A Survey of Predictive Text

Joshua Leppo[1]

[1]Computer Science, Penn State Harrisburg, Penn State University,
777 West Harrisburg Pike, Middletown, 17057, PA, USA.

Abstract

TODO

Keywords: TODO

1 Introduction

Natural Language Processing (NLP) is a field within machine learning that deals with human language. While there are many subfields within NLP, in this literature survey, we will review text prediction within the text processing field. A computer's ability to recognize text patterns and make meaningful and useful predictions about which characters or words might come next has a variety of practical applications.

Text predictions can be used to speed up the time it takes to respond to emails or write letters. It can lead to decreased spelling and grammatical errors. It can assist with the translation from one language to another. These abilities can benefit business by increasing productivity and efficiency. Individuals with cognitive or motor impairments will benefit from needing less keystrokes to complete their writing as well as minimizing errors. There are a number of practical applications that have an even greater number of benefits. Research on NLP text prediction methods is required to ensure efficient, useful text predictions.

In this literature survey, we will explore twelve papers ranging from 2005 up to and including 2023. There are three papers from 2020 to current, three papers from 2015 to 2019, three papers from 2010 to 2014 and three papers written before 2010. This should provide a range of views and methods on the topic of text prediction, covering almost twenty years of research.

2 Papers Before 2010

In this section, we review [1], which predicts sentence completion using an n-gram model with beam search. [2] focuses on single word predictions and a user interface for efficient use of the model. In [3], word disambiguation is approached using a classification system rather than a straight forward prediction model.

2.1 Predicting Sentences using N-Gram Language Models

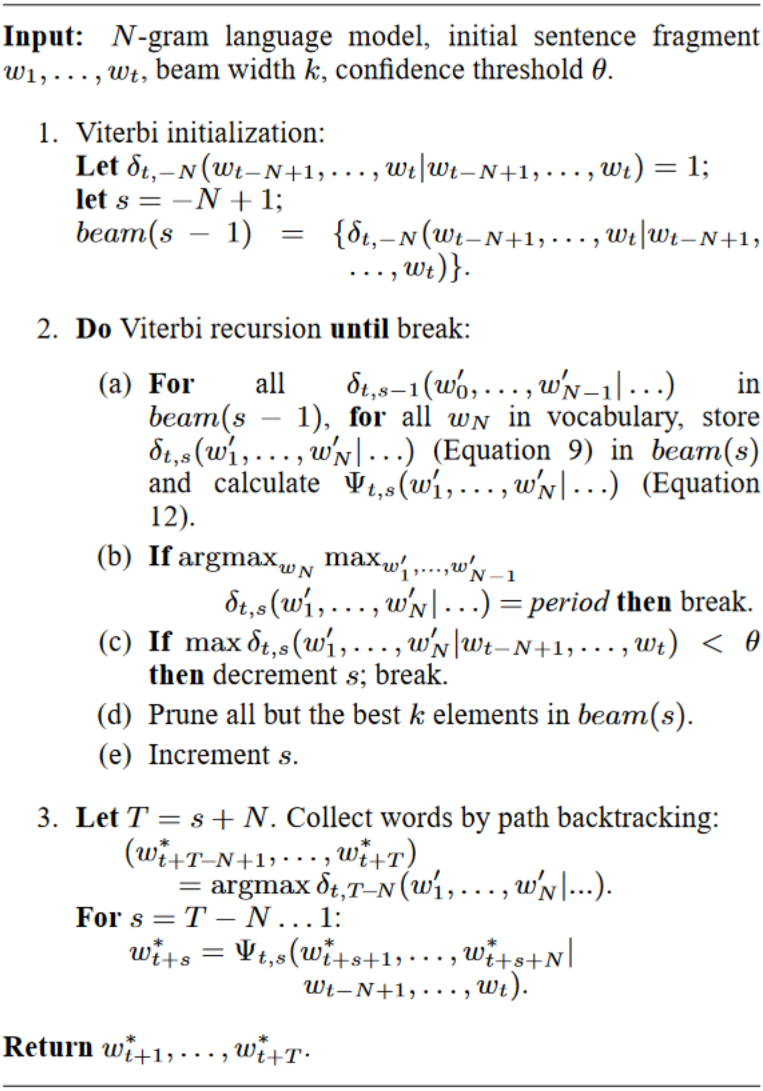
In [1], the authors address the problem of text prediction using an N-Gram language model shown in the following equation:

$$\operatorname{argmax} \prod_{j=1}^{T} P(w_{t+j} | w_{t+j-N+1}, \ldots, w_{t+j-1})$$

The Viterbi principle is used to simplify the problem and improve efficiency. The model attempts to address the issue of predicting subsequent words given some beginning sentence fragment. This feature has the ability to greatly increase productivity in menial repetitive tasks, such as answering emails or entering command line commands in a terminal. The full algorithm is seen below.

Since the purpose is to improve productivity, the model quantifies improvement by considering only perfect predictions. Consider a text prediction for an email draft. If the user accepts the prediction, but still needs to go back to make corrections, then the prediction did not improve efficiency. For this reason, the results only consider a good prediction one in which the prediction was accepted in its entirety and not corrected.

They use an instance-based method of text prediction as their base line and compare that to the performance of their model. They use a diverse corpus of emails, weather reports and recipes. Evaluation is based primarily on two components. They look at precision, which is the ratio of accepted predictions, and they look at recall, which is a ratio that essentially quantifies the number of keystrokes saved by the accepted prediction. Ultimately, their research shows increased precision and recall where training documents have less diversity. For example, service center emails achieved much higher keystroke savings than recipes. However, in all situations, their model provides some benefit in improved efficiency and productivity.



```
Input: $N$-gram language model, initial sentence fragment $w_1, \ldots, w_t$, beam width $k$, confidence threshold $\theta$.

1. Viterbi initialization:
   Let $\delta_{t,-N}(w_{t-N+1}, \ldots, w_t | w_{t-N+1}, \ldots, w_t) = 1$;
   let $s = -N + 1$;
   $beam(s - 1) = \{\delta_{t,-N}(w_{t-N+1}, \ldots, w_t | w_{t-N+1}, \ldots, w_t)\}$.

2. Do Viterbi recursion until break:
   (a) For all $\delta_{t,s-1}(w'_0, \ldots, w'_{N-1} | \ldots)$ in $beam(s - 1)$, for all $w_N$ in vocabulary, store $\delta_{t,s}(w'_1, \ldots, w'_N | \ldots)$ (Equation 9) in $beam(s)$ and calculate $\Psi_{t,s}(w'_1, \ldots, w'_N | \ldots)$ (Equation 12).
   (b) If $\operatorname{argmax}_{w_N} \max_{w'_1, \ldots, w'_{N-1}}$
           $\delta_{t,s}(w'_1, \ldots, w'_N | \ldots) = \textit{period}$ then break.
   (c) If $\max \delta_{t,s}(w'_1, \ldots, w'_N | w_{t-N+1}, \ldots, w_t) < \theta$ then decrement $s$; break.
   (d) Prune all but the best $k$ elements in $beam(s)$.
   (e) Increment $s$.

3. Let $T = s + N$. Collect words by path backtracking:
     $(w^*_{t+T-N+1}, \ldots, w^*_{t+T})$
         $= \operatorname{argmax} \delta_{t,T-N}(w'_1, \ldots, w'_N | \ldots)$.
   For $s = T - N \ldots 1$:
     $w^*_{t+s} = \Psi_{t,s}(w^*_{t+s+1}, \ldots, w^*_{t+s+N} |$
                 $w_{t-N+1}, \ldots, w_t)$.

Return $w^*_{t+1}, \ldots, w^*_{t+T}$.
```

2.2 Advances in NLP applied to Word Prediction

The approach in [2] focuses on individual word prediction rather than sentence completion. Word prediction greatly benefits individuals with cognitive or motor disabilities by greatly reducing the amount of time it takes to write and reducing the amount of potential errors due to misspellings. They developed a method that includes both statistical analysis as well as rule-based features called FastType.

FastType is a user interface designed to decrease necessary keystrokes while improving spelling. The authors improve on the original FastType by implementing DonKey, which is simply an improved interface with an improved underlying language model to improve predictions. Predictions are improved by adding POS n-gram models and Tagged Word n-gram models together in their model. The DonKey user interface simply provides a ranked list of suggestions based on the typed characters. The user can choose a word from the list by clicking a button that corresponds to the word or by using a mouse or other pointing device.

The system is evaluated using three metrics, the ratio of keystrokes saved, the number of keystrokes until the correct word appears in the list and the amount of time that is saved by using the system instead of typing the full word out. They tested using list lengths of 5, 10 and 20. While there was some variation between the list length, the keystrokes saved ratio was between about 46% and 55%, increasing with list length. Keystrokes until completion ranged from 2.55 to 2.06, decreasing with list length. The amount of time saved was about 25% to 29%, increasing with list length. In all areas, their improvements to the language model and their improvements to the user interface produced improved results when compared with current benchmarks at their time.

2.3 A Learning-Classification Based Approach for Word Prediction

In this paper, [3] looks at word prediction as a classification problem. They use machine learning methods with feature extraction techniques that are developed from Mutual Information and Chi-Square techniques. This model is able to produce candidate words from a given context and then classify them to predict the most likely next word.

Given a corpus text, multiple sets of "confusion" words, or words that are often confused, are extracted along with some number of words preceding them. The features they extract are not just the preceding words but are words that are more helpful in telling one confused word from another, which is largely what sets their model apart from other similar feature extraction language models. These features are selected using Mutual Information and Chi-Square techniques and are arranged into feature vectors.

The vectors are used to train the classifier using Support Vector Machines (SVM). Each confused word is used as a class to train the classifier. This is done for each set of confused words, so that they each have a classifier model. Predictions are made by considering the given context and classifying it to the most likely confused word.

The evaluation of the model was done using Naïve Bayes as the baseline and a number of different datasets taken from various websites and databases. They used three confusion sets comprised of various confused words used extensively in word prediction research. The Bayesian approach outperformed their method in only one of the datasets and only by about 1%. The other three datasets saw increases over the Bayesian approach by as much as 7%. Overall, their approach is able to produce high accuracy while considering only small amounts of context.

2.4 Results

Using n-gram language models to predict and generate sentences is largely dependent on the input dataset the model is trained on. Using a dataset comprised of the works of Shakespeare, I was able to successfully implement a functional model that generates text with the distinct flavour and voice of Shakespearean plays. As stated in [1], sentence prediction usefulness requires a diverse set of documents that are "thematically related." This allows the model to be focused to the type of text generation that is required.

3 Papers Between 2010 and 2014

In this section, we will review papers between 2010 and 2014 that offer a variety of aproaches to text generation and/or text prediction. In [4], we look at using synchronous context-free grammar (SCFG) to assist in text-to-text bilingual translation. In [5], we are shown how n-grams with smoothing offer a powerful tool in word prediction with homographic words that occur in great numbers in certain languages. Finally, in [6], we are introduced to Recurrent Neural Networks (RNNs) as a powerful tool in text generation and are shown a variation that allows for reasonable performance allowing one to train large datasets using the model.

3.1 Learning Sentential Paraphrases from Bilingual Parallel Corpora for Text-to-Text Generation

In this paper, [4] research methods to generate paraphrases from bilingual parallel corpora. Key uses for this technique include language translation as well as text compression. Analysis is done on the types of paraphrases and on the sort of transformations they are able to complete with them. They also introduce methods for how one could alter their models to fit it to other text-to-text generation problems. As a result, they are able to show how their model is suitable for a number of text generative tasks while producing excellent results when compared to other methods at that time.

They use a syntactically informed synchronous context-free grammar (SCFG) in their model. The grammar is a 5-tuple and is defined as follows:

$$\mathcal{G} = \langle \mathcal{N}, \mathcal{T}_S, \mathcal{T}_T, \mathcal{R}, S \rangle,$$

where $\mathcal{N}$ is a set of nonterminal symbols, $\mathcal{T}_\mathcal{S}$ and $\mathcal{T}_\mathcal{T}$ are the source and target language vocabularies, $\mathcal{R}$ is a set of rules and $\mathcal{S} \in \mathcal{N}$ is the root symbol. The rule in $\mathcal{R}$ take the form:

$$C \rightarrow \langle \gamma, \alpha, \sim, w \rangle,$$

where the rule's left hand side $\mathcal{C} \in \mathcal{N}$ is a nonterminal, $\gamma \in (\mathcal{N} \bigcup \mathcal{T}_{\mathcal{S}})^*$ and $\alpha \in (\mathcal{N} \bigcup \mathcal{T}_{\mathcal{T}})^*$ are strings of terminal and nonterminal symbols with an equal number of nonterminals $c_{NT}(\gamma) = c_{NT}(\alpha)$ and

$$\sim: \{1 \ldots c_{NT}(\gamma)\} \rightarrow \{1 \ldots c_{NT}(\alpha)\}$$

constitutes a one-to-one correspondency function between the nonterminal in α and γ.

In translation, the bitext is parsed and aligned, and this grammar is used with extraction methods to generate rules used for translation. Weights are assigned to the rules and trained iteratively using a number of features. These can include "phrase translation probabilities, word-for-word lexical translation probabilities, a 'rule application penalty', and a language model probability." Then, an input sentence is translated by using the grammar and the CKY algorithm to find the derivation with the highest probability.

In paraphrasing, they simply make minor adjustments to the translation rules of the grammar and use lexical and phrasal probability features. They are basically accomplishing paraphrasing by creating an English to English translation. This methods allows for a number of text-to-text applications including text simplification, compression, query expansion.

Their method was compared to state-of-the-art approaches at their time. They also used human judges to review their output. Judges were asked to consider grammar and meaning. When reviewing compressed (paraphrased) sentences, were grammatically correct while still maintaining their original meaning. Their results were promising showing that their compression methods did comparable or better than other approaches.

3.2 Probabilistic Analysis of Sindhi Word Prediction using N-Grams

We return to n-gram modeling in this paper where [5] use bigram, trigram and 4-gram models with Laplace smoothing within the context of Sindhi word prediction. They capitalize on the probabilistic approach of n-grams in order to distinguish between homographic words which occur regularly in the Sindhi language. Due to the morphological ambiguity created from the lack of diacritics in most application of written text, the information contained in n-gram modeling is well suited for word prediction in this language. Results are based mainly on perplexity however, comparative studies on other languages with morphological structure are also looked at.

The n-gram model is based on the probability of a word given it's preceding word, $P(w_n|w_{n-1})$. However, this bigram model can be generalized to n-gram as follows:

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-N+1}^{n-1})$$

Smoothing is used by adding one to any zero probability bigrams in the model as follows:

$$C_i^* = (C_i + 1)\frac{N}{N+V}$$

They use a back-off technique where if a 4-gram is not present, they look for a trigram and use that probability. If that is not present, then they look for a bigram and use that probability. They use number of media to build their corpus including books, newspapers, internet etc... An algorithm is used to tokenize the Sindhi words. Bigram, trigram and 4-gram models are created and stored in three separate databases before adding the smoothing.

Results were promising for all models but show that 4-gram is best suited for the Sindhi language. The perplexity of the 4-gram model was 69.07, which is almost half of the bigram model at 109.89. Since Sindhi is orthographically similar to other languages like Arabic, Urdu or Persian, [5] provides strong evidence that probabilistic analysis will also be useful for those as well.

3.3 Generating Text with Recurrent Neural Networks

In [6], Recurrent Neural Networks (RNNs) are used to make next character predictions based on the previous sequence of characters. The approach used makes use of the Hessian-Free optimizer (HF) to simplify training. A slight change to the standard RNN architecture is made to improve performance by using multiplicative connections ”which allow the current input character to determine the transition matrix from one hidden state vector to the next.”

The multiplicative RNN, dubbed MRNN by the authors, did comparatively well against other character predicting methods but what stood out most interesting was it’s ability to generate text with high-level linguistic structure and grammatical structure. The standard RNN with input vectors $(x_1, ..., x_T)$, hidden states $(h_1, ..., h_T)$ and outputs $(o_1, ..., o_T)$ is iterates from $t = 1$ to T as follows:

$$h_t = tanh(W_{hx}x_t + W_{hh}h_{t-1} + b_h)$$
$$o_t = W_{oh}h_t + b_0)$$

Where W_{hx} is the input-to-hidden weight matrix, Whh is the hidden-to-output weight matrix and b_h and b_0 are biases. Exploding and vanishing gradient is managed through the use of the HF optimizer, which is shown to work well for their application. The ”MRNN computes the hidden state sequence $(h_1, ..., h_T)$, an additional ’factor state sequence’ $(f_1, ..., f_T)$ and output sequence $(o_1, ..., o_T)$ by iterating the equations:”

$$f_t = diag(W_{fx}x_t) \cdot W_{fh}h_{t-1}$$
$$h_t = tanh(W_{hf}f_t + W_{hx}x_t)$$
$$o_t = W_{oh}h_t + b_0$$

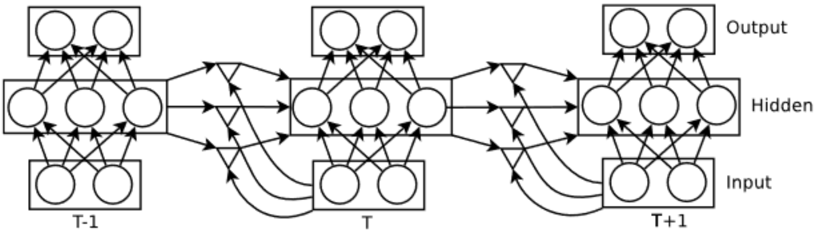


Fig. 1: "The Multiplicative Recurrent Neural Network 'gates' the recurrent weight matrix with the input symbol. Each triangle symbol represents a factor that applies a learned linear filter at each of its two input vertices. The product of the outputs of these two linear filters is then sent, via weighted connections, to all the units connected to the third vertex of the triangle. Consequently every input can synthesize its own hidden-to-hidden weight matrix by determining the gains on all of the factors, each of which represents a rank one hidden-to-hidden weight matrix defined by the outer-product of its incoming and outgoing weight-vectors to the hidden units. The synthesized weight matrices share 'structure' because they are all formed by blending the same set of rank one matrices. In contrast, an unconstrained tensor model ensures that each input has a completely separate weight matrix."

Figure 1 is taken directly from the paper and provides a detailed explanation of the network. The difficulty with this approach is with the product of weights in gradient descent that can result is artificially high derivatives. However, the HF optimizer offers the solution to this problem as a 2nd order method that is particularly good at dealing with those kinds of issues.

The MRNN is trained along side standard RNN as well as a few other current state-of-the-art language modelers on three different datasets. The datasets used were a cleaned up dump of Wikipedia, a collection of articles from the New York Times and a corpus of machine learning papers downloaded and converted to text. The results were promising showing that the MRNN did perform comparatively to the other methods placing slightly better the one while slightly less than the other when comparing performance using bits per character (bpc). The author's also produce various text generated by the three different models to show MRNN's ability to "learn" information about the input corpus. For example, when using the Wikipedia model and prompted with a list of countries, the MRNN model recognizes the input is a list and generates text accordingly.

[6] propose a novel approach to RNNs that solve computational difficulties the made use of model virtual impractical at the time. Using their MRRN approach, they were able to manage performance and train large datasets comparatively well consider other current methods. Further, they were able to show the raw power of RNN models in their ability to learn from the input and model their corpus with a high level of accuracy

4 Papers Between 2015 and 2019

This section explores data mining to predict user entries in user entry forms in [7]. In [8], a convolutional neural network (CNN) using an attention mechanism is used to predict medical codes given medical notes. Finally, text generation is approached using inverse reinforcement learning (IRL) model in [9]. Also, results for the IRL using the synthetic oracle dataset are given at the end of the section.

4.1 Predicting user entries by using data mining algorithms

A number of applications for text prediction have been discussed so far. Here, [7] tests data mining with a number of methods to make data entry systems easier and more user-friendly. Naive Bayes, Rule Induction, K-NN and Decision Tree methods are tested for high predictive accuracy in an effort to make Government decision system more efficient and easier to use.

The dataset used was taken from the Government decision system of the Palestinian Ministers Council. Data mining preparation and preprocessing was performed on the dataset to prepare it. The methods used were Replace Missing Values, Select Attributes, and Filter Examples. After preprossing, association rules are set up using two methods, FP-Growth and Create Association Rules. Once the dataset was ready, it is used to create classification models that will predict certain entries in the system based on the user initial entries, i.e., as they input data at the beginning of the form, it makes suggestions about entries in other areas of the form. The data was ran through the four models noted earlier. The equation for Naive Bayes classification is given below as it performed best.

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

While results showed Naive Bayes outperforming the other methods with K-NN coming in second, accuracy overall stayed somewhat low at a max of 68.41%. The author's noted the large number of class's and the varying number of possible inputs to the system. Overall, the process and methods show promise as another strong method for making text predictions that ultimately make system usage easier and an end user's use of it more user-friendly.

4.2 Explainable Prediction of Medical Codes from Clinical Text

Assigning proper medical codes can be a challenging task due to the large volume of available codes and the fact that the medical texts are doctor's notes, which tend to have non-standard abbreviations, irrelevant information and even misspellings. In this paper, [8] propose a convolutional neural network (CNN) that uses attention to predict medical codes given medical

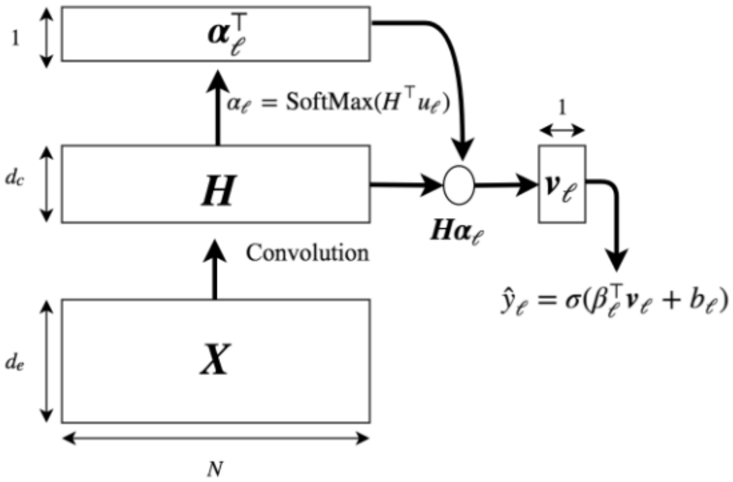

$\alpha_\ell^\top$
1
$\alpha_\ell = \text{SoftMax}(H^\top u_\ell)$
d_c
H
Convolution
d_e
X
N
$H\alpha_\ell$
1
v_ℓ
$\hat{y}_\ell = \sigma(\beta_\ell^\top v_\ell + b_\ell)$

Fig. 2: CAML Architecture

text as input. The datasets used are two versions of MIMIC, which are text discharge summaries from intensive care unit stays. They call their method Convolutional Attention for Multi-Label classification (CAML). The attention mechanism allows the model to provide explanations for the predictions by assigning importance values to n-grams from the input.

Medical code predictions are modeled as a multi-label text classification problem. The attention mechanism is used to choose the parts of the text that are the most likely for each of the possible medical codes. A sigmoid activation function is used to transform the output into a probability. The model's architecture is shown in figure 2.

The model is evaluated using a number of methods including baselines of a one-dimensional CNN, bag-of-words (BOW) logistic regression and bidirectional gated recurrent unit (Bi-GRU). Various parameters are tuned using an optimization package. Metrics used for evaluation include area under the ROC curve (AUC), F1 and precision 8. They use micro and macro metrics for AUC and F1 where they treat each text/code pair as an individual prediction, or they average the per-label metrics. Prediction results are also provided to physicians to provide feedback on the predicted codes as well as how informative the k-grams were that were used by the attention mechanism to make the prediction.

Final results show CAML producing better results in all of the metrics for all but one of the datasets that were tested. Their approach offers a strong model able to predict useful medical codes. Further, the model contains an attention mechanism that is able to provide explanation to why a particular prediction was made. Because of these feature, the model could be a strong building block on which future automated medical coding could be built.

4.3 Toward Diverse Text Generation with Inverse Reinforcement Learning

In this paper, [9] approach text prediction using inverse reinforcement learning (IRL). The model trains a reward function and an optimal policy alternating back and forth. Training this way allows the model to generate instant rewards at each step creating more dense reward signals. The generation policy works by sampling one word at a time to create sequences of text. The benefits of this approach is that it creates high quality diverse text.

The generated text sequence is given as $s_{1:T}$. The probability of generating the text sequence is given by:

$$q_\theta(x_{1:T}) = \prod_{t=1}^{T-1} \pi_\theta(a_t = x_{t+1} | s_t = x_{1:t})$$

There are two steps to the IRL model. Train the reward function, and then, train the optimal policy that maximizes the reward to generate text. Figure 3 shows an illustration of the model.

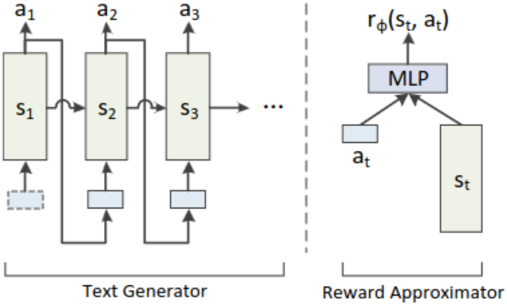


Fig. 3: Illustration of the model

The reward function is given by:

$$\mathcal{J}_r(\phi) = \frac{1}{N}\sum_{n=1}^{N} \log p_\phi(\tau_n) = \frac{1}{N}\sum_{n=1}^{N} R_\phi(\tau_n) - \log Z$$

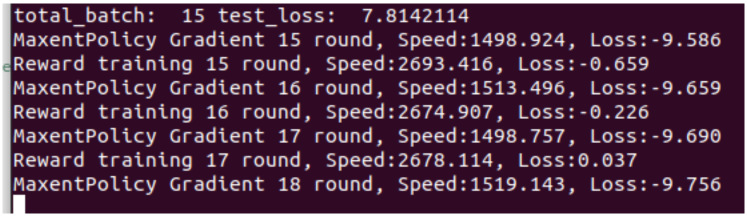
where τ_n denotes the n^{th} sample in the training set. The text generator is modeled after a long short term memory (LSTM) network. Training follows the same steps outlined previously where the model iteratively updates the reward based on the token, then, generates a token based on that reward and then repeats.

The model is trained on three datasets. They use the synthetic oracle dataset, COCO image captions dataset and IMDB movie reviews dataset. The models are evaluated differently for each of the datasets. The synthetic oracle dataset is evaluated using the average negative log-likelihood (NLL). When compared to state-of-the-art maximum likelihood estimation (MLE) models, the IRL model has a lower NLL score, which indicates a higher quality of the generated text sequences. The COCO image captions dataset is evaluated using a number of differnt BLEU metrics. In most areas, the IRL outperformed the MLE models. The exception being the BLEU metric which seeks to measure the precision of the generator. One MLE model did outperform the IRL model. This is believed to be due to the fact that the leading MLE model has a tendency to repeat safe text sequences while the IRL model produces more diverse texts. Finally, The IMDB move reviews dataset is evaluated using the same BLEU metrics as the COCO dataset. However, it is also evaluated by humans where sentences are asked to score them, and the scores are averaged out for the sentences. In either case, their model outperformed the others by producing high quality diverse text sequences.

This paper approaches text generation using the inverse reinforcement learning (IRL) model where reward and token generation are updated at each step of training. The model is trained on a diverse set of datasets and is evaluated using a number of metrics including human opinion. In most areas, the model outperforms current state-of-the-art models. The model's success is in it's ability to generate texts that are diverse and high quality.

4.4 Results

Figure 4 shows current, in progress, results of the IRL model. I made need to cut this short as my laptop is super slow.



```
total_batch:  15 test_loss:  7.8142114
MaxentPolicy Gradient 15 round, Speed:1498.924, Loss:-9.586
Reward training 15 round, Speed:2693.416, Loss:-0.659
MaxentPolicy Gradient 16 round, Speed:1513.496, Loss:-9.659
Reward training 16 round, Speed:2674.907, Loss:-0.226
MaxentPolicy Gradient 17 round, Speed:1498.757, Loss:-9.690
Reward training 17 round, Speed:2678.114, Loss:0.037
MaxentPolicy Gradient 18 round, Speed:1519.143, Loss:-9.756
```

Fig. 4: Current Process of IRL Model. This model will take some time to train.

5 Papers Since 2020

TODO

6 Conclusion

TODO

References

[1] Bickel, S., Haider, P., Scheffer, T.: Predicting sentences using n-gram language models. In: Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, pp. 193–200 (2005)

[2] Aliprandi, C., Carmignani, N., Deha, N., Mancarella, P., Rubino, M.: Advances in nlp applied to word prediction. University of Pisa, Italy February (2008)

[3] Al-Mubaid, H.: A learning-classification based approach for word prediction. Int. Arab J. Inf. Technol. **4**(3), 264–271 (2007)

[4] Ganitkevitch, J., Callison-Burch, C., Napoles, C., Van Durme, B.: Learning sentential paraphrases from bilingual parallel corpora for text-to-text generation. In: Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pp. 1168–1179 (2011)

[5] Mahar, J.A., Memon, G.Q.: Probabilistic analysis of sindhi word prediction using n-grams. Australian Journal of Basic and Applied Sciences **5**(5), 1137–1143 (2011)

[6] Sutskever, I., Martens, J., Hinton, G.E.: Generating text with recurrent neural networks. In: Proceedings of the 28th International Conference on Machine Learning (ICML-11), pp. 1017–1024 (2011)

[7] Alhaj, B.A., Maghari, A.Y.: Predicting user entries by using data mining algorithms. In: 2017 Palestinian International Conference on Information and Communication Technology (PICICT), pp. 110–114 (2017). IEEE

[8] Mullenbach, J., Wiegreffe, S., Duke, J., Sun, J., Eisenstein, J.: Explainable prediction of medical codes from clinical text. arXiv preprint arXiv:1802.05695 (2018)

[9] Shi, Z., Chen, X., Qiu, X., Huang, X.: Toward diverse text generation with inverse reinforcement learning. arXiv preprint arXiv:1804.11258 (2018)