CS230

Recursive Neural Network for Generating Novel Brand Names for Therapeutic Medicines

Zeke Randolph
Department of Computer Science
Stanford University
`uniment@stanford.edu`

Abstract

Developing prescription therapeutic drugs is an immensely expensive process, costing an average of $2.6Bn.[1] Drug names must be attractive to market these products and recoup the costs of R&D expenditures.

At the same time, drug naming is a matter of public health, and is heavily regulated, requiring government agency approval. Medical error, including drug name and dose mistakes, is the third leading cause of death in the U.S. after heart disease and cancer[2], and errors in prescribing, dispensing, and taking medications results in 1.5 million Americans sickened, injured, or killed each year.[3] As a result, “name safety" is a chief priority.

It is essential to develop drug names which are unique and do not evoke associations with other substances to avoid scenarios in which the wrong drug could be consumed. Due to the high stakes involved in drug naming, specialized “name engineering" creative agencies are employed to develop brand names which are both appealing in the marketplace and highly unique.

To facilitate the creation of novel drug names, a LSTM RNN has been developed and employed to randomly generate unique drug names, potentially easing the expensive process of naming drugs.

1 Overview

Text processing, such as name generation, is inherently a time-sequence process, best handled by recurrent neural networks (RNNs). To enable drug brand name generation, a long short-term memory (LSTM) RNN neural network was used.

Recognizing that existing drugs already follow the guidelines that would be required of new drug names, it was determined that their names would serve as a suitable template by which new names could be created. A list of existing drug names was compiled and fed into the LSTM model for parameter training, and then the model was deployed to create randomly-generated novel names.

2 Dataset

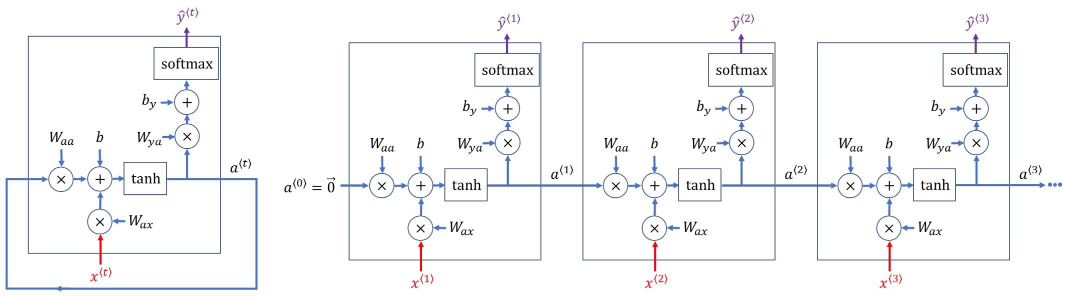
To create a database of existing drug names, data from the NIH[4] was manually compiled and curated into a list of 2,805 brand names. Before feeding the drug names into the neural network, they were processed into a format amenable to matrix mathematics. To do this, each character of each name was converted into a one-hot vector representation wherein the position within the vector representing the character’s position in the alphabet was set to 1, while all other values were 0.

3 LSTM Recurrent Neural Network Model

To create a model that could accurately predict drug names, a LSTM RNN model was implemented as illustrated in Figure 1. The model takes an input character sequence x and produces an output character sequence $\hat{y}$ (or, more properly, a character probability sequence) using a combination of weights (W_{aa}, W_{ax}, and W_{ya}) and biases (b and b_y), and activation functions ($\tanh$ and $\mathrm{softmax}$).

Generally speaking, neural networks create a mapping between x and $\hat{y}$, and by comparison of $\hat{y}$ with a "ground truth" sequence y the model can be trained to produce an accurate mapping. In this case, the intended mapping is a simple $y = x$, so that the LSTM can be trained to predict the drug name that is being fed into it.

As shown in Figure 1(a), the structure of the model is recursive, in that only one neural network cell is used at a time, processing a single character $x^{\langle t \rangle}$ of the input sequence to produce a single character $\hat{y}^{\langle t \rangle}$ of the output sequence. To carry information about previous characters, the cell also produces information stored in a "hidden state" vector $a^{\langle t \rangle}$, which is generated at each timestep and used for the next letter.



(a) Recursive Architecture (b) Unwrapped Loop for Illustration

Figure 1: LSTM RNN Architecture

To illustrate the process, Figure 1(b) shows the recursive loop unwrapped into three representative timesteps. The internal state is initialized to an initial value $a^{\langle 0 \rangle}$ (an arbitrary but consistent vector of zeros), and then the sequence proceeds.

The first letter of the input sequence $x^{\langle 1 \rangle}$ is multiplied by weight matrix W_{ax} and summed with the bias vector b and the product of the internal state $a^{\langle 0 \rangle}$ and weight matrix W_{aa}, and the result is fed into a $\tanh$ activation function. The choice of a $\tanh$ activation function is useful for its easily differentiable behavior, its output saturation characteristic, and its odd symmetry.

Activation by the $\tanh$ function produces the hidden state vector $a^{\langle 1 \rangle}$, ready to be used in the next timestep. This value is also used to produce the first character of the output $\hat{y}^{\langle 1 \rangle}$, by first multiplying by a weight matrix W_{ya}, adding bias vector b_y, and activating with a $\mathrm{softmax}$ function (where for a vector a, $\mathrm{softmax}\ (a) = \frac{e^{a-\max(a)}}{\sum e^{a-\max(a)}}$). The $\mathrm{softmax}$ activation function is useful because it takes an input vector of positive or negative values, and outputs a vector of values in the interval $(0, 1)$ that sum to 1, and can therefore be interpreted as a probability distribution describing the probability that the output character will take on any particular value of the alphabet.

Having produced output letter $\hat{y}^{\langle 1 \rangle}$, the next value of the input sequence $x^{\langle 2 \rangle}$ and the previously generated internal state $a^{\langle 1 \rangle}$ are fed into the network again to produce $\hat{y}^{\langle 2 \rangle}$ and $a^{\langle 2 \rangle}$. The loop repeats, taking input $x^{\langle 3 \rangle}$ to produce $\hat{y}^{\langle 3 \rangle}$ and $a^{\langle 3 \rangle}$ and so on, until the entire input sequence is exhausted.

To reiterate, the input $x^{\langle t \rangle}$ is a one-hot vector whose size is equal to the alphabet length and whose value represents the t^{th} character of the input, while output $\hat{y}^{\langle t \rangle}$ is a probability distribution vector, also of size equal to the alphabet length, which describes the probability that any particular letter will exist for the t^{th} output character.

4 Name Generation

To use the LSTM RNN for name generation, its architecture is slightly modified as shown in Figure 2, while keeping the neural network cell structure and parameters intact. In contrast with the structure of Figure 1, which had as its input x a sequence of characters from an existing drug name, the name generator of Figure 2 generates its input characters based on its own previous outputs.

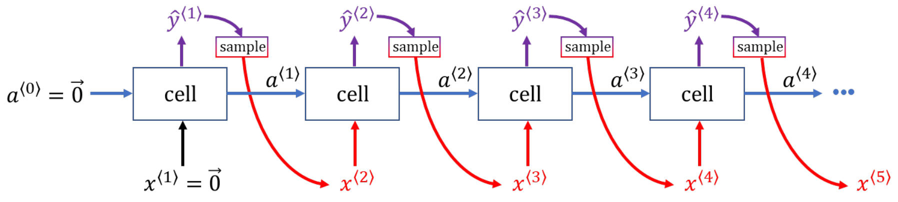


Figure 2: LSTM RNN Name Generator

To start the generator, inputs $x^{\langle 1 \rangle}$ and $a^{\langle 0 \rangle}$ are initialized to zero and fed into the cell. For a well-trained network, the resulting output $\hat{y}^{\langle 1 \rangle}$ is a vector probability distribution describing the probabilities of any particular letter being the first character of a drug name. A letter is randomly sampled from the alphabet based on the probabilities described by $\hat{y}^{\langle 1 \rangle}$, and it is selected to be the first character, $x^{\langle 2 \rangle}$, of the generated name.

$x^{\langle 2 \rangle}$ and $a^{\langle 1 \rangle}$ are then fed into the cell as inputs, producing $\hat{y}^{\langle 2 \rangle}$ which describes the probabilities for the second letter, and internal state $a^{\langle 2 \rangle}$. The second generated letter $x^{\langle 3 \rangle}$ is then sampled randomly according to the $\hat{y}^{\langle 2 \rangle}$ probabilities, and the sequence repeats with $x^{\langle 3 \rangle}$ and $a^{\langle 2 \rangle}$ producing $\hat{y}^{\langle 3 \rangle}$, which is sampled to become $x^{\langle 4 \rangle}$, and so forth.

At the end of the loop, which terminates whenever the sampled $x^{\langle T_s \rangle}$ represents an end-of-name character, the values $x^{\langle 2 \rangle}, x^{\langle 3 \rangle}, x^{\langle 4 \rangle} ... x^{\langle T_s \rangle}$ are constructed into a sequence of letters constituting the randomly generated name. Although the letters are selected randomly, because they are selected according to the probabilities of their existence following the previous sequence of letters, the resulting name is a plausible drug name.

To avoid generating names of insufficient length, the sampler was modified slightly to prevent generating end-of-name characters for names shorter than six letters in length.

5 Model Training and Hyperparameter Search

Before the LSTM could be used to generate drug names, it needed to be trained to have appropriate values loaded into its weight matrices and bias vectors. Before training, the weight matrices were initialized with random values and the bias vectors were initialized to zero-valued vectors. With randomly initialized weights and biases, the name generator could only generate random gibberish, as shown in Figure 3.

```
Olzxwudmfqoeyhsqwasjkjvu
Kofbaetvgbamwkwgflynfczchrvbbxvnengdqoxmifpqmepfvr
Lzxwuemfqoeyhsqwasjkjvu
Ofbaeuvgbamwkwgflynfczchrvbbxvnengdqoxmifpqmepfvrm
Zxwuenfqoeyhsqwasjkjvu
```

Figure 3: Random Gibberish Generated by Untrained LSTM Name Generator

To create a working model capable of generating plausible drug names, the LSTM's weights and biases were trained using the architecture of Figure 1 using the dataset of existing drug names as both x and y.

The core principle of neural network training is to calculate the error, or loss, between the "ground truth" output y and the network's "predicted" output $\hat{y}$ produced during forward-propagation; to calculate the slope by which each parameter affects the error during back-propagation; and to update the parameters by a small amount in proportion to the error and the slope to descend the gradient and reduce the error. Through repeated iterations of this process, the error is progressively minimized until it reaches a satisfactory level.

The network was trained using stochastic gradient descent enacted upon one drug name at a time. The loss was computed as the sum of the cross-entropy losses of each letter in the sequence between y and $\hat{y}$. To promote stability in gradient descent, the loss was smoothed using a first-order difference equation, $L^{\langle t \rangle} := 0.01L^{\langle t \rangle} + 0.99L^{\langle t-1 \rangle}$, and exploding gradient problems were avoided by clipping gradients to within the range $(-5, 5)$. Further, to prevent instability, a low learning rate was selected.

After LSTM RNN training, absurdly short drug names were generated, preventing the generation of longer and more interesting sequences. Prior experience with the same model generating names that had greater similarity to each other informed the notion that drug names had more diversity, and therefore were more difficult to learn patterns for.

5.1 First Dead End

Observing that the increased pattern diversity was causing an over-representation of end-of-name characters in the model's $\hat{y}$ probabilities, an approach was tried to incentivise the model to produce longer names. In gradient descent, the parameter update rate was artificially increased (by a factor of $\eta = t^{1/4}$) for later characters, effectively increasing the penalty for predictions that were too short.

Although this approach improved the quality of generated names, it opened a Pandora's box of numerical instabilities which would manifest as generation of endless strings of letters. Balancing the early-termination penalty η with learning rate α and number of iterations was delicate and non-obvious, and adjusting other hyperparameters would throw the balance off in unpredictable ways. Furthermore, deviating from the established, analytically satisfying path was unsettling, so this change was reverted.

5.2 Second Dead End

A second approach was tried due to a fundamental misunderstanding of LSTMs. Recognizing that the RNN lacked sufficient memory of previous letters, and hoping to improve its memory, the $x^{\langle t \rangle}$ vector was modified into a vector with a 1 representing the current letter at step t, and a δ value representing the previous letter at step $t - 1$ (where δ was varied between 0.1 and 0.9). Another attempt was made wherein the vector also contained information from the $t - 2$ letter.

It was initially thought that such modifications would promote memory of letters from a more distant history, but it instead seemed to do the opposite—the resulting generated names had long sequences of repeated letters. This phenomenon was not understood, and the approach was quickly abandoned.

5.3 Final Training Approach

Taking a step back, it was realized that the internal state $a^{\langle t \rangle}$ contains not just information from the letter at position $t - 1$ in the sequence, but also residual information from all previous letters, so a sufficiently sophisticated LSTM model should be able to tolerate the pattern diversity of drug names. As a result, it was decided to simply follow the original algorithm and adjust hyperparameters.

The key to storing state is the $a^{\langle t \rangle}$ vector, so more state can be stored by making it larger. It was originally a 50-dimensional vector, which, while sufficient for names of low diversity, proved entirely unsatisfactory for drug names; as its size was increased, the quality of the generated drug names improved monotonically. Eventually, it was decided to make it a $2,000$-dimensional vector, which was as large as the server would successfully execute without changing the code architecture.

Even with a low learning rate of $\alpha = 0.005$, chosen to ensure numerical stability, the loss quickly settled to an asymptote after several thousand iterations as shown in Figure 4. After reaching the asymptote, the generated names did not seem to improve dramatically. As a result, only $10,000$ training iterations were deemed useful.

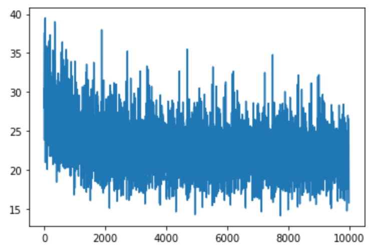


Figure 4: Training Loss

6 Results

After training, the model was used to generate names as illustrated in Figure 5. With the exception of short-name suppression in the name generator, no other strange hacks were required to generate plausible drug names; all that was required was a sophisticated LSTM with a sufficiently large $a^{\langle t \rangle}$ hidden state vector and a sufficiently low learning rate for numerical stability.

Imutro	**Etirase**	**Azalir**	**Clevidine**	**Imadine**	**Isarine**	**Amerit**
Hidaadiq	Iselive	**Robetex**	**Nekidone**	Laaede	Upadat	Pitetagel
Ivotide	Orobime	Yceroc	**Matidine**	Aadena	**Rasine**	Iumide
Idaadiq	Selive	Obetezid	Idelore	**Adiniden**	Ofadogin	**Inocine**
Votide	**Mocile**	**Alisadose**	**Atidine**	Aidehetid	Acasdir	**Umodene**
Edaetoia	**Elivem**	**Anomyl**	Cietina	Eenaltic	Atasratis	**Iramine**
Troepene	**Rasene**	**Estacore**	**Toline**	Lisent	Sarodini	**Tanetiyl**
Aadire	Eptiedu	Irezfe	**Letine**	Amdexic	Inetrite	Ramesiron
Soinete	Arenexe	**Otacore**	**Olline**	Egexid	Astatis	Anetiyl
Adoted	Iumedu	**Rezela**	Etined	Ramide	Netrite	**Amesiron**
Ofelin	Omenot	Racpride	Doopaa	Amtida	Osbist	Hephoso
Anethex	Unedycbir	Hylece	**Ronedalis**	**Cirasine**	Hlunose	Ipoxino
Idinex	**Idazalen**	**Actone**	Ooqaaf	Exidado	Oxeuse	Ortino
Redanocol	**Edxalis**	Yleceido	Inaaede	**Edolene**	Astocinox	Oxinoita

Figure 5: Generated Drug Names. "Plausible" Names Highlighted

Although the generated names are, for the most part, believable as drug names, a substantial fraction is not. By a manual count, per the author's judgment, only about 40% of the generated names are plausible. It is believed that a larger state vector, or a deeper network, could increase this fraction.

7 Conclusion and Future Directions

The result of this project is a LSTM RNN that generates novel drug names for future drugs, and could be used gainfully to streamline the drug name creation process.

A larger $a^{\langle t \rangle}$ vector could result in generating a larger fraction of plausible names, as could a deeper network; these were not explored in this work. That said, it is the author's opinion that such improvements would likely be of marginal practical benefit; recognizing that ultimately the objective is to create unique names that are to some degree dissimilar to existing names, a more complex model could result in the model gaining sufficient sophistication to over-fit the training data, resulting in generated drug names that are too similar to existing drug names.

In practice, drug names are chosen based on their association with words that inspire emotions and ideas which the drug manufacturer hopes to evoke for market appeal.[5] Also, to avoid confusion, drug names must be dissimilar, both in pronunciation and in appearance, to existing drug names. As a result, a more practical pursuit could be to develop a name pruner to eliminate unsatisfactory names and present only those of the highest quality for manual review.

Conceptually, a name pruner could be implemented by first training a LSTM to recognize the degree of similarity between words and word fragments with similar pronunciations and appearances, although of different spellings. Then, the name pruner could be applied to eliminate generated names whose similarity to existing drug names exceeds some threshold. Further, by comparison with words and their synonyms that the manufacturer intends their drug to be associated with, the pruner could select only names whose similarity exceeds some value.

References

[1] Joseph A. DiMasi, Henry G. Grabowski, Ronald W. Hansen *Innovation in the pharmaceutical industry: New estimates of R&D costs*. Journal of Health Economics, https://www.sciencedirect.com/science/article/abs/pii/S0167629616000291, 2016.

[2] Martin A Makary, Michael Daniel *Medical error—the third leading cause of death in the US*. The British Medical Journal, https://www.bmj.com/content/353/bmj.i2139, 2016

[3] Institute of Medicine *Ethical and Scientific Issues in Studying the Safety of Approved Drugs*. National Academies Press, https://www.nap.edu/catalog/13219/ethical-and-scientific-issues-in-studying-the-safety-of-approved-drugs, 2012

[4] NIH *Drug Names*. U.S. National Library of Medicine, https://druginfo.nlm.nih.gov/drugportal/drug/names/a, 2020

[5] Susan Scutti *'Creation engineering': The art and science of naming drugs* CNN, https://www.cnn.com/2016/11/25/health/art-of-drug-naming/index.html, 2016