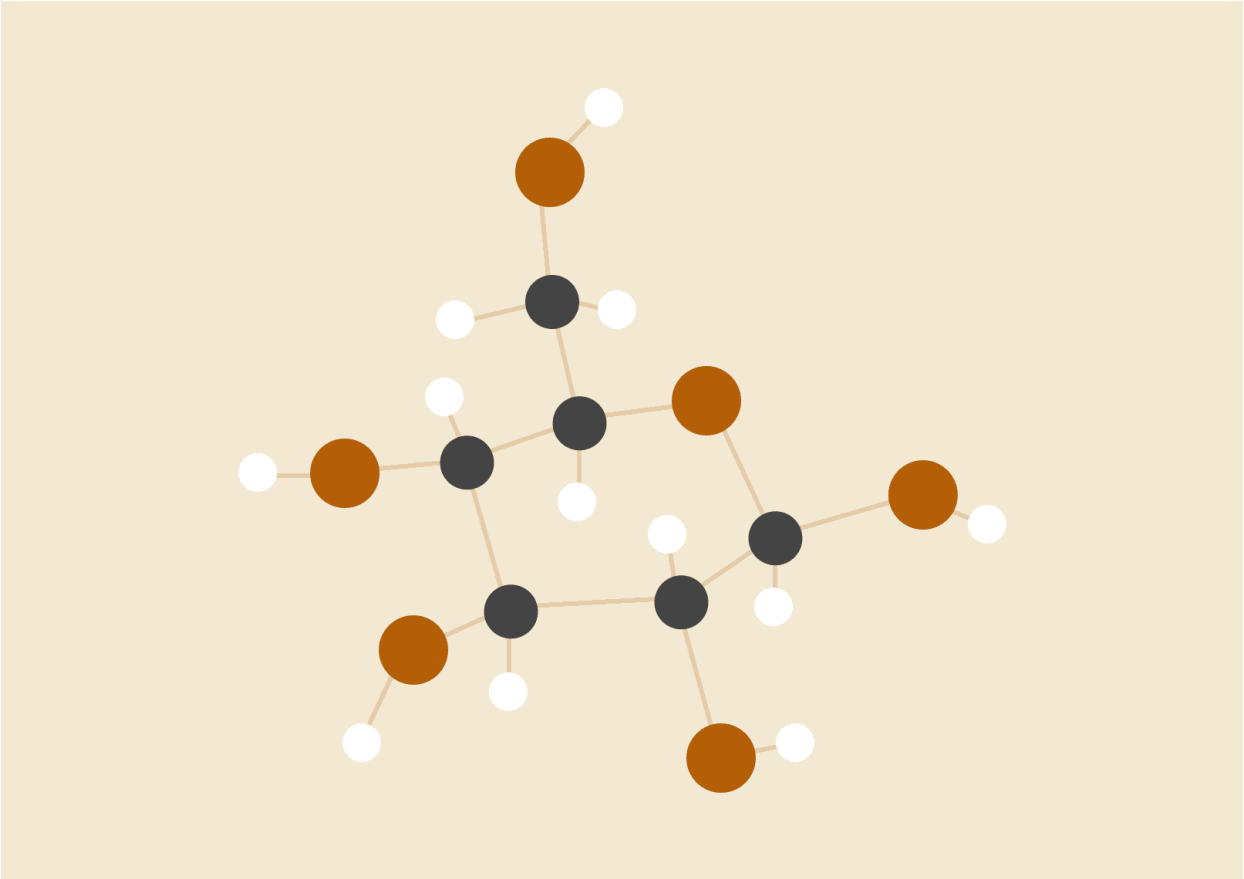
Text Normalization Via Machine Learning Algorithms

IMT 574 Data Science II: Machine Learning and Econometrics



Josh Breiger
Chinar Patel
Yonas Woldekidan

INTRODUCTION

Natural language (NLP) is a part of artificial intelligence that uses machine learning to study a human language. Text normalization is a part of this NLP technique used to process or transform text or characters into machine-readable forms. It is used to transform currency, numbers, symbols, dates, appreciations, etc., to a system readable form. In this paper, we will create a model that best fits and predicts our data.

OUR MACHINE LEARNING PROBLEM

Our data IS a supervised learning problem, specifically a classification problem that predicts an output using our training data. Supervised learning problems contain a different discrete set of values or categorizes. Our dataset has a classification problem called text normalization, which is a labeling problem that occurred in most speech recognition systems. The data needs to convert from spoken form into written form. The process is called inverse text normalization (ITN) (Pusateri, 2017).

Our data contains two columns of tokenized words, ' before' and 'after.'

The 'before' column is before transforming from spoken form to written form (before normalizing the text).

The 'after' column is after we normalized the 'before' column.

The 'class' contains categories for the words and characters listed in the 'before' column.

Examples of spoken-form and written-form

Class	Before	After
LETTERS	IUCN	i c u n
CARDINAL	91	ninety one
DECIMAL	0.161	point one six one
DATE	2006	two thousand six

Our goal is to identify the problem in our given data, transform it to machine-readable form, and create a model that will maximize the accuracy of this text normalization.

MODEL SELECTION

This paper will formulate some questions that need to be answered before we choose our model.

Because it will help us to understand the ML problem clearly.

- Is there any machine learning rule or criteria used to select a model?
- What is the industry practice used to select a model in text normalization?
- How do we choose between a model for a given data?

There is no standard rule used for model selection but, based on the behavior or type of data, the industry used different models for specific data types. There are different characters in models that determine the selection. Some of them are:

- Model Fit - the measure of how well our model performs in the given data.
- Generalization - the ability to adopt new data that is trained in training data.
- Interpretability - how the users easily understand the outcomes.
- Data Size - Some models perform in small-size data, and others are good for large datasets.

After some research and carefully considering the above key factors, we decide to

use the two most common text normalization models.

- Multinomial Naive Bayes
- Linear Support Vector Machine

DATA CLEANING AND DATA PRE-PROCESSING

Preparing and cleaning data is very important before diving into building a model. For our model we first cleaned our data. Our data had 74 null values so we replaced all our null values to space so we don't lose any information. Here we chose to replace it with space to preserve all the data points but one can also drop/remove the null values, replace it with mean/median/mode value, predict a unique value or just use a model that supports missing values. It depends on one's requirement for building the model.

As we are dealing with textual data we had to prepare it first before using it for building our model. We used CountVectorizer, as it converts a collection of text documents to a matrix of token counts. It tokenizes a collection of text documents and builds a vocabulary of known words, but also encodes the new documents using that same vocabulary. Thus, we used a CountVectorizer to convert our text into a matrix of word count vectors on which we implemented ML models like: Naive Bayes and SVM. We did this because the data we were dealing with was in text format and most ML algorithms are based on numerical calculations. So, we embedded our text data into vectors using CountVectorizer.

CODE OVERVIEW

Exploratory Analysis

We began by doing some basic exploratory analysis of the dataset that we were assigned. It had approximately 1 million rows and 5 columns.

After taking a look at class, we saw some clear patterns. Over 90% of the data was in one of two classes, "punctuation" or "plain". This uneven distribution of the data was important to consider when selecting a model.

We also compared the "before" column with the "after" class. We saw that 93% of the time, these columns were equal. This means that no manipulation would be

needed in order to get the after column in spoken form. However, there was still the challenge of trying to determine which of these records actually needed a change.

Countvectorizer

Our input data was in text format and to use ML algorithms, which are mostly based on numerical calculations we parsed our textual data to remove certain words. This process is called tokenization. Once we tokenized our data we encoded them as integers, or floating-point values, for use as inputs in machine learning algorithms and this process is called feature extraction/ vectorizing or embedding.

We used CountVectorizer for this. Scikit-learn's CountVectorizer, converts a collection of text documents to a vector of token counts. It also enables the pre-processing of text data prior to generating the vector representation. CountVectorizer is a highly flexible feature representation module for text data.

Predicting Class

After we used the count vectorizer, we used train-test-split on our training data and predicted the class for our before column using SVM and Naive Bayes. We got 71.9% avg accuracy for Naive Bayes and 90.5% avg accuracy for SVM.

Predicting "After Column"

We then transformed the data based on our model's class prediction using regex and other libraries.

We wrote a long if-else statement in our code depending on the class that the model predicted. Here is some pseudo code of how it worked:

If predicted class is equal to cardinal:

> Use the inflector package to turn the text into the spoken for. Use Regex to remove any weird characters
>
> Return the output as the "predicted after" column

If predicted class is equal to letters:

If there are more than 2 capital letters

Make them lower case. Space them out. Use regex

Return the output as the “predicted after” column

If X class

Do Y

Return output as predicted after column

Else:

Return as the before column

So say our model predicted a specific record to be in the cardinal class. We would then run the if else statement and get an output of the predicted after class.

We did one specific transformation for each different classe.

The output would be the predicted “after” column.

Final Accuracy

To get the final accuracy, we compared the “before” column to the “after” column. If they were equal, we would consider that as correct. So our accuracy was calculated as the total amount of correct labels divided by the total number of rows.

PRACTICAL EXPLANATION

Text normalization is a process by which text is transformed to a more uniform sequence. For example, Product, product and products become product; naïve becomes naive; etc.

When we work with natural language, we are dealing with text data that is closer to human interpretation of any language thus, it has randomness. And machines cannot understand these randomness, it's a challenge for them. So, by normalizing the text data we reduce the randomness and introduce standards which convert our data into a standard form understandable by machines which

helps improve efficiency of our machine.

Normalization brings the data with randomness closer to ‘normal distribution’. As by normalizing we change a natural language input into a standard expected behaviour/shape. Thus, it helps us reduce the variations and dimensionality of our data, increasing the overall performance of our model.

Many automatic speech recognition applications use text normalization to convert written expressions into appropriate "spoken" forms. However, machines often struggle to interpret things like dates and slang that is not easily understood.

These machines need some help to convert these written texts into spoken form. For example, machines need help to convert 12:47 to "twelve forty-seven" and $3.16 into "three dollars, sixteen cents."

REFERENCES

Chin, S.-C., DeCock, R., Street, W.N., & Eichmann, D. (n.d.). Query-based Text Normalization Selection Models for Enhanced Retrieval Accuracy. https://dollar.biz.uiowa.edu/~street/research/hlt10final.pdf

Duque, T. (2020, April 02). *Text Normalization*. Towards Data Science. https://towardsdatascience.com/text-normalization-7ecc8e084e31

Idoko, C. (2019, October). *RandomForest Classifier Vs Multinomial Naive Bayes for a multi-output Natural Language classification problem*. Analytic Vidyah. https://medium.com/analytics-vidhya/randomforest-classifier-vs-multinomial-naive-bayes-for-a-multi-output-natural-language-2426381a5217

Li, S. (2018, September 24). *Multi-Class Text Classification Model Comparison and Selection*. Toward Data Scince. https://towardsdatascience.com/multi-class-text-classification-model-comparison-and-selection-5eb066197568

Murugesan (MRK). (2018). *Exploring the data - Basic analysis*. https://www.kaggle.com/. https://www.kaggle.com/murugesan/exploring-the-data-basic-analysis

Phillips, I. (2012, April 17). *What Is Text Normalization?* http://ianp.org/. http://ianp.org/2012/04/17/what-is-text-normalization/

Pusateri, E., Ambati, B. R., Brooks, E., Platek2, O., McAllaster, D., & Nagesha, V. (2017, August 20-24). *A Mostly Data-driven Approach to Inverse Text Normalization*. https://www.researchgate.net/publication/319185143_A_Mostly_Data-Driven_Approach_to_Inverse_Text_Normalization#:~:text=Introduction,and%20processing%20by%20downstream%20components.

Text Normalization. (2020, December 21). DEVOPEDIA for developers. by developers. https://devopedia.org/text-normalization

understanding-normalization. (n.d.). https://packtpub.com. https://subscription.packtpub.com/book/application_development/9781784391799/2/ch02lvl1sec20/understanding-normalization