CSCM38: Adv Topic - Artificial Intelligence and Cyber Security - Coursework 1

Andy Gray
445348

10/11/2020

1 Introduction

We will be looking at some issues surrounding an advanced topic within natural language processing (NLP). NLP is a form of artificial intelligence (AI) that aims to as the automatic manipulation of natural language, like speech and text, by software [3]. However, human language is highly ambiguous, ever-changing and evolving. People are great at producing language and understanding language, allowing them to be capable of communicating, perceiving, and understanding very complex and nuanced meanings. At the same time, while we humans are great users of language, we are also not very good at formally recognising and explaining the rules that dictate our language [9]. So if the human language is difficult for humans, the process, therefore, can not be straight forward for computers either. However, some advancements over the years, like sentiment analysis has allowed us to analyse general moods, and speech-to-text has genuinely revolutionised the way people can interact and create documents.

A Swiss linguistics professor in the 1900s, Ferdinand de Saussure, created the concept of "Language as a Science [15]." However, Professor Saussure, around 1911, offered three courses at the University of Geneva. At this university is where it got developed as a proposal for describing languages as "systems." Within the language, a sound represents a concept, a concept that shifts its meaning as the context changes [6]. It was not until the 1950s where a British mathematician named Alan Turing wrote a paper, laying out a test for a "thinking" machine. In his paper, he said that if a machine could be part of a conversation, and it imitated a human so well that there were no noticeable differences. The machine could be considered capable of thinking [22]. However, the Hodgkin-Huxley model showed how the brain uses neurons in forming an electrical network. These events helped inspire the idea of AI, NLP, and the evolution of computers [6].

In 1957 previous linguistic concepts got revolutionised, concluding that for a computer to understand a language, the sentence structure would have to be changed [5]. These revolutions created the style of grammar called Phase-Structure Grammar, which methodically translated natural language sentences into a format that is usable by computers [6].

In the 1960s is when research into AI and NLP stopped due to the technology not being where it needed to be and costing more than hiring people to translate [6]. It took until the 1980s for NLP and AI [14] research to resume after the failed expectations in the earlier years [6]. However, it was not until the 1990s, where statistical models for NLP popularity started to grow. The pure statistics NLP methods have become remarkably valuable in keeping pace with the vast amounts of online text [6]. N-Grams have become useful, recognising and tracking clumps of linguistic data, numerically [16]. In 1997, LSTM and recurrent neural net (RNN) models [17] were introduced and found their niche in 2007 for voice and text processing. Currently, neural net models are considered the cutting edge of research and development in the NLP's understanding of text and speech generation [6].

With the growth of RNN, these helped advance the field of NLP with ML. However, it got quickly found out that they had several issues, including the vanishing gradient problem, and it only has a short term memory. This problem brought about the introduction of LSTMs, and then later a simplified version called a Gated Recurrent Units (GRU). With the introduction of RNN and the growth with popularity with frameworks like TensorFlow and PyTorch. We will be proposing a project plan that will be looking at LSTMs and GRU cells to see which one performs better on a given dataset about natural disasters [13] using the TensorFlow deep learning library [**?**].

First, we will look into the related work, covering what NLP is and some of its overviews, then we will look at the more recent advancements of NLP and look into how they work and some of their drawbacks. These techniques include RNN, LSTMs and GRU. Additionally, we will be looking at what advantages they are claiming to have over each other.

2 Related Work

ML for NLP and text analytics involves using ml algorithms and "narrow" AI [2]. These approaches allow the computer to understand the meaning of text documents, which can be about anything that contains the text. Some examples include social media comments, online reviews, survey responses, even financial, medical, legal and regulatory documents. Therefore the aim of ML and AI in NLP is to improve, accelerate and automate the underlying text analytics functions and NLP features that turn any given unstructured text into useable data and insights [2].

Generally speaking, NLP breaks down the language into shorter, more basic pieces. These shorter pieces are called tokens which can get made up of words and periods, for example. The NLP will then attempt to understand the relationships of the tokens, which the process often uses higher-level NLP features, such as [6] content Categorisation, which is a process that creates a linguistic document summary that includes content alerts, duplication detection, search, and indexing. Topic Discovery and Modeling that aims to capture the themes and meanings of text collections and applies advanced analytics to the text. Another process is that contextual extraction, which will automatically pull structured data from text-based sources. Sentiment Analysis aims to identify the general mood, or subjective opinions, stored in large amounts of text. Use-

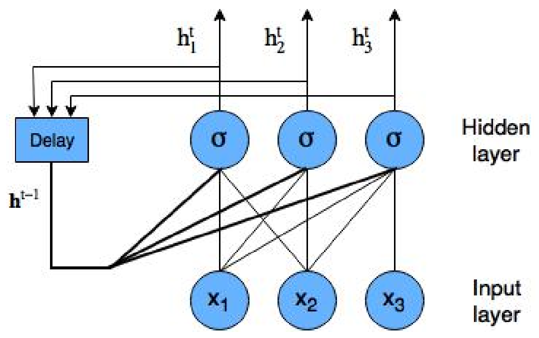

Figure 1: A reprentation of a RNN [1]

ful for opinion mining. Text-to-Speech and Speech-to-Text Conversion process aim to transform voice commands into text, and vice versa. Document summarisation is a process that automatically creates a synopsis, condensing large amounts of text. Machine Translation is another process that automatically translates the text or speech of one language into another.

2.1 Recurrent NN (RNN) for NLP

A recurrent neural network (RNN), at its most fundamental level, is simply a type of densely connected neural network (NN). However, the main difference to a dense fully connected NN and an RNN is the introduction of time into the model. What this means is that the output of the hidden layer in an RNN gets fed back into itself [1, 8].

In fig: 1, we have a simple RNN, also referred to as a vanilla RNN [8], which has three input nodes. As usually expected in a dense NN, the hidden layers have sigmoid activation functions, from the output of each previous layer. However, where an RNN changes to a dense NN is that the output of the hidden layer then gets fed back into the same hidden layer. This step allows the hidden layer outputs to get passed through a conceptual delay block to allow the input of h^{t-1} into the hidden layer. By doing this within the RNN, it allows the model to be able to model time or sequence-dependent data [1, 8].

Recurrent neural networks are very flexible [8]. In the implementation shown in fig: 2, we have a many-to-many model. What this means is that we have the input sequence "A girl walked into a bar..." and we also have many outputs. These outputs can be notated as h_0 to h_t [1]. However, we can also have other setups, for example, one-to-many and many-to-one [?]. One-to-many is the process of supplying one input and predicting multiple outputs. What this approach is trying to do is to generate sentences based on a single input, or a single word. Additionally, a many-to-one is the process of supplying many words as input, like the sentence, and the model will then predict the next word [1].

However, one thing to take into account is that a vanilla RNN does not get used often in practice [11]. One of the reasons being is the vanishing gradient

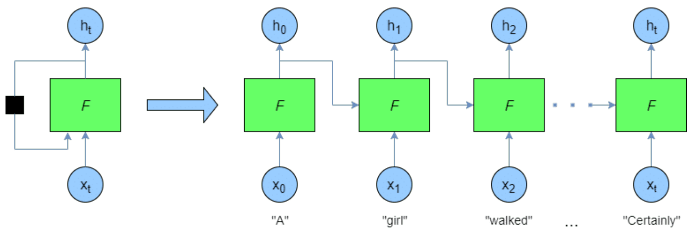

Figure 2: A reprentation of a many to many RNN [1]

problem [8]. The vanishing gradient is where the contribution from the earlier steps becomes insignificant in the gradient descent step [20]. For RNN to work effectively, we would want to have long memories. We would want this longer-term memory so the network can connect data relationships at significant distances in time, therefore having a more extended memory than currently on offer. Due to the lack of long term memory RNN get referred to as a short term memory network. Long term memory networks could make real progress in understanding how language and narrative works or how stock market events get correlated, for example. However, due to the limitations of the RNN, the more time steps we have, the more possibility we have of back-propagation gradients either accumulating and exploding or vanishing down to nothing [1]. The exploding gradient happens when the algorithm assigns high importance to the weights, without much reason. However, this problem can be solved by truncate or squash the gradients [20].

2.2 LSTM for NLP

Long Short-Term Memory (LSTM) is a specific RNN architecture that got designed to model temporal sequences and their long-range dependencies more accurately than conventional RNNs [21, 11]. LSTM RNNs have shown to be more effective than Deep NNs (DNNs) and conventional RNNs. Especially for acoustic modelling [21]. LSTM cell got first proposed in 1997 [12] but has gradually got improved over the years [8, 21, 23].

How LSTM works is that the cell within the RNN has two vectors, $h_{(t)}$ and $c_{(t)}$. Therefore the best way to think of this is that c the cell, $c_{(t)}$ is the long term state and $h_{(t)}$ is the short term state [8]. When visually conceptualising the network, all RNN have the form of a chain of repeating modules of NN. In vanilla RNNs, this repeating module will have a straightforward structure, such as a single tanh layer. LSTMs also have this chain-like structure. However, instead of having a single NN layer, LSTM has four, interacting in a very different way to a vanilla RNN [18].

The key to LSTMs is the cell state, the horizontal line running through the top of the diagram (see fig: 3). The cell state runs straight down the entire chain, with only some minor linear interactions. They are therefore allowing for the information to flow through the cells, unchanged if needed. However, LSTM does have the ability to remove or add information to the cell state.

These changes get done by the structures called gates. Gates are a way to let

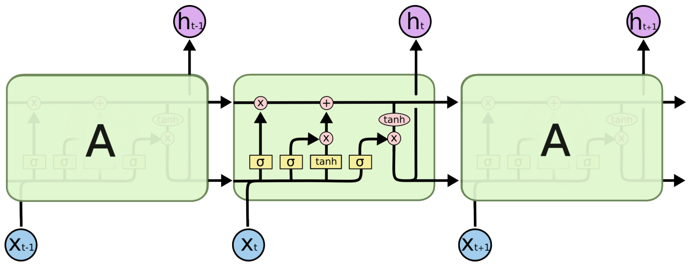

Figure 3: A reprentation of a many to many RNN [18]

information through optionally, depending on the criteria set, which get formed out of a sigmoid NN layer and a pointwise multiplication operation [18]. The sigmoid layer outputs numbers between zero and one, describing how much of each element should get let through. A value of zero means will let nothing through, while a value of one means everything will get through [18, 8]. An LSTM has three of these gates, to protect and control the cell state [18]. The three gates are a forget gate, input gate, and output gate [19, 8]. The forget gate, which is controlled by the $f_{(t)}$ determines what part of the long-term state should get forgotten. While the input gate, controller by the $i_{(t)}$, decides on which parts of $g_{(t)}$ should get added to the long-term state. Then the output gate, controlled by the $o_{(t)}$ figures out what part of the long-term state should be read and outputted through the time step, both to $h_{(t)}$ and $y_{(t)}$ [8].

Overall an LSTM RNN will perform better than a vanilla RNN. Training will converge faster, and it will detect long-term dependencies in the data [8]. LSTMs have made remarkable developments in the fields of language modelling, text generation, Image processing, speech and handwriting recognition, music generation and language translation [7]. However, LSTMs do have a few drawbacks [7]. While LSTMs became popular because they can solve the vanishing gradient, which they can, they do not though remove it altogether. The problem gets caused by the model still having to move data from one cell to another. Another drawback is that they require a lot of resources and time to train. Additionally, with the rise of big data, developers are looking for a model that can remember the context for a lot longer than what LSTMs can now. LSTMs can be prone to overfitting and can get affected easily by different random weight initialisations.

2.3 GRU for NLP

GRU stands for Gated Recurrent Units, and despite being quite similar to LSTMs but a more simplified version of the cell [8, 11], GRUs have never been as popular as they are now [7, 8]. However, GRUs are a simpler cell compared to an LSTM but perform just as well as the LSTM [10]. GRUs are an RNN that have a gated mechanism to obtain dependencies of different time scales as effectively and adaptively as possible [4].

GRUs have an update gate, and a reset gate (see fig: 4), which is responsible for selecting what knowledge is to get carried forward and the rest gate is in

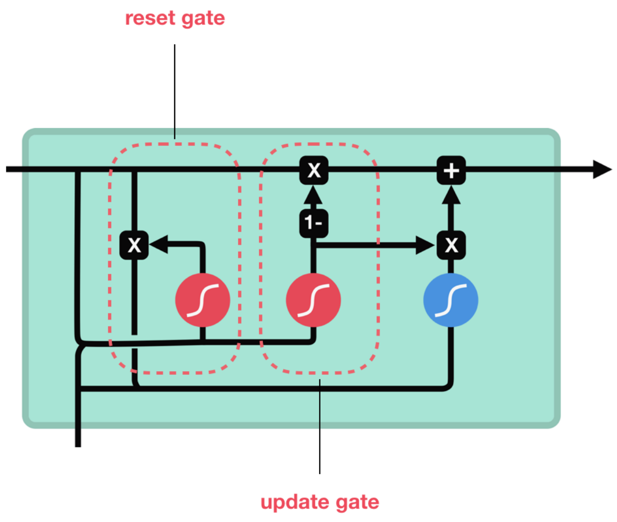

Figure 4: A reprentation of GRU cell [19]

between the two successive recurrent units. This gate decides on how much information needs to get forgotten. A single gate controller $z_{(t)}$ controls the forget and the input gate [8]. Another striking aspect of GRUs is that they do not store cell state in any way. Therefore, there is no output gate. The full state vector gets outputted at every time step [8]. Therefore they are unable to control how much memory the next unit will get given [7].

GRUs provide most of the same benefits of LSTMs but have fewer tensor operations and are therefore a little speedier to train then LSTMs [19]. While GRUs are a crucial reason RNN are a success at NLP, they do however have limited short-term memory and also have difficulty learning long-term patterns in sequences of 100-time steps or above [8].

3 Project Plan

While both GRUs have their benefits and drawbacks, there is not a clear winner which one is better. Researchers and engineers usually try both to determine which one works better for their use case [19]. So with this in mind, we are proposing an experiment to determine which one is better at a classification of text problem. We aim to see which one performs better for the provided NLP dataset out of an LSTM and GRU.

We want to be able to see what are the noticeable differences or benefits over using an LSTM over a GRU or vice versa. We will be using the "Real or Not? NLP with Disaster Tweets [13]" dataset to test the two RNN cell types. The data has two CSV files, training and testing. However, we will be using the training dataset only and splitting this data. The training dataset contains the text of a tweet; a keyword from that tweet; the location the tweet got sent.

However, we must note that some of this information might be blank, and some of these attributes might get discarded during the preprocessing stage. While the network aim is to predict if the given tweet is tweeting about a real disaster, with a "1" label, or not, "0" label.

In order to decide on what cell performed better, we will be using several metrics. The metrics that will get used to comparing the different cells are the time it takes to train the network, the training loss and validation loss values, the RMSE and MAE values. We also intended to train and run the experiment at least five times with each changed parameters of the networks, and this is to allow us to see if there is a consistent trend within the results or if they change depending on the set parameters used.

References

[1] Adventures in Machine Learning. Recurrent neural networks and lstm tutorial in python and tensorflow, 2020.

[2] Barba, P. Machine learning (ml) for natural language processing (nlp), 2020.

[3] Brownlee, J. What is natural language processing?, 2019.

[4] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[5] Chomsky, N. *Syntactic structures*. Walter de Gruyter, 2002.

[6] Foote, K. D. A brief history of natural language processing (nlp), 2019.

[7] Geeks for Geeks. Understanding of lstm networks, 2020.

[8] Géron, A. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[9] Goldberg, Y. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies 10*, 1 (2017), 1–309.

[10] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems 28*, 10 (2016), 2222–2232.

[11] Grus, J. *Data science from scratch: first principles with python*. O'Reilly Media, 2019.

[12] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[13] Kaggle. Real or not? nlp with disaster tweets, 2020.

[14] Knight, M. What is artificial intelligence (ai)?, 2018.

[15] Koerner, E. F. *Ferdinand de Saussure: Origin and development of his linguistic thought in western studies of language*, vol. 7. Springer-Verlag, 2013.

[16] Kumar, P. An introduction to n-grams: What are they and why do we need them?, 2017.

[17] Le, J. Recurrent neural networks: The powerhouse of language modeling, 2019.

[18] Olah, C. Understanding lstm networks, 2015.

[19] Phi, M. Illustrated guide to lstm's and gru's: A step by step explanation, 2018.

[20] purnasai gudikandula. Recurrent neural networks and lstm explained, 2019.

[21] Sak, H., Senior, A. W., and Beaufays, F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling.

[22] Turing, A., Braithwaite, R., Jefferson, G., and Newman, M. Can automatic calculating machines be said to think?(1952). *B. Jack Copeland* (2004), 487.

[23] Zaremba, W., Sutskever, I., and Vinyals, O. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* (2014).