Bibliography Report

Supervisor: Marianne Clausel
Authors: Wenjun Sun, Guillaume Richez, Adrien Claudel

December 2020

1 Motivation

The purpose of this project is to learn and master different word embedding methods, and compare these methods through experiments. There are many ways to realise word embedding. Word2Vec and GloVe are the two most widely used methods.

Word embedding is an important way to map a word into a vector space. And then computer can take advantages of it to achieve NLP tasks such as word clustering and document classification etc. Word embedding methods can overcome the sparsity shortcomings of one-hot encoding and represent a word more reasonably and efficiently.

Based on the word embedding, many other methods are developed. For example, we can map a document into a vector and so on.

2 Corpora

Corpora is a collection of data that contains the written or audio content. It is the premise of the NLP tasks and the quality of the corpus affects the completion quality of the entire task.

In this project, we only use written corpora and decide to choose the 20News Group corpora.

2.1 Description of the 20Newsgroups

The 20Newsgroups data set is one of the international standard data sets used for text classification, text mining and information retrieval research. The data set collected about 20,000 newsgroup documents, evenly divided into 20 newsgroup collections with different topics.

There are three versions of the 20newsgroups data set. The first version 19997 is the original and unmodified version. The second version bydate is divided into training (60%) and test (40%) data sets in chronological order, and does not contain duplicate documents and newsgroup names (newsgroup, path, membership, date). The third version 18828 does not contain duplicate documents, only the source and subject.

2.2 The advantages of 20Newsgroups

First, 20Newsgroups is a mature and authoritative corpus, which is used by NLP researchers almost all over the world, so the experimental results can be easily checked for correctness. And Sklearn in python supports this corpus very well. The content and tags of the corpus can be retrieved efficiently for various experiments.

Due to the limitation of the computing power of the equipment, we randomly selected 4 news labels for the experiment:'alt.atheism','soc.religion.christian','comp.graphics' and 'sci.med'.

3 Description of the different embeddings

3.1 TDF-IDF

TDF-IDF is a statistical method used to evaluate the importance of a word to a document set or one of the documents in a corpus. The importance of a word increases in proportion to the number of times it appears in the document, but at the same time it decreases in inverse proportion to the frequency of its appearance in the corpus. The main idea of this method is that if a word appears in an article with a high frequency and rarely appears in other articles, it is considered that the word or phrase has good classification ability and is suitable for classification. TDF-IDF is a commonly used weighting technique for information retrieval and text mining.TDF-IDF includes two parts: Term Frequency(TF) and Inverse Document Frequency(IDF).

The Term Frequency is the frequency of a term appears in a document. Let $n_{i,j}$ be the number of times of $term_i$ appears in the $document_j$. The calculation formula of TF is:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{1}$$

And IDF diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely. We assume N is number of documents in the corpora and df_t is the number documents including $term_i$. The IDF is calculated by this method:

$$idf_t = \log \frac{N}{df_t + 1} \tag{2}$$

The factor df_t needs to plus one so that the situation that a document doesn't contain the term can also be calculated.

Finally the TDF-IDF value of the term i in the document d is :

$$tdf - idf_{t,d} = tf_{t,d} \cdot idf_t \tag{3}$$

In text classification tasks, we can take the tf-idf matrix as input.Each column in the matrix is the TF-IDF value of each term and each row represents a document. The structure of the matrix is shown as Table 1:

	$trem_1$	$term_2$	...	$term_n$
doc_1	0.23	1.52	...	2.33
doc_2	1.24	2.49	...	0.39
doc_3	0.47	0.64	...	0

Table 1: tdf-idf Matrix

3.2 GloVe

Currently, there are two main types of learning word vector representation methods: one is based on global matrix factorization, such as LSA. And the other

is a local context window method, such as CBOW and skip-gram proposed by Mikolov in 2013. However, these two methods have their own shortcomings. LSA effectively uses statistical information, but it is very poor in terms of vocabulary analogy, while although CBOW and skip-gram can perform vocabulary analogy very well, because of these two The method is based on a local context window method, it does not effectively use the global lexical co-occurrence statistics.

In order to overcome the shortcomings of global matrix decomposition and local context window, Jeffrey Pennington et al. proposed a new GloVe method in 2014, which is based on the statistical information of global vocabulary co-occurrence to learn word vectors, thereby combining statistical information with local The advantages of the context window method are combined and found that its effect is indeed improved.

The realization of Glove is divided in three steps:

1) Create a Co-occurrence Matrix basing on the corpora

2) Construct an approximate relationship between Word Vector and Co-occurrence Matrix

3) Construct a loss function for this model

For the first step, each element $X_{i,j}$ represents the number of times that word j occurs in the context of word i. Let $X_i = \sum_k X_{i,k}$ be the number of times of any word appears in the context of word i. Then we can make the probability that word j appear in the context of word i be $P_{i,j} = X_{i,j}/X_i$. In the paper, Jeffrey Pennington et al. provided an example to illustrate the relationship between these factors. According to Figure1, we can obtain that

Probability and Ratio	k = *solid*	k = *gas*	k = *water*	k = *fashion*
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Figure 1: Example

the ratio $P(k|ice)/P(k|steam)$ can indicate the relationship between word k, ice and steam: For words k like water or fashion, that are either related to both ice and steam, or to neither, the ratio should be close to one.

Based on above finds, we can start the second step. By training the word vector, the above ratio can be obtained after the word vector is calculated by a certain function:

$$F(w_i, w_j, \tilde{w_k}) = \frac{P_{i,k}}{P_{j,k}} \tag{4}$$

w_i, w_j and $\tilde{w_k}$ are word vectors representing word i, j and k. F is a unknown function. Because all word vectors are all in a same linear vector space, w_i and

w_j can be differentiated, the function above is transformed as:

$$F(w_i - w_j, \tilde{w_k}) = \frac{P_{i,k}}{P_{j,k}} \tag{5}$$

The left side of the equation is a function for two word vectors, and the right side is a scalar. So the inner product of the vector can be introduced in order to reduce the complexity of the function:

$$F((w_i - w_j)^T \tilde{w_k}) = \frac{P_{i,k}}{P_{j,k}} \tag{6}$$

To unify the operations on both sides of the equation, we can use the exponential method. You can finally get:

$$w_i^T \tilde{w_k} + b_i + \tilde{b_k} = \log(X_{ik}) \tag{7}$$

Among them, b_i and $\tilde{b_k}$ are bias terms introduced to maintain the symmetry of the function.At this time, the goal of the model is transformed into learning the representation of the word vector so that the two sides of the above formula are as close as possible. Therefore, the square difference between the two can be calculated as the objective function.

$$J = \sum_{i,k=1}^{V} f(X_{i,k})(w_i^T \tilde{w_k} + b_i + \tilde{b_k} - \log(X_{ik}))^2 \tag{8}$$

$f(X_{i,k})$ is a weight function and f must have the following characteristics:

1) when the number of co-occurrences of word is 0, the corresponding weight must be 0

2) f(x) must be a non-decreasing function, so as to ensure that the greater the number of co-occurrences of the vocabulary, the weight will not decrease

3) For those words that are too frequent, the function must be able to give them a relatively small value so that they will not be overweighted.

Based on the above three points, the author proposed such a weight function:

$$f(x) = \begin{cases} (x/x_{max})^{\alpha}, & x < x_{max} \\ 1, & otherwise \end{cases} \tag{9}$$

The author set x = 100 in the experiment and found that $\alpha = 3$ is better. Glove combines the advantages of the statistical information of global vocabulary co-occurrence and the advantages of the local window context method. It can be said to be a synthesis of two mainstream methods. Since GloVe does not need to calculate the words whose co-occurrence number is 0, it can greatly reduce the amount of calculation and data storage space.

However, GloVe takes the number of word frequency co-occurrences in the corpus as the goal of word vector learning. When the corpus is relatively small, the number of co-occurrences of some words may be less, and the author thinks that a phenomenon that may mislead the direction of word vector training may occur.

3.3 BERT

BERT is a data pre-processing algorithm developed by google in 2018[1]to improve the understanding of user queries on their search engines. This model processes words according to all the other words in the sentence. This method significantly improves the results of the current models for data mining tasks.

- GLUE 80.5% → +7.7%
- MultiNLI 86.7% → +4.6%
- SQuAD v1.1 93.2 → +1.5%
- SQuAD v2.0 83.1 → +5.1%

This additional treatment aims in particular to better balance the importance of linking words which are often neglected although they can completely change the meaning of a sentence.

4 Comparison between these embeddings

4.1 Preprocessing of each document

In the task of text classification and clustering, we use four types of corpora in the 20newsgroup corpus to conduct experiments:'alt.atheism', 'soc.religion.christian', 'comp.graphics' and 'sci.med'. Among them, the training data set includes 2257 articles, and the test set includes 1502 articles. First, use NLTK to convert all words in these articles to lowercase, and then remove punctuation. After this step, the article will be processed into an ordered bag of words. In addition, in experiments, the method of stemming has been tried. But for some words, this method will produce ambiguity. Such as, the word *leaves* in the sentence $Heleavesme$, will be transformed to $leaf$. So, this method has not been adopted yet. In future experiments, part-of-speech recognition will be added to the preprocessing process so that the stem can be extracted correctly.

For tdf-idf, a dictionary based on all articles is created. The key of this dictionary is each word that appears in the training set, and its value is the word frequency of the corresponding word.

For glove, the article was reorganized and split into individual sentences, and each sentence was sequentially stored in a new file. In this new document, each line is a separate sentence. This is done to lay the foundation for the next training glove vector.

The whole preprocessing method is shown below.

[1]Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* _eprint: 1810.04805. 2019.

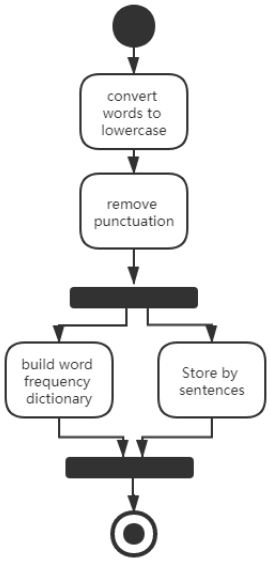

Figure 2: Preprocessing

4.2 Classification tasks

For the tdf-idf method, after feature extraction, the final result is a document feature matrix, and the shape of train matrix is (2257, 35788), the shape of test matrix is (1502, 35788). Each column of the matrix is the tdf-idf value of a word.

For the glove method, First, through the processing of the model, the vector representation of each word will be obtained. Each word will be represented as a 50-dimensional vector. For the final feature representation of the article, we simply adopted this method:

$$F = \frac{\sum V_{word}}{Num_{word}} \tag{10}$$

In the above formula, V_{word} represents a glove vector and Num_{word} represents total number of words contained in the document.

This experiment uses the SVM classifier, and its kermel is set as rbf. The results about classification using tdf-idf and glove are shown as below:

Method	Accuracy
tdf-idf	89.01%
glove	69.84%

Table 2: Results

4.3 Document clustering tasks

K-means algorithm is the most classic partition-based clustering method, and it is one of the ten classic data mining algorithms. The basic idea of the K-means algorithm is: clustering around k points in the space and classifying the objects closest to them. Through an iterative method, the value of each cluster center is updated successively until the best clustering result is obtained.

In the clustering task, we use the kmeans method and set the K value to 4.

For the evaluation of the clustering results, we adopted the Normalized Mutual Information(NMI) method and the Adjusted Rand Index(ARI) method. For NUI, It can show the degree of difference between the experimental result and the standard result. The closer the result is, the closer the NMI value is to 1, and vice the closer to 0. Let $X = 0, 1, 2, 3$ and $YX = 0, 1, 2, 3$ to represent the label set of the experimental result and the standard result respectively. And let $test_result = [1, 0, 2, ..., 3, 2]$ and $true_result = [1, 1, 2, ..., 3, 1]$ to represent experimental results and standard results respectively. Then the way to compute the value of NMI is :

$$NMI(X, Y) = \frac{2 * MI(X, Y)}{H(X) + H(Y)} \tag{11}$$

Among this equation, the method to get the value of $MI(X, Y)$ is:

$$MI(X, Y) = \sum_{i=1}^{|X|} \sum_{j=1}^{|Y|} P(i, j) log(\frac{P(i, j)}{P(i)P(j)}) \tag{12}$$

In the equation, $P(i) = \frac{X_i}{N}$ and $P(i, j) = \frac{|X_i \cap Y_j|}{N}$.

$H(X)$ and $H(X)$ represent the entropy of X and Y:

$$H(X) = -\sum_{i=1}^{|X|} P(i)log(P(i));\ H(Y) = -\sum_{i=1}^{|Y|} P(j)log(P(j)) \tag{13}$$

The ARI method is based on RI. The Rand Index computes a similarity measure between two clusterings by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings. We can compute ARI in this way:

$$ARI = \frac{RI - E(RI)}{max(RI) - E(RI)} \tag{14}$$

After processing, the experimental results are shown in the table below. It

Method	NMI	ARI
tdf-idf	22.24%	14.54%
glove	15.85%	14.64%

Table 3: Clusering Results

is can be observed that the clustering results of these two methods are not very satisfactory. There is one assumption, that it is because we just add all vectors of all words or just compute the average of all vectors. But a document is not just a sum of the words contained in it, it is a sequence of words. While we extracting the feature of the document, we ignore this point. So the ideal feature is not noticed. For promotion, the Recurrent Neural Net(RNN) can be very useful, because it can deal with a sequence of input.

4.4 Conclusion

In TDF-IDF, the weight of words is the first thing taken into account. It is a good way to describe the degree of importance for a word in a certain document. This method has low computational complexity. For a collection of highly specialized articles, this method can achieve good results. But TDF-IDF method ignores the relationship between different words. And because of this drawback, a lot of sematic information will be lost.

Glove integrates the statistical information and the vocabulary information of words. The word-vector obtained after Glove method can show the relationship between words. It is a good way to extract features of words. But in the application in document classification, it is more complex to take advantage of word-vectors than TDF-IDF. Because after TDF-IDF, a document can be represented as a vector with every word has one TDF-IDF value. But in Glove, every word is with a high-dimensional vector. Adding all vector and other linear calculation method are not good way to describe the feature of a whole article. So to extract features of a document, a more complicated method is needed when Glove is used.

5 Neural networks

5.1 RNN

RNN is very effective for data with sequential characteristics. RNN is a kind of neural network with n-to-n structure. It can mine the timing information and semantic information in the data. Using this ability of RNN, deep learning models can be used to solve NLP fields such as speech recognition, language models, machine translation, and timing analysis.The characteristic of RNN is that for each RNN neuron, its parameters are always shared, that is, for a text sequence, any input undergoes the same processing to obtain an output. In the structure of the traditional fully connected neural network, neurons do not affect each other, and there is no direct connection, and neurons are independent of each other. In the RNN structure, the neurons in the hidden layer begin to be connected through a hidden state. The structure of RNN is shown as below:

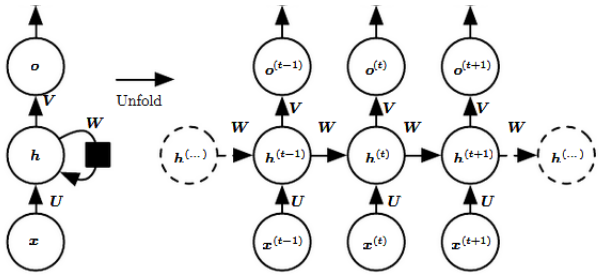

Figure 3: RNN structure

Among the figure, the X^t presents the input at time t and is a vector like $[x_1, x_2, ..., x_n]$. U is a weight matrix from input layer to hidden layer. V is a weight matrix from hidden layer to output layer. And W is a weight matrix from hidden layer of previous time step to the next time step. h^t is the value of hidden layer at time t. O^t is the output at time t.

5.1.1 Forward Pass

Input For every time t, RNN needs an input. In NLP tasks, the input is mainly the word-vector. And in application, we usually set a input window.

Hidden Layer The state of a hidden layer h^t at time step t is based on the current input x^t and the state of previous hidden layer h^{t-1}. The h^t can be presented as:

$$h^t = f(Ux^t + Wh^{t-1} + b) \tag{15}$$

In the equation above, f is a activation function which is a non-linear transformation. The activation f is normally $tanh$ or $ReLU$. And b is a bias vector. U, W and b are the parameters that should be learn by RNN model. RNN units at the same layer share the same parameters, that is to say, data sharing.

Output After getting the state of hidden layer, if the output of time step t is needed, data should be linearly transformed:

$$O^t = Vh_t + c \tag{16}$$

$$y_t = g(O_t) \tag{17}$$

V and c are parameters. g is an activation function, usually $softmax$ function.

$$softmax = \frac{exp(x_j)}{\sigma_{i=1}^{n} exp(x_i)} \tag{18}$$

5.1.2 Backward Pass

The idea of the RNN backpropagation algorithm is to obtain the appropriate RNN model parameters U, W, V, b, c through multiple iterations of the gradient descent method.

For RNN, every time we have a lost function in every position of the sequence, so the final lost is:

$$L = \sum_t L^t \tag{19}$$

The gradients for parameter V and c are calculated as below:

$$\frac{\partial L}{\partial c} = \sum_t \frac{\partial L^t}{\partial c} = \sum_t \widehat{y}^t - y^t \tag{20}$$

$$\frac{\partial L}{\partial V} = \sum_t \frac{\partial L^t}{\partial V} = \sum_t (\widehat{y}^t - y^t)(h^t)^T \tag{21}$$

For the calculation of the gradients for parameter W, U and b, we define the gradient of hidden state at time step t:

$$\delta^t = \frac{\partial L}{\partial h^t} \tag{22}$$

The gradients for parameter W, U and b are calculated as below:

$$\frac{\partial L}{\partial W} = \sum_t diag(1 - (h^t)^2)\delta^t (h^{t-1})^T \tag{23}$$

$$\frac{\partial L}{\partial b} = \sum_t diag(1 - (h^t)^2)\delta^t \tag{24}$$

$$\frac{\partial L}{\partial U} = \sum_t diag(1 - (h^t)^2)\delta^t (x^t)^T \tag{25}$$

6 Perspective and timeline of the send semester

We therefore took the time to familiarise ourselves with the pre-processing and vectorisation stages. During the next few months we will be working on neural networks with a particular focus on LSTMs. We also plan to learn different statistical tools that will allow us to compare corpuses in the future. Our aim is to be able to use all the knowledge acquired with this project and our courses to manipulate real corpus (given by CREM) and to be able to draw information from it.

Overall, in the future, our work will focus on the following:

- Work on a real life corpora provided by CREM
- Encode the texts using embeddings
- Discover another statistical stuff : causality to compare different copora

6.1 Personal notes

Wenjun Sun : My understanding of LSTM is not deep enough. This problem will affect me in my future work. I will learn the principles and application skills of RNN and LSTM as soon as possible. So far, I have reflected on my previous learning, I have paid too much attention to how to apply it, and ignored the principles behind it, especially the mathematical theory.
Guillaume Richez : I think we all three learned a lot about methods we will most likely use in NLP-based careers. I, however, still have shortcomings, most notably on the neural network methods. I should solve — and manage to solve — those shortcomings before the second phase of the supervised project starts.
Adrien Claudel : As far as I'm concerned, the only part that I found difficult was the understand the LSTM, i think that with a little practice I could solve this problem.

References

Devlin, Jacob et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. _eprint: 1810.04805. 2019.

Bird, S., Klein, E., & Loper, E. (2019). Accessing Text Corpora and Lexical Resources.
In Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit (Python 3 / NLTK 3). https://www.nltk.org/book/ch02.html
Licence: CC BY-NC-ND 3.0 US

Brownlee, J. (2016, July 25). Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras. Machine Learning Mastery. https://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/

Brownlee, J. (2017, September 26). Datasets for Natural Language Processing. Machine Learning Mastery. https://machinelearningmastery.com/datasets-natural-language-processing/

Charan, R. (2019, December 15). Understanding Logistic Regression Coefficients. Medium. https://towardsdatascience.com/understanding-logistic-regression-coefficients-7a719ebebd35

Cheng, R. (2020, June 29). Text Preprocessing With NLTK. Medium. https://towardsdatascience.com/nlp-preprocessing-with-nltk-3c04ee00edc0

Chollet, F. (2017, September 29). A ten-minute introduction to sequence-to-sequence learning in Keras. The Keras Blog. https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html

Crawford, C. (2017, July 26). 20 Newsgroups. Kaggle. https://kaggle.com/crawford/20-newsgroups

Davydova, O. (2018, October 15). Text Preprocessing in Python: Steps, Tools, and Examples. Medium. https://medium.com/@datamonsters/text-preprocessing-in-python-steps-tools-and-examples-bf025f872908

Dorsey, B. (2020). Brannondorsey/GloVe-experiments [Python]. https://github.com/brannondorsey/GloVe-experiments (Original work published 2017)

Ho, S. (2017, September 22). Word Mover's Distance as a Linear Programming Problem. Medium. https://medium.com/@stephenhky/word-movers-distance-as-a-linear-programming-problem-6b0c2658592e

Hui, J. (2020, July 1). NLP — Word Embedding & GloVe. Medium. https://jonathan-hui.medium.com/nlp-word-embedding-glove-5e7f523999f6

Karani, D. (2020, September 2). Introduction to Word Embedding and Word2Vec. Medium. https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa

"keitakurita." (2018, April 29).
Paper Dissected: "Glove: Global Vectors for Word Representation" Explained.
Machine Learning Explained. https://mlexplained.com/2018/04/29/paper-dissected-glove-global-vectors-for-word-representation-explained/

Khandelwal, R. (2019, December 28). Word Embeddings for NLP. Medium. https://towardsdatascience.com/word-embeddings-for-nlp-5b72991e01d4

Koehrsen, W. (2018, November 5). Recurrent Neural Networks by Example in Python. Medium. https://towardsdatascience.com/recurrent-neural-networks-by-example-in-python-ffd204f99470

Kusner, M. J., Sun, Y., Kolkin, N. I., & Weinberger, K. Q. (2015). From Word Embeddings to Document Distances.
Proceedings of the 32nd International Conference on Machine Learning, W&CP volume 37, 957–966. http://proceedings.mlr.press/v37/kusnerb15.pdf
event-place: Lille, France

Li, Z. (2019, June 1). A Beginner's Guide to Word Embedding with Gensim Word2Vec Model. Medium. https://towardsdatascience.com/a-beginners-guide-to-word-embedding-with-gensim-word2vec-model-5970fa56cc92

Manimaran. (2019, May 29). Clustering Evaluation strategies.
Medium. https://towardsdatascience.com/clustering-evaluation-strategies-98a4006fcfc

Manning, C. D., Raghavan, P., & Schütze, H. (2009a). Probabilistic information retrieval. In Introduction to Information Retrieval (Online viewing, p. 505). Cambridge University Press. https://nlp.stanford.edu/IR-book/pdf/irbookprint.pdf

Manning, C. D., Raghavan, P., & Schütze, H. (2009b). Scoring, term weighting and thevector space model. In Introduction to Information Retrieval (Online viewing, p. 505). Cambridge University Press. https://nlp.stanford.edu/IR-book/pdf/irbookprint.pdf

Nabi, J. (2019, July 21). Recurrent Neural Networks (RNNs).
Medium. https://towardsdatascience.com/recurrent-neural-networks-rnns-3f06d7653a85

NLTK Project. (2020, April 13). nltk.corpus package—NLTK 3.5 documentation. NLTK.Org. https://www.nltk.org/api/nltk.corpus.html

Olah, C. (2015, August 27). Understanding LSTM Networks. Colah's Blog. https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Phi, M. (2020, June 28). Illustrated Guide to Recurrent Neural Networks. Medium. https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2020). Dive into Deep Learning. https://d2l.ai/d2l-en.pdf
Website: https://d2l.ai/index.html