Sentence Auto-complete Inspired By G-mail Smart Compose

Lakehead University
Vatsal Patel-0883974
Chirag Patel-0883437
Dhruvi Desai-0884718

***Abstract*—This paper discusses topic of sentence or word auto-complete, which is in world of Natural Language Processing also known as Language Modelling. Idea of this project is inspired by Smart Compose provided Google G-mail and mobile phone keyboards. Language Modelling uses NLP and machine learning to interactively offer sentence completion suggestions as user type anything, allowing user to draft sentence or text faster and efficiently. In our model suggestion are usually made based on the training corpus text. This application can potentially reduce grammar errors as well help in sentence formation. Paper includes Importance of this application and NLP, Dataset, Architecture, Performance, Literature Review, pros and cons of using this application and GUI component.**

***Index Terms*—Sentence Auto-complete, Language Modelling, Smart Composing, NLP, Google, Gmail, n-garm, char-gram, RNN, LSTM.**

I. Introduction

Natural Language Processing is the subfield of Artificial Intelligence. NLP mainly focuses on the comprehension by computers of the meaning and structure of human language, which enables users to interact with the computer using natural sentences. In other words, NLU is Artificial Intelligence that uses computer Algorithms or software to interpret the text and any type of unstructured data provided by humans. NLP can digest a text, translate it into computer language, and produce an output in a language that humans can understand.

There are much application of NLP and since most of them are somehow related to human language and text, and voice. It is one of the most widespread used application is sentence auto-complete or suggestion. This application uses concept of Language Modeling. A trained language model learns the likelihood of occurrence of a word based on the previous sequence of words used in the text. So its a sequence to sequence modeling. Auto-complete is used at many places like Google search engine, G-mail, Mobile Key-pad and many more. Smart compose by G-mail makes prediction based on current email context, mail they are replaying and subject body. Google also uses the large dataset of emails to train the model. And users will be suggested subsequent word while they type.

We have used NLP concepts such as n-gram, char-gram, data pre-processing, Language Modelling and probabilistic distribution, Tokenization, word embedding, Recurrent Neural Network, RNN with Long Short Term Memory (LSTM).

II. Importance of the Application

We engage in a lot of text based communication on a daily basis. Most of the web and mobile apps today come with many great features and applications to improve our productivity. The NLP is becoming an increasingly growing topic of conversation both in workplace and outside of it. Some of the most used application of NLP are Word Embedding, automatic summarizing, sentiment analysis, image captioning and many more.

Sentence Auto-complete might not seem so significant as in application but it helps us a lot in day to day life. Now a days in age of texting and emails, use of auto-complete comes in handy in many ways. Google has been providing this feature since long time in their search engine but recently they introduced G-mail's Smart compose which is one step further than smart reply by G-mail and its works in real time. It saves lots of time by providing suggestion based on what user is typing. That is way user don't have to type entire sentence and just press TAB to apply the suggestion. It also reduces the grammar and spelling mistakes made by humans. This service is not limited to English but also available in Spanish, French, Italian and Portuguese. Google is also making Smart Compose better by adapting to an individual users style of mailing. If a manager of a company prefers a casual writing style when addressing him department, for example, Gmail will observe previous mails and may suggest Hey team rather than a more formal greeting. Apart from that, it not only suggest subsequent words, but Smart Compose also now recommends subject line for an email, based on the message content that the user writes. Lots of people who uses G-mail might not have English as their first language, in those cases smart compose is proven to be very useful, causing them minimal stress composing the long emails. Apart from that Whatsapp keyboard also predicts the next possible word and presents you with the top 3 possibilities. While it is model based, it only predicts the next word (unigram) or at most the next word pair (a bigram), but nothing further. The three main functionality of auto-complete are :

- **Suggestions of possible next word or sequence of words**
- **Reduce Grammar Errors**
- **Improved sentence formation**
- **Reduced text compose time**

III. Literature Review

As we consider the literature review on G-Mail Smart Compose, we know it is not a standard machine learning program with existing applied algorithms. As we know that G-Mail Smart compose uses sequence-to-sequence leaning framework and that uses long short-term memory networks know as LSTMs to predict the next word [6] in the sentence. We will review the two main core components that help in being this model, which is predicting responses and identifying a target response in space.

Predicting responses would be analyzing natural language in open domains like social media to predict if a response was given or not, predicting the next work or threads.

Anjuli, Karol, Sujith and Tobias has used full response prediction on Twitter post to perform complete analysis to approach the problem from the view of machine translation which was first applied by [1],[5]. Using this, they would translate the Twitter post which is translated it into response using phrase-based statistical machine learning (SMT). They had a similar approach, where they proposed a neural network machine translation model called sequence-to-sequence learning. This uses long short term memory networks (LSTMs) [2] to predict the next sequence of text.

Sordoni, Galley, Auli and the team [3] have also applied Recurrent neural networks RNNs or LSTMs to predict full response where they reply completely on generating number of best lists for responses.

Where as O. Vinyals and Q. V. Le [4] has worked on a research proposed to work on mapping sequences to sequences. The research framework that they have referred to has been used for neural machine translation on English-French and English-German translation which achieves great improvements.

IV. Dataset

To build a effective Language Model we need to train the model on very large relevant English text corpus. Since we are focusing mainly on predicting next word or sequence of words in an e-mail, the dataset must contain the frequently used terms in an emails. After doing intensive research we came to know about "Enron Email Dataset". It consist of about 500,000 emails exchanged by 150 employees of the company. It is not just the e-mail body but the subject line, data and time of the e-mails sent. Since it is really huge dataset and requires lot of pre-processing and computing power to train the model, we came up with smaller but similar to this dataset consisting about 1,000 commonly used phrases in most of the e-mails nowadays.

```
['\ufeffHey, place the toolbox in the truck!\n',
 '\n',
 'Hey, what are you doing here?\n',
 '\n',
 'Hey, give me a break.\n',
 '\n',
 'Hey, John.\n',
 '\n',
 "Hey, what are you doing? Don't touch it, or you'll break the balance!\n",
 '\n']
```

Fig. 1: Sample of raw data

The above image is sample of raw data. This dataset consist of commonly used phrases like "Hey, what are you doing here?" and "Hey, John".

A. Pre-processing of the data

As part of pre-processing we are removing all the punctuation like "*+-/¡=¿@[]‘— " from the text and converting the corpus into lower case. After that we applied char-gram on the text sentence basically dividing the sentence into sequence of characters as input and sequence of characters as output increasing number of total samples. We have divided dataset into 80:20 ratio where 80% is training with 30010 sample and 20% is validation with 7503 samples.

	input	output
60	<start> hey, what are you doing he <end>	<start> re? <end>
61	<start> hey, what are you doing her <end>	<start> e? <end>
62	<start> hey, what are you doing here <end>	<start> ? <end>
63	<start> hey, what are you doing here? <end>	<start> <end>
64	<start> he <end>	<start> y, give me a break. <end>

Fig. 2: Sample Text after pre-processing and applying char-gram

Above images shows the final data after pre-processing that we are using. Since we are using char-gram one sentence is divided into multiple sequence of input and output, for example in index 60 "hey, what are you doing he" is the input sequence and "re?" is output sequence.

V. Architecture

Auto-Complete is a feature that replicates G-mail smart compose which uses machine learning to interact and offer suggestions for sentence completion as we type, allowing us to draft email faster. Also, it offers new ways to help compose emails, whether responding to an incoming email or drafting a new message from scratch. When user inputs come are short, word level predictive texts are a great fit. And in emails, the user has to type in a lot of text at once across multiple emails. Also, in emails, the user ends up repeating a lot of sentences, from greetings to basic pleasantries to closure notes. Here comes in smart compose, which is different a true productivity lifter. It uses the current text as well as it also uses the subject and the previous message to complete the sentence for you. On observing closely, we will notice that different completions are delivered at different levels, some completions were delivered after two or three words while some after a few characters.

Models used for language generation models to predict the next word conditioned on the prefix word sequence are:

- **Language models:**The main aim of a language model is to compute a probability of a sentence or a sequence of words Language Model (LM) a grammar of a language as it gives the probability of word that will follow.
- **Char-Grams:** It is used for different tasks like natural language processing or text mining, etc. It is a type on N-Gram which deals with one character at a time. There is a set of co-occurring characters within a given

window and while computing the char-grams are moved one character forward(for other scenarios you can move n word forward, n-gram, where N- represents the number of words to move foward). For example, for the sentence ”Full moon day.” Char gram would take each character, so the char-gram would be: ”F”, ”u”, ”l”, ”l”, ”m”, ”o”, ”o”, ”n”, ”d”, ”a”, ”y”, While developing a language model, char-grams are used to develop not just character sequences but also used as uni-gram(N=1) models, bi-gram(N=2) and tri-gram(N=3) models and when N is greater than 3 which referred to as four grams or five grams and so on.

- **RNN:** Recurrent Neural Networks are a very important variant of neural networks which is heavily used in Natural Language Processing. It gives the flexibility for the network to work with varying lengths of sentences which cannot be obtained in a standard neural network because of its fixed structure. It also provides the additional advantage of shared features learned across different positions of text or corpus which can not be obtained in a standard neural network. An RNN treats each word of a sentence as a separate input occurring at time t and use the activation value at t-1 also, as an input in addition to the input at a time t. An RNN recursively applies computation to every instance of an input sequence conditioned on the previously computed results. These sequences are typical representation of a fixed-size vector of tokens which are fed sequence by sequence or one by one to a recurrent unit. The figure below illustrates a simple RNN framework below. The core strength of a Recurrent Neural Network has the capacity to remember and store the results of previous computations and use the information in the current computation. This makes Recurrent Neural Network models suitable to context dependencies in inputs of arbitrary length to create a proper composition of the input. RNNs been used to study various NLP tasks such as machine translation, image captioning, and language modeling, among others. There are 3 RNN architecture is One to Many RNN, Many to One RNN and Many to Many. For suggesting next word to the user for emails, Many to Many architectures is used. This architecture refers to many inputs are read to produce many outputs, where the length of inputs are not equal to the length of outputs.
- **Seq2Seq Model:** This model has evolved the process of translation by adapting the use of deep learning. It not only takes the current input into account while translating but also its neighborhood. It takes an input as a collection sequence of words(sentence or sentences) and generates an output sequence of words/sentences. It does so by use of the RNN. It is also known as encoder-decoder RNN architecture.

  Working: Language Model generates a random sequence of arbitrary length with or without prefix or prompt, which can be a word or another sequence of arbitrary length. Conditioned generation is a generation of an arbitrary length sequence with given input as arbitrary length sequence and creates output which is based on the input. This term is called Seq2Seq modeling. Machine Translation is the official example of conditioned generation because it is easy to drive the point to the readers. As shown below is a layout of a classical Seq2Seq model where x1,x2xn being the input sequence and y1,y2..ym being the output sequence, where ¡start¿ and ¡end¿ are teacher forcing delimiters.

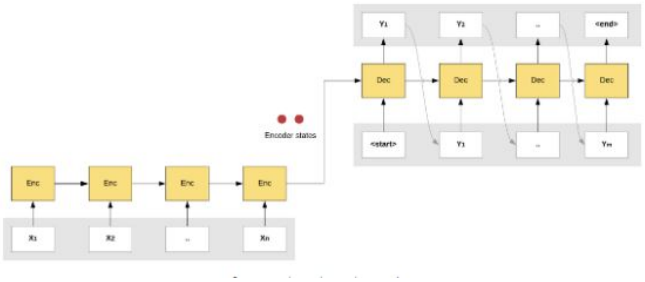


Fig. 3: Layout of an RNN based Conditioned Generator

From this, we learn that Gmail smart compose a conditioned generation with the input sequence as current email text + subject line + previous email text(for replying) and the output sequence as predicted sentence completions. The below diagram shows one possible smart compose model architecture

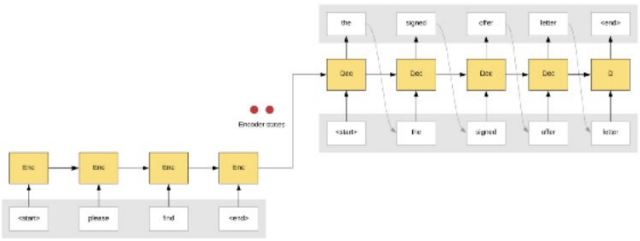


Fig. 4: Conditioned generation in Smart Compose

Markov models have been used by people for learning a Language model in the past. But it had many disadvantages, instead of Markov models RNNs has emerged as a goto architecture to do sequence modeling and also to do Language model. Especially the fact that RNNs can oppose Markov limitation and factor in long-range dependencies.In the below figure shows a couple of different Seq2Seq modeling flavors RNNs offer. On comparing the conditioned generator layout above with these images, it is clear that the many to many (n:m) are favored viz. where input sequence length is not equal to the output sequence length which is exactly same as a conditioned generator.

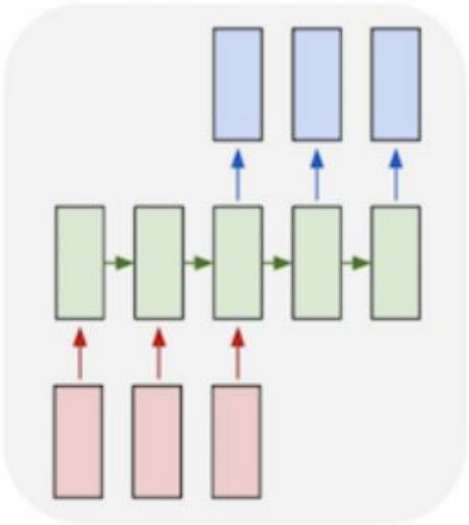


Fig. 5: many to many (n:m)

The conditioned generator is also named as Encoder-Decoder(seq2seq model) architecture, where the term conditioned generator explains what the architecture does the term Encoder-Decoder simply names the components in the architecture.

To obtain improved results, a combination of the BoW model and RNN-LM, which is faster than Encoder-Decoder models. In this hybrid model, the subject and previous email are encoded by averaging the word embeddings in each field. Then these averaged embeddings are combined and fed to target sequence RNN-LM at every step of decoding as shown in below figure.

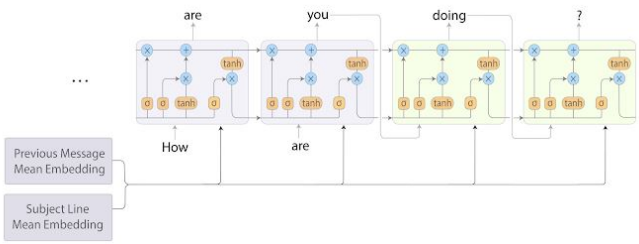


Fig. 6: Smart Compose RNN-LM model architecture.

Our Proposed Model:

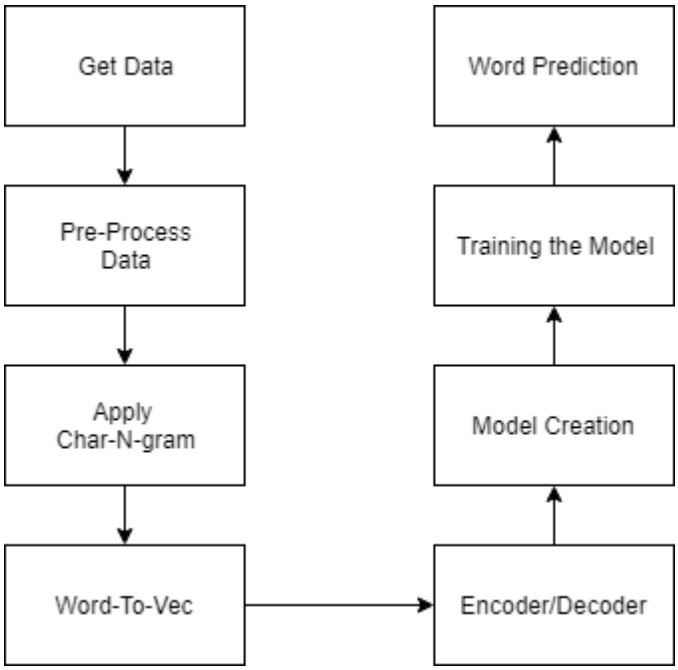


Fig. 7: Proposed Model

The block diagram above shows our proposed model for compiling auto complete. We first get the data and pass it through pre-processing which removes all the unwanted punctuation and null values as explained in the pre-processing step of the data set explained above. Than we apply the Char-Gram algorithm to convert the data set to a stream of character that can be used to training the model. This stream of characters are then convert to integers using word2vec algorithm where each character is assigned with an array of integers in 1's and 0's.

The reason for converting characters to integers is because it makes the machine to easily understand and train the model instead of training the characters. This series of integers are then passed to the Encoder/Decoder. The Encoder takes the input and train on it to pass the last state of the recurrent layers and initial inputs to the recurrent layer of the decoder. The decoder than takes the input from the encoders last recurrent layer and gives out the input in the format that we want.

Once we have received the output from the decoder we train our model on the sequence of inputs for 50 epochs and compile with the loss function of Sparse categorical cross-entropy. The reason we use sparse categorical crossentropy rather than categorical crossentropy is because sparse categorical crossentropy works better for integers and categorical crossentropy works with one-hot encoded data. We achieve a accuracy of 99 percent using sparse categorical crossentropy and minimum loss. After training the model, pass inputs to the model to test the prediction for next words. It would suggest the most possible next word or sequence of words.

VI. Performance Details

- **Suggestions:** After finding these results thoroughly we came to know that this application is more effective for

some of the common phrases like ”I hope” following ”You are doing well” and ”have a” followed by ”Nice day”. But in other and most of the time it doesn't suggest the auto-complete sentence. It require more training and the more user uses the features the more accurate suggestion it gives.

- **Training:** One of the most important parameter of building any model is the training part. Performance the prediction accuracy depends on number of epochs. When we trained the model on 50 epochs instead of 10 epoch we got better prediction, and training loss reduced as well.
- **Scale:** G-mail is used by more than 1.4 billion diverse users. Similarly our model has too high modeling capacity and computation power, but due to the lack of hardware resources we cannot compare it with G-Mail smart compose.We have trained our model only on the content of the E-Mail body but to improve out model it can be trained on the subject line, reply and content of the reply to make better suggestions.
- **Fairness and Privacy:** While training the model, one needs to make sure that the models never expose the users private information.

VII. Pros and Cons

In today's world, Google is that friend that knows you so well that it can finish your sentences! Auto-complete feature has freed people up to less time on email. It is one of the other automation services provided by our project, which people might wonder that their jobs are reduced to hitting tab. It is an algorithm which suggests something while to type an email to say to the prospective customer. This algorithm suggests our next line of text, and we hit Tab. Even though with suggestions provided by the algorithm doesnt preclude anyone from taking those suggestions and doing something even more creative with them. For example, employees at companies see suggestions from the same algorithms, the one who figures out innovative ways in using this suggestion (viz know when to hit Tab and knows when to do tasks completely different) to stand out will be a winner among st. The best workers will know how to automate mundane tasks and invent something fresh.

Sometimes by not behaving the way users expect, auto-complete could even take your customers on detours that affect the search experience, which results in less use of auto-complete search.

Since auto-complete is seen as a method to speed up the search while reducing unintended matches by only displaying validated results. Auto-complete learns from the users' previous inputs and predicts the word user wants to enter only after a few characters have been typed into a text field.

It is considered that people read faster than they type, so by displaying valid results as people key the word in the input field, which means there are also time savings possible and also the system prompts the user to pick a validated result.

After experimenting with this application we came to know that smart compose does not provide suggestion in most of the cases and it is very limited to certain phrases. It tends to improve over time as feature is used more and more, this can be considered as major con of smart compose.

However, despite the Cons of auto-complete search in our project, the major Pros are:-

- People are used to this style of data input
- It can help eliminate email address entry errors through misspelling
- It can radically reduce the time being taken to type and reply to an email.

VIII. Experimental Results

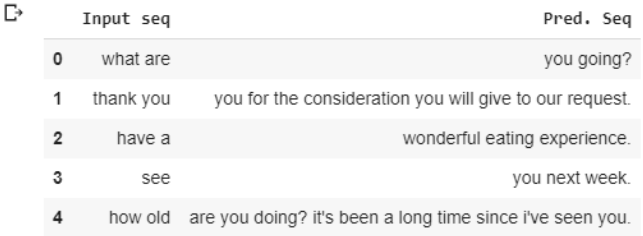
We have trained our model on different epochs ranging from 10 to 50. This sections shows various input sequence and output that we got. We came to know that the prediction might not always be right and sometime model yielded predictions that didn't make any sense. Also training the model on more epochs gave us better prediction comparatively.

TABLE I: OUTPUT.

Input Sequence	Output Sequence
My wife	get away with this
Haven't seen	you are greeting the person in the same manner.
how are	you say hullo or hello, you are getting a person in the same manner.
stopped	to say hello to him for me.
thank you for	see you next time.
thank you one	that we would like to discuss urgently regarding club activities, so please come to the staff room.
see you next	are you for the high volume of calls.
what	to see you next time.

This example illustrates some sequence of input and their predicted sequence of output. As per example 7 user typed ”See you next” and model predicted ”Year”, and in example 8 user type ”good morn” and model predicted ”ing, how are you sitting in the house? there are two cats”. This example shows how char-gram works, as user typed ”morn” model knows next few characters will be ”ing” because it was trained such way.

article



	Input seq	Pred. Seq
0	what are	you going?
1	thank you	you for the consideration you will give to our request.
2	have a	wonderful eating experience.
3	see	you next week.
4	how old	are you doing? it's been a long time since i've seen you.

Fig. 8: Example-2

The accuracy of the model highly depends on the training data context and size. If user entered such words are not present in the training data or rarely used then it might not predict the next word properly. In example 4 we can see that user entered ”how old” and it predicted ”are you doing?” which doesn't make sense. This can be overcome by training on more epoch and with help of better and larger data set.

IX. Conclusion

The conclusion of this experimental paper is that Auto-complete uses various models to predict the next sequence of output. We also learned about different models when combined with other models the accuracy can increase. The current working of this experiment provides good accuracy as it being trained on a small database. It learns the patterns and model, and then is suggests for auto-complete options which saves times and corrects the grammar. It can be used or real-time application with more work and research.

REFERENCES

[1] Data-Driven Response Generation in Social Media, Alan Ritter Computer Sci Eng. University of Washington Seattle, WA 98195, Colin Cherry National Research Council Canada Ottawa, Ontario, K1A 0R6, William B. Dolan Microsoft Research Redmond, WA 98052

[2] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural Computation, 9(8):17351780, 1997

[3] A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J.-Y. Nie, J. Gao, and B. Dolan. A neural network approach to context-sensitive generation of conversation responses. In In Proceedings of NAACL-HLT, 2015.

[4] O. Vinyals and Q. V. Le. A neural conversation model. In ICML Deep Learning Workshop, 2015.

[5] S. M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recogniser. IEEE Transactions on Acoustics, Speech, and Signal Processing, 35:400401, 1987.

[6] B. Pang and S. Ravi. Revisiting the predictability of language: Response completion in social media. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 14891499, Jeju Island, Korea, July 2012. Association for Computational Linguistics

[7] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[8] Jason Brownlee, How to Develop Word-Based Neural Language Models in Python with Keras

[9] Sebastian Ruder, A Review of the Neural History of Natural Language Processing