Imperial College London

Data Science Winter School

Imperial College London

Department of Computing

Natural Language Processing Project Report

Author:
Hongzhen Du
Zeyang Yuan
Ziyun Zhang
Kaiyan Zhu

May 3, 2022

Abstract

Natural Language Processing is a significant part of the uprising artificial intelligence world. This project attempts to tackle the conversion from words to vectors and its representation. Word Embedding models and t-SNE have been utilized to reach the purpose. Multiple figures of word representation have been generated with medium accuracy. This project will possibly help exploring language-related AI problems.

Contents

List of Figures

Chapter 1

Introduction

Natural language processing (NLP) is a part of artificial intelligence that concerned with giving computers the ability to understand text and spoken words in much the same way human beings can [Edu20]. Therefore, NLP contributes to in dealing with large amounts of documents, extracting information and insights as well as categorizing and organizing them. NLP works by taking unstructured data and converting it into a structured data format. Its pipeline contains three stages: text processing, feature extraction and modeling [Kar15].

This project extracts individual words from COVID-19 related articles,converts them into high-dimensional vectors using various language models and achieve word representation utilizing t-SNE.

Chapter 2

Background

The natural language processing has some important steps. Parsing,Tokenization,language models and the complex and multiple ways of dealing with word representations.In this project we choose t-sne to show the result of word representation.

Parsing can turn the original dataset into the corpus which can be recognized by computer more easily. Splitting a text into smaller chunks is a task that is harder than it looks, and there are multiple ways of doing so. Tokenization divides strings into lists of substrings. We have tried three ways to tokenize the string. Depending on the rules people apply for tokenizing a text, a different tokenized output is generated for the same text.Thus we finally decide the best methodology for our project after trying all three ways.

Models that assign probabilities to sequences of words are called language models or LMs. Language modeling not only is the task of determining which words follow which, but also determines what words mean, as individual words are only weakly meaningful, deriving their full value only from their interactions with themselves are formed. In practice, we hope to predict words or simple linguistic components given previous or connected words or components from speech recognition to machine translation. Usually people use associated probabilities of words trained by language models to reach this purpose.

The associated probabilities of words and components are called word vectors. If a dataset of words are taken by the embedding vectors,mathematics can be performed to obtain highly sensible results. Thus, word representation is important for us to study the dataset. In our experiment, we tried two ways to build word representations for each extracted word. Because of the limitation of hardware resource, we used a sample of 1000 tokens to complete this step.

T-Distributed Stochastic Neighbor Embedding (t-SNE) is a technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. In this project we used this technique to convert the trained high-dimensional(256) word vectors into a matrix of pairwise similarities thus we can visualize the final result.Also,we catched and colored Biomedical Entities by using t-SNE.

Chapter 3

Project

3.1 Parse

Dataset was downloaded from Download CORD-19 | Semantic Scholar , 2021-07-26 version. Files in it are all tidied up to dictionary data structure which saved in json type. Therefore, after confirming raw data type, we chose to use *jsonpath* package to extract certain value from it.

Since all text part maintains information, in the initial attempt, we extract all value with key equals 'text', except its super key equaling 'cite_spans' or 'ref_entries' where text value contains almost all numbers and person names. After scanning over all documents, we got a list of string saved in pkl up to 12.5GB.

We reduced the file to 1.7GB considering our limited computational ability by extracting titles for each document only.

3.2 Tokenize and Entity-Recognize

Randomly choosing some part from the parsed string, for example, we set total string list length $num = 1500$ and use `randomint(0,len(str_list)-num-1)` to get the desired slice's beginning index.

`split()`,which is a python built-in method, together with regex method, can be used to construct a tokenizer, with regulations such as whitespace or punctuation or any other rules able to be represented in regex format. However, by simply splitting parts away, words like 'a', 'the' will be included, therefore, we need help from outside dictionaries. Turning to *NLTK* and *scispaCy*, we used their internal dictionaries or copula, successfully excluding punctuation and stopping words. Besides, their stemming or lemmatization method helps transforming each word to its regular format, decreasing the size of vocabulary we generated a lot.

Different from other NLP project, one of the difficulties in this paper lies in biomedical contextual. Luckily, *scispaCy* provides an `ent_` attribute for its Doc datatype. On top of that, we applied another method as well, querying from the outside biomedical dictionary in ontology offering by the NLP project instruction.

BPE has been applied in tokenization part as well. On account of matching tokens to biomedical entities, the pre_trained model GPT2 isn't that suitable to our project, for it uses some certain character to represent whitespace. Instead of transferring to a new pre_trained model, we trained the tokenization model with our data using BPE algorithm from scratch.

3.3 Build Word Representations

3.3.1 n-gram model

Firstly, in order to predict a word by it's previous words, we used n-gram model to build word representation. This language modeling was introduced by Daniel Jurafsky and James H. Martin[JM09]. An n-gram is a sequence of n words. In this project we used trigram model. We have to get the probability of a word ω given two history words $P(\omega|\omega1,\omega2)$. By computing probabilities of entire sequences $P(\omega1,\omega2,\omega)$ using the chain rule of probability, we can get:

$P(\omega 1,\omega 2,\omega) = P(\omega 1)P(\omega 1|\omega 2)P(\omega|\omega 1,\omega 2)$. Next, applying the Markov assumption to the probabilities: $P(\omega_n|\omega_{1:n-1}) = P(\omega_n|\omega_{n-1})$. Then approximating the probability of a word given its entire context as follows: $P(\omega_n|\omega_{1:n-1}) = P(\omega_n|\omega_{n-2:n-1})$. Finally we get the maximum likelihood estimation estimate for the parameters of a trigram model by getting counts from a corpus, and normalizing the counts so that they lie between 0 and 1.

In our trigram model, we set a list of trigrams after getting all the tokens. Then we created a dictionary for each nonredundant word and set the wished dimension of word representation up to 256. In the process of training the corpus, the model turned the words into integer indices and wrapped them in tensors. To get log probabilities over next words, we ran the forward functions defined in the model. And last, we also computed the loss function to see whether the training was accurate enough to predict words.

As mentioned above, our corpus has only 878 words. So the epochs of the training process must be big enough. We set the epochs up to 200 and observe the loss function result in every epoch. Around 200 epoch, the loss tends to be stable as picture below showed. In order to see the effect of the training result, we also output the embeddings into 'json' files with words and their vectors.

Figure 3.1: The loss result of tri-gram model


```
"previo":[
    0.9675319194793701,
    -0.9311603307723999,
    0.02403833530843258,
    -0.3997882604598999,
    0.8438745141029358,
    1.423478126525879,
    -1.708791732788086,
    0.0121830515563488,
    1.511397123336792,
    0.008626925759017467,
```


Figure 3.2: The dict form of word vectors trained by tri-gram

3.3.2 Skip-gram model with negative sampling

Secondly, to predict the nearby words of a given center word, we choose to use Skip-gram model introduced by Tomas Mikolov [MSC+13]. The picture below is the Skip-gram model architecture: the input is one word, and after the projection people will get the output of surrounding words of the given word.

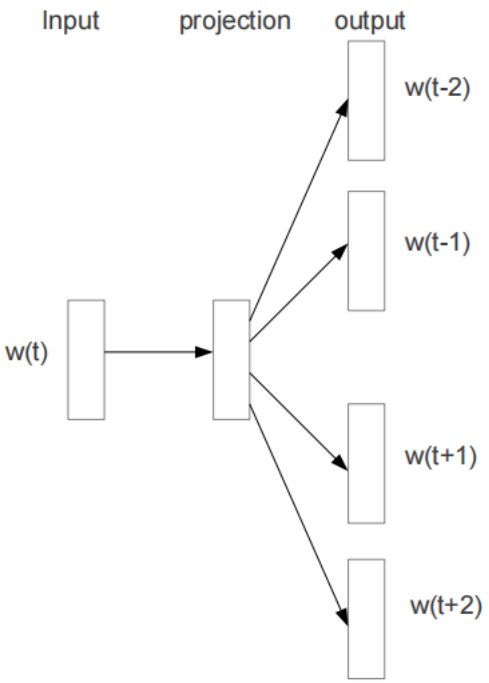

Figure 3.3: The architecture of Skip-gram model

Given a sequence of training words ,the core of the model is to maximize the average log probability (is the size of the training context, is the center word): $\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logp(\omega_{t+j}|\omega_t)$

A good model should be able to differentiate data from noise by means of logistic regression, so NCE(Noise Contrastive Estimation) introduced by Gutmann and Hyvarinen [GH12] is used in Skip-gram model. Thus NCE is simplified into NEG(Negative Sampling) by the objective:

$$log\sigma({v'_{\omega O}}^T v_{\omega I}) + \sum_{i=1}^{k} E_{\omega_i \sim P_n(\omega)}[log\sigma(-{v'_{\omega i}}^T v_{\omega I})]$$

In our Skip-gram model, we encoder each nonredundant word and collect their word frequency. In the process of setting the dataset, we put the first three words and last three words of each word into a group : $y_1, y_2, y_3, x, y_4, y_5, y_6$. As the learning objective is computational inefficient, we apply negative sampling to accelerate the training. Thus we randomly select 12 words excluding y as negative samples $neg[:12]$. Also we choose the dimension of vectors up to 256. In the training process, as y_1 to y_6 are predicted by x , we have to calculate the joint probability of y. It is the same way to deal with neg_1 to neg_{12} . Thus we can get the loss result by add up y and neg losses. Surprisingly, when the epoch is 20 we can find that the loss is tending to be stable. As the picture shows, the loss is around 9. We also get the 'json' file to see vectors of each words trained by Skip-gram model.

We define a test function to see whether the trained Skip-gram model can predict with accuracy. The nearest words' vectors of a center word have the smallest included angles with the center word vector. By printing the predicted words and observing them, we can know the effect of our trained Skip-gram model.

```
第 7 轮： 10.154947280883789
第 8 轮： 9.062763214111328
第 9 轮： 9.216845512390137
第 10 轮： 9.67368221282959
第 11 轮： 9.274986267089844
第 12 轮： 10.326301574707031
第 13 轮： 8.853076934814453
第 14 轮： 10.546208381652832
第 15 轮： 8.746736526489258
第 16 轮： 9.163549423217773
第 17 轮： 9.464311599731445
第 18 轮： 8.855155944824219
第 19 轮： 8.885441780090332
```

Figure 3.4: The loss result of Skip-gram model


```
"ante":[
    [
        -0.10256464034318924,
        0.002733733505010605,
        -0.14166224002838135,
        0.2461010068655014,
        -0.07738242298364639,
        0.32966873049736023,
        0.11262322962284088,
        0.058884602040052414,
        0.26506179571151733,
```


Figure 3.5: The dict form of word vectors trained by Skip-gram

3.4 Explore the Word Representations

Visualization of high-dimensional data is an important problem in many different fields. Now, there are many different ways to visualize high-dimensional data, such as Sammon mapping, Isomap and LLE. In our processing, we choose the t-SNE method (t-Distributed Stochastic Neighbor Embedding) to convert high-dimensional datasets into matrices with pairwise similarities. Originated from SNE, t-SNE models high-dimensional objects by a two-dimensional point that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability [vdMH08].

Stochastic Neighbor Embedding (SNE) first converts high-dimensional Euclidean distances between data points into conditional probabilities representing similarity. Considering the conditional probability is given by the formula:

$$p_{j|i} = \frac{exp(-|x_i - x_j|^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-|x_i - x_k|^2/2\sigma_i^2)}$$

where σ_i is the variance of the Gaussian that is centered on datapoint $_i$, which is aimed at computing pairwise affinities with perplexity. Moreover, in t-SNE, we adopt a t-distribution with one degree of freedom as the heavy-tailed distribution in the low-dimensional mapping. The joint

probability is defined as

$$q_{ij} = \frac{(1 + |y_i - y_j|^2)^{-1}}{\sum_{k \neq l}(1 + |y_k - y_l|^2)^{-1}}$$

, for which would correspondingly low-dimensional affinities. Similarly, the gradient is calculated through the formula

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + |y_i - y_j|^2)^{-1}$$

These formulas contribute to the process of the t-SNE algorithm (Shown in Figure 3.6) [vdMH08].

Data: data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity *Perp*,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity *Perp*
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  for t=1 **to** T **do**
    compute low-dimensional affinities q_{ij}
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
  end
end

3.4.1 Visualize the Word Representation by t-SNE

The dataset we employed is the N-gram tokenization completed in Section 3.3.1, containing 220049 sample tokens. For our experiments, we randomly selected 100 tokens for computational reasons.

We reduce the dimensionality of the data to 8 using PCA, speeding up the computation of pairwise distances between data points. We then converted the 8-dimensional representation into a 2-D plot and displayed the resulting plot in a coordinate system. The category information is only used to select the symbol of the map point.

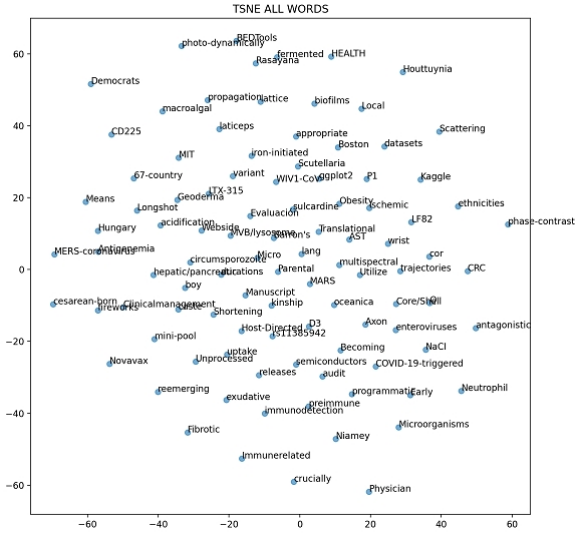

Figure 3.6: Visualization of 100 tokens from N-Gram dataset produced by version of t-SNE

The inspection part shows that much of the local structure of the data is captured. The t-SNE method constructs a map that did not depict obvious clusters but reflect a discrete distribution among the graph.

3.4.2 Visualize the Word Representation of Biomedical Entities by t-SNE

Instead of visualizing the whole dictionary, we are going to visualize the biomedical entities. The sample owl dictionary contains 1800 words, from which we extracted 141 classified as biomedical entities.

For words extracted from the dictionary, we use `rdflib.graph.query()` to acquire the each corresponding URL, from which the words can be labelled by sorting biomedical entities, for instance, CHEBI and PATO. Similarly, we first use PCA to reduce the dimensionality of the data to 8. Then we execute dimensionality reduction technique to display the resulting plot in 2-D form, with 4 kinds of labels regarding the various meaning of words. Since class information is not used

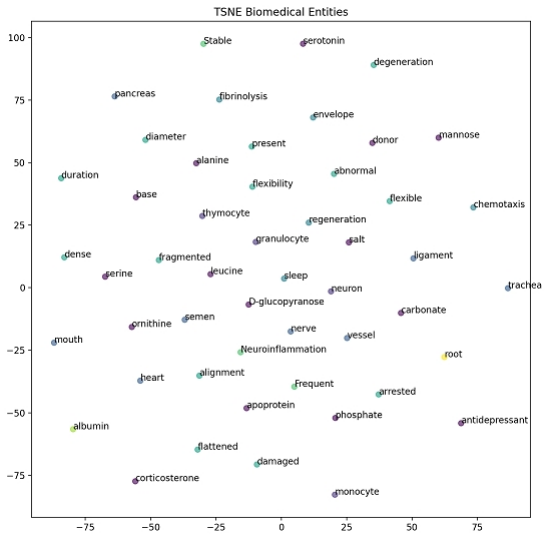

Figure 3.7: Visualization of 50 biomedical entities produced by t-SNE

to determine the spatial coordinates of map points, coloring provides a way to assess how well a map retains similarity within each category. In the scatterplot, 4 different meanings represented by colors discretely displays the 50 biomedical entities.

3.4.3 Co-occurrence

Co-occurrence refers to the probability that two terms (also called coincidences or co-occurrences) in a corpus of text occur side by side in a certain order [Kro05]. However, such "raw" corpus do not have good properties. Because common words appear more frequently, they are more likely to be identified as having a strong correlation with the target vocabulary, even though there are actually less commonly used words that are more relevant.

To solve this problem, we introduce the concept of Pointwise Mutual Information (PMI), which is a measure of association used in information theory and statistics for single events. Denote the co-occurrence matrix as C, the number of co-occurrences of words x and y as C(x, y), and the number of occurrences of words x and y as C(x) and C(y), respectively, with the number of words denoted as N. The formula is shown as [RE09]:

$$\mathrm{PMI}(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)} = \log_2 \frac{\frac{C(x,y)}{N}}{\frac{C(x)}{N}\frac{C(y)}{N}} = \log_2 \frac{C(x,y)\cdot N}{C(x)C(y)}$$

We execute `LabelEncoder` and `Counter` to digitize and uniformly count the previously parsed and split data in JSON file, then continue to calculate PMI for co-occurrence. The output list

consisting the words that co-occurs with *COVID-19* is shown as followed:

```
['vaccine', 'pandemic', 'infection', 'Patients', 'patients', 'Disease', 'cases', 'outbreak', 'Wuhan', 'severe', 'pneumonia', 'epidemic', 'disease', 'Pandemic']
```

3.4.4 Semantic Similarity

Semantic similarity is a metric defined over a set of terms, which distance between items is based on the likeness of their meaning or semantic content [HRJM15]. In this process, We choose the Word2vec method to perform word embedding, generating gensim for topic vector representation for unsupervised learning of textual hidden layers from unstructured text. Cosine similarity is also implemented to analyze the semantic similarity to COVID-19.

```
('circular', 0.22090131044387817),
('accumulation', 0.193635493516922),
('broad', 0.19314929842948914),
('artery', 0.18390682339668274),
('convex', 0.18253907561302185),
('Bacteremia', 0.17330783605575562),
('progressive', 0.16059379279613495),
('title', 0.15936130285263062),
('pressure', 0.15278802812099457),
('mass', 0.14851871132850647)]
```

By calculating the 10 most similar entities with the coronavirus, the result is as followed: The inspection part shows that much of the local structure of the data is captured as well.

3.4.5 Mining Biomedical Knowledge

The aim of biomedical text mining is to study that how it may be applied to texts and literature of the biomedical and molecular biology domains. In this section, we are going to implement relationship discovery so as to identify interactions between subsequent events over time or causal relationships [Bou09].

To analyzed the semantic similarity between biomedical entities, we compute the cosine similarity and set a standard that the pair words would be sematically similar since the cosine value is greater than 0.25.

Moreover, as for co-occurrence, we executed the PMI function again to judge the association between different entities. When the result is positive, defined as PPMI, the entity pair would classified as co-occurrence.

```
In [30]: simi
Out[30]:
[('vein', 'similar', 'hard'),
 ('green', 'similar', 'insulin'),
 ('dense', 'similar', 'serine'),
 ('behavior', 'similar', 'oocyte'),
 ('Occasional', 'similar', 'license'),
 ('cytoskeleton', 'similar', 'pit')]
In [31]: relationship
Out[31]:
[('transport', 'co-occurrence', 'cholesterol'),
 ('chloride', 'co-occurrence', 'ammonium'),
 ('organ', 'co-occurrence', 'multiple'),
 ('mucosa', 'co-occurrence', 'feces'),
 ('lymphocyte', 'co-occurrence', 'thymus'),
 ('heart', 'co-occurrence', 'heart'),
 ('acute', 'co-occurrence', 'protein'),
 ('liver', 'co-occurrence', 'kidney'),
 ('aspartate', 'co-occurrence', 'structure'),
 ('cell', 'co-occurrence', 'nucleated'),
```

Displayed in a sorted list, there are 6 entity pairs of semantic similarity and 927 co-occurred biomedical entities. Constrained by the length of the article, we cannot present all the results here

Chapter 4

Evaluation

In the parsing and modeling part,there are several aspects to be considered for evaluation algorithms and models applied.

1. The way to randomly choose data from the original 12.5GB parsed data.

   Since computational ability, which definitely cannot be be ignored in those optimization projects, is limited due to lack of powerful servers, hence raw data needed selecting.

   The first method is to randomly pick some samples using `random.sample(str_list,num)`. While it sufficiently achieves randomness, the information between paragraphs lost terribly. For instance, it may pick one sentence from the first document, then the second another, yet then the third turn back to the first document, which may cause a huge information loss considering co-occurrence and similarity between words.

   The second method is to randomly select certain slice from the original data, the logic between paragraphs maintained, but obviously, data being limited for it could possibly be the slice of a handful of passages, lacking representation for the whole dataset.

   The third method is to only select title from each document, which may seem a little tricky for titles contain less meaningless words and tend to be more regular. The only shortage here may be extensibility, for in practice, it's always hard to collect such regularized data.

   On the basis of the must to decrease the size of the input data, balancing information loss and practibility, we chose the second method in this paper.

2. The tokenization method. There comes several aspects to be examined.

   1. The output data's regularity.
   2. Vocabulary size and computational cost.
   3. Suitability for afterwards process.

   The first aspects considered the effectiveness of the tokenization algorithm, the second the efficiency, and the third stands for the consistency for the whole project. As mentioned in project part, the simple `split()` combining regex lost *NLTK* and *scispaCy* from the first aspect, and BPE lost for the second and third(Although its quite efficient using pre_trained model, the chosen gpt2's output format isn't suitable here. So it only considers the BPE model trained from scratch with our data). Besides, *scispaCy* provides biomedical NER, though broad, gaining another bonus compared to *NLTK*. So we uses *scispaCy*'s results for later word representation.

3. Biomedical entity recognition. Since the output from *scispaCy*'s `Doc.ent_` attribute seems too broad, we queried from Human Phenotype Ontology dictionary.

In the modeling part, we used two kinds of models to do the word representation. Due to the limitation of computer hardware,we used only a small part of the dataset to train our model. Thus the prediction can't meet the expectation. But we tried the best parameters to control the loss into a stable range. All the words of our corpus are represented into vectors of 256 dimensions.

In the t-SNE part, we emphasized on using t-Distributed Stochastic Neighbor Embedding to perform perform dimensionality reduction and visualization on word representation, particularly for biomedical entities in this case. In addition, we manually selected the co-occured entities that accompanied with COVID-19, as well as generating word embedding that encode the semantic similarities in a vector space.

Chapter 5

Conclusion

Regardless of improvement from computational ability, like training more data, ways we can do to further polishing results from tokenization and biomedical entity recognition can be as follows.

1. Use parsed titles to pretrain BPE model and then apply the pretrained model to the whole text. BPE can effectively prevent OOV(out of vocabulary) and the regularity of the titles boosts biomedical entity recognition.

2. Use *scispaCy*'s `Doc.ent_` together with Human Phenotype Ontology. It may further increase the completeness and accuracy in recognizing biomedical entities and their relationships, like replacing the tokenization results by `Doc.ent_`'s results, which helps focusing totally on the meanings and relationships between biomedical entities.

We used only a small part of the dataset to train our model. Therefore, bias could be shown in the result.In the future experiment, a large amount of words is indispensable in the training process. The ngram can be optimized by setting the n up to 10 even 100, as we only used trigram model in this process. The Skip-gram model can be optimized by choosing a larger batch of words.

Similarly, limited by the degree of mastery of data visualization and word embedding programming techniques, we did not get very clear results, especially in the analysis of co-occurence. For further analysis, there is a must to increase the size of the training set and increase the times of iterations.

Bibliography

[Bou09] Gerlof Bouma. Normalized (pointwise) mutual information in collocation extraction. 2009.

[Edu20] IBM Cloud Education. Natural language processing (nlp). 2020.

[GH12] Michael U. Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(11):307–361, 2012.

[HRJM15] Sébastien Harispe, Sylvie Ranwez, Stefan Janaqi, and Jacky Montmain. Semantic Similarity From Nnatural Language and Ontology Analysis. *Synthesis Lectures on Human Language Technologies*, 8(1):1–254, 2015.

[JM09] Dan Jurafsky and James H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2nd Edition.* Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International, 2009.

[Kar15] Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks. 2015.

[Kro05] Paul R. Kroeger. *Analyzing Grammar: An Introduction.* Cambridge University Press, 2005.

[MSC+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality, 2013.

[RE09] Raul Rodriguez-Esteban. Biomedical text mining and its applications. *PLoS Computational Biology*, 5, 2009.

[vdMH08] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.