.84	0.908
		1 (+ve)	0.76	0.76	0.76		
	Validation	0 (−ve)	0.95	0.95	0.95	0.82	0.914
		1 (+ve)	0.73	0.73	0.73		
SVM	Training	0 (−ve)	0.93	0.93	0.93	0.81	0.897
		1 (+ve)	0.72	0.72	0.72		
	Validation	0 (−ve)	**0.96**	**0.96**	**0.96**	**0.86**	0.924
		1 (+ve)	**0.78**	**0.78**	**0.78**		
Random Forest	Training	0 (−ve)	0.95	0.95	0.95	0.85	0.913
		1 (+ve)	0.77	0.77	0.77		
	Validation	0 (−ve)	**0.96**	**0.96**	**0.96**	0.84	**0.930**
		1 (+ve)	0.77	0.77	0.77		
Best	**Test**	0 (−ve)	0.94	0.94	0.94	0.85	0.917
		1 (+ve)	0.77	0.78	0.77		

Our baseline model is a simple argmax calculation of the likelihood based on Eq. (1). The results of the baseline model are presented in the first four rows of Table 3. In this table, we have reported precision, recall and $F1$-score for both of the classes and also the auROC. Note that the results of the baseline model is encouraging with a high auROC of 0.908. However, note that the positive class performance is poor compared to the negative class which leaves room for improvement.

Next we have experimented with the supervised learning approach described in Sect. 3 using Support Vector Machine (SVM) and Random Forest classifiers. In Table 3, we notice that the performance in both of the classes are improved using this approach in both of the classes compared to the baseline model. We note that the performance in the negative class is the same. However, the $F1$-score of the Random Forest classifier is slightly lower in the positive case which is not that significant (0.77 vs 0.78). The fact is evident in auROC. There we see significant improvement achieved by the Random Forest classifier compared to SVM. The best values are shown in boldface font in the table. Thus we conclude that Random Forest is the best performing model for stage 2.

Finally, we have tested the performance of the best performing Random Forest model on the test set and the results are shown in the last row of Table 3. We see that the performances in the test set are stable and similar to the validation set.

4.5 Chart Type Prediction Task

At the third stage, the task is to predict the suitable chart type from the given text. Note that for all the texts in the dataset, the bar chart is common and thus we exclude it from classification models. We train two separate models: one for the pie chart and another for the line chart. This model uses fastText embedding

Table 4. Experimental results for chart type prediction task.

Problem	Dataset	Specificity	Sensitivity	MCC	auROC
Pie chart	Training set	0.742	0.944	0.51	0.86
	Validation set	0.6945	0.714	0.32	0.66
	Test set	0.573	0.75	0.22	0.64
Line chart	Training set	0.9634	0.963	0.96	0.96
	Validation set	0.990	0.933	0.92	0.98
	Test set	0.893	0.733	0.51	0.91

with bidirectional LSTM layers. The network architecture and structure is kept the same for both of the classifiers. The neural network has three hidden layers. The first two layers are the LSTM layers with 128 neurons each followed by a dense layer of 512 neurons. The output layer is a simple sigmoid layer. We have used the RMSprop algorithm to train the models.

For pie chart recognition, we set the batch size to 128 and the learning rate to 4e-4. As we have a highly imbalanced dataset, we achieve good enough results in terms of MCC, scoring 0.22 in the test set as shown in Table 4. The obtained auROC for pie charts is 0.64 in the test set. We have achieved a better result in terms of recall or sensitivity of 0.94 in the training set, 0.71 in the validation set and 0.75 in the test set. For line charts, we set the batch size to 256 and the learning rate remains as default to 1e-3. In Table 4, we find outstanding results in terms of auROC score of 0.96 in the training set, 0.98 in the validation set and over 0.91 in the test set. Our obtained MCC in the train, validation and test sets is 0.96, 0.92 and 0.51 which is a better score than the prediction of pie charts.

4.6 Overall Performance

In order to discuss the overall performance of our work, we have created a pipeline same as shown in Fig. 1. Our pipeline merges all the stages of our work and outputs the results we have already discussed and shown in this section. After obtaining the final results, we have checked for all possible errors that occur after completion of each stage. After completing stage 1, if both of the entity sets have a similar number of entities ($N = M$) then we consider 1-to-1 sequential mapping. The cumulative frequency of error count for each of the stages is shown in Fig. 2. This plot shows how each stage cumulatively produces error in the pipeline. However, we notice that although we have a good number of samples without error, there is a room to improve and as shown in the figure, the most error-prone task is task 3 due to the poor performance in pie chart type prediction. We also show one partially correct and one fully correct chart example generated by Text2Chart in Table 5.

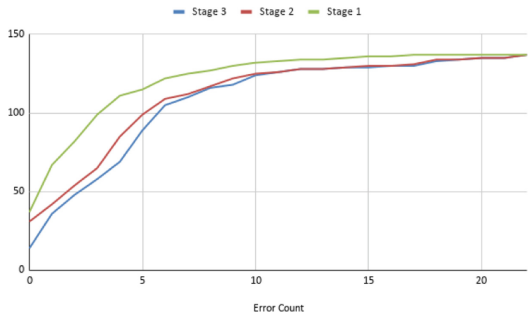

Fig. 2. Cumulative frequency of error of three states put in a pipeline on the test set.

Table 5. Sample input and outputs of Text2Chart.

Input	Sample text	Tzuyu is a gaming expert . She surveyed 200 individuals to judge the popularity of the video games among her all time favorites . After her survey she concluded that 25 people voted for World of Warcraft , 46 voted for Black Ops , 12 voted for Overwatch , 25 for Modern Warfare , 30 for PUBG , 50 for Sims and 40 for Assassin ' s Creed
Output	x Entities	['World of Warcraft', 'Black Ops', 'Overwatch', 'Modern Warfare', 'PUBG', 'Sims', 'Assassin', 's Creed']
	y Entities	['25', '12', '25', '30', '50', '50', '40', '40']
	Chart type	['bar']
Input	Sample text	Mr . Jamal worked in the Meteorological Department for 8 years . He noticed a strange thing in recent times . On certain days of the month , the weather varied strongly . He wrote down the information to make a pattern of the event . The information of the paper is as follows : on the 3rd day of the month the temperature is 36 °C , 7th day is 45 °C , 9th day is 18 °C , 11th day is 21 °C , 17th day is 9 °C , 19th day is 45 °C , 21st day is 36 °C , 27th day is 21 °C and 29th day is 45 °C . He finds a weird pattern in these dates and makes a report and sends it to his senior officer
Output	x Entities	['3rd day', '7th day', '9th day', '11th day', '17th day', '19th day', '21st day', '27th day', '29th day']
	y Entities	['36', '45', '18', '21', '9', '45', '36', '21', '45']
	Chart type	['bar', 'Line']

5 Conclusion

In this paper we have presented Text2Chart, an automatic multi-staged technique that is able to generate charts from human written analytical text. Our technique has been tested on a dataset curated for this task. Despite having a short corpora, Text2Chart provides satisfactory results in every stage regarding automatic chart generation. One of the limitations of our work is the size of the dataset. With a larger dataset, we believe the methodology presented in this paper will provide further improved results. Text2Chart is currently limited to the prediction of only three basic chart types: bar charts, pie charts and

line charts. It is possible to extend it for further types. Recently a dataset for chart-to-text has been proposed in [5]. It is possible to use that dataset for the reverse problem also. We believe it is possible to tune and experiment with more types of suitable neural architecture further for all the stages to improve overall accuracy.

A Network Architectures

See Figs. 3 and 4.

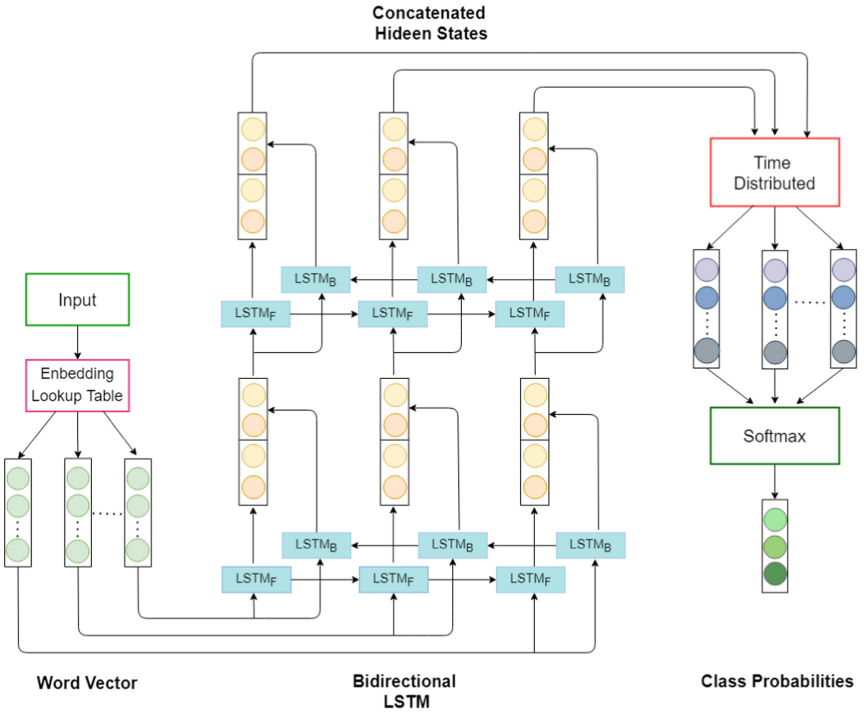

Fig. 3. Proposed neural architecture for recognition of x-axis and y-axis entities.

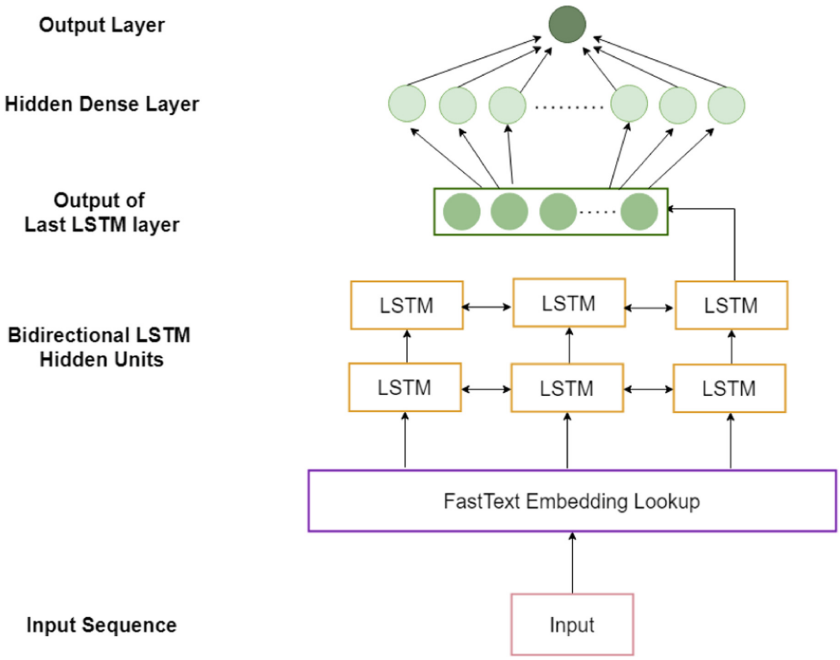

Fig. 4. Proposed neural network architecture for chart type prediction.

References

1. Liu, Y., Lapata, M.: Text summarization with pretrained encoders. arXiv preprint arXiv:1908.08345 (2019)
2. Balaji, A., Ramanathan, T., Sonathi, V.: Chart-text: a fully automated chart image descriptor. arXiv preprint arXiv:1812.10636 (2018)
3. Sang, E.F., De Meulder, F.: Introduction to the conll-2003 shared task: language-independent named entity recognition. arXiv preprint arXiv:cs/0306050 (2003)
4. Shaw, P., Massey, P., Chen, A., Piccinno, F., Altun, Y.: Generating logical forms from graph representations of text and entities. arXiv preprint arXiv:1905.08407 (2019)
5. Obeid, J., Hoque, E.: Chart-to-text: generating natural language descriptions for charts by adapting the transformer model. arXiv preprint arXiv:2010.09142 (2020)
6. Guo, Q., Jin, Z., Qiu, X., Zhang, W., Wipf, D., Zhang, Z.: Cyclegt: unsupervised graph-to-text and text-to-graph generation via cycle training. arXiv preprint arXiv:2006.04702 (2020)
7. Cui, W., et al.: Text-to-viz: automatic generation of infographics from proportion-related natural language statements. IEEE Trans. Visualiz. Comput. Graph. **26**(1), 906–916 (2019)
8. Brown, T.B., et al.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020)
9. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)

10. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Trans. Assoc. Comput. Linguist. **5**, 135–146 (2017)
11. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805. (2018)
12. Kobayashi, I.: Toward text based information processing: with an example of natural language modeling of a line chart. In: IEEE SMC 1999 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 99CH37028), vol. 5, pp. 202–207. IEEE (1999)
13. Zhou, Y., Tan, C.L.: Learning-based scientific chart recognition. In: 4th IAPR International Workshop on Graphics Recognition, GREC, pp. 482–492. Citeseer (2001)
14. Huang, W., Tan, C.L., Zhao, J.: Generating ground truthed dataset of chart images: automatic or semi-automatic? In: Liu, W., Lladós, J., Ogier, J.-M. (eds.) GREC 2007. LNCS, vol. 5046, pp. 266–277. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88188-9_25
15. Xu, T., et al.: Attngan: fine-grained text to image generation with attentional generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1316–1324 (2018)
16. Kim, D.H., Hoque, E., Agrawala, M.: Answering questions about charts and generating visual explanations. In: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, pp. 1–13 (2020)
17. Wang, Q., Chen, Z., Wang, Y., Qu, H.: Applying machine learning advances to data visualization: a survey on ml4vis. arXiv preprint arXiv:2012.00467 (2020)
18. Luo, Y., Qin, X., Tang, N., Li, G.: Deepeye: towards automatic data visualization. In: 2018 IEEE 34th International Conference on Data Engineering (ICDE), pp. 101–112. IEEE (2018)