VisNLP: A Visual-based Educational Support Platform for Learning Statistical NLP Analytics

Amorn Chokchaisiripakdee and Chun-Kit Ngan
Data Science Program, Worcester Polytechnic Institute, 100 Institute Rd., Worcester, U.S.A.

Keywords: Statistical-based NLP Learning, Educational Support, Data Visualization.

Abstract: We develop and implement a web-based, interactive visual NLP learning platform that enables novice learners to study the core processing components of statistical NLP analytics in sequence. More specifically, the contributions of this work are three-fold: (1) the ease of learners to access and use our platform through any web browser at no cost; (2) the interactive and dynamic visuals (e.g., mouseover events, collapsible tree diagrams, and animations) that enhance the study environment and learners' engagement; and (3) the in-focus step-by-step process, using the job posting classification as an example, to demonstrate the core processing components of statistical NLP approaches.

1 INTRODUCTION

In the Big Data era, there is a large amount of text information available on the Internet such as social media, web advertisements, and online documents. To manage and process text information, Natural Language Processing (NLP) is the critical AI technology that can understand, analyze, and generate text without requiring a human lift a finger. By using text analytics and mining, NLP uses computational and statistical methods to give insight into observed human language phenomena and make computers perform various tasks with human languages. For example, NLP text classifiers, e.g., MonkeyLearn (Text Analysis, 2020), can automatically analyze text and then assign a set of pre-defined tags or categories based on its content. NLP spell-checker applications, e.g., Grammarly app (Grammarly., 2020), are able to identify and correct any spelling mistakes in a text. NLP machine translation technologies, e.g., Google Translate (Google Translate, 2020), allow automatic translation from one language to another without any human intervention. However, to master NLP processes and techniques to develop such kinds of applications is not a trivial task even at an introductory level, as it requires learners, particularly to novice professionals, junior data scientists, and college students, to have a good understanding of linguistic and computation that NLP is based on.

Presently, there are many studying approaches that can assist novice learners in mastering the concepts of NLP. First, the simplest and most economic approach is to self-study and self-read NLP-related text and electronic books (Deng & Liu, 2018; Rodrigues & Teixeira, 2015; Shaalan et al., 2017). This approach requires the learners to (1) read and study a text that consists of several hundred pages and (2) possess a strong self-learning capability. Not only is it time consuming but also noninteractive that results in deterring learners from investigating in this area. The second approach to learn NLP is to complete some on-campus or online programs and courses at educational institutions (Data Science, 2020a; Data Science, 2020b; Statistics and Data Science MicroMasters, 2020). Based upon well-structured and well-organized program curricula and course syllabuses, learners are able to study NLP step by step and piece by piece with the guidance and support of highly experienced instructors. In addition to that, under this active learning and interactive environment with other students, learners can study NLP faster and more effectively. However, this approach is very expensive that requires learners to pay a high tuition cost just for studying one subject area of NLP.

To bridge the above research gap, Zobia and Stefania (Rehman & Kifor, 2015) designed and developed an ontology-based educational Protégé tool to demonstrate the NLP ontology to help learners study NLP. Specifically, this ontology tool presents a tree diagram of conceptual class nodes and subclass nodes of NLP. For example, NLP is the root class node which branches out to many subclass nodes such as

Chokchaisiripakdee, A. and Ngan, C.
VisNLP: A Visual-based Educational Support Platform for Learning Statistical NLP Analytics.
DOI: 10.5220/0010318202240232
In *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2021) - Volume 3: IVAPP*, pages 224-232
ISBN: 978-989-758-488-6

POS Tagging, Sentiment Analysis, Spell Checking, and Text Classification. These subclass nodes also have more subclass child nodes. For instance, there are four subclass child nodes of Text Classification, including Logistic Regression, Naïve Bayes, Support Vector Machine, and MaxEnt, which are all the common text classifiers. This ontology-based approach combines the strengths of both previous two approaches (i.e., inexpensive and interactive), in which learners can download the Protégé software (Protege, 2020) for free and learn NLP with the interactive tool respectively. However, the NLP scope covered by this tool is very broad that includes 16 study areas of NLP and only illustrates the main concept and its sub-concepts without providing any in-depth explanations and descriptions that definitely could not assist new learners in understanding any one of those topics. More importantly, most of the concepts displayed by the ontology are the NLP applications but not the core processing components of NLP, including text pre-processing, token building, and text vectorization.

Apparently, all the aforementioned approaches lack the three important educational elements, i.e., inexpensive, interactive, and in-focus, to help amateur learners master NLP. To mitigate the shortcomings of the existing approaches, we propose the development and implementation of a visual-based educational support platform for learning NLP Analytics (VisNLP). Currently, there are two broad approaches for the NLP analytical processes: (1) statistical-based (Bengfort et al., 2018; Hastie et al., 2020; Lane et al., 2019) and (2) neural network-based (Goldberg, 2017; Kamath et al., 2020; Reese & Bhatia, 2018). The former approach is to perform statistical techniques to process and analyze text data. The latter approach is to use deep neural networks to conduct text mining and analytics. In this paper, we mainly focus on statistical-based methods using One-Hot Encoding, Term Frequency–Inverse Document Frequency (TF-IDF), and Word Probability approaches. Specifically, we develop and implement a web-based, interactive visual NLP learning platform that enables learners to study the core processing components of statistical NLP analytics in sequence: (1) Text Preprocessing (e.g., splitting sentences, spelling check, lowering cases, converting numbers, and removing punctuations); (2) Token Building (i.e., Bag of Words and N-grams tokenization); (3) Text Vectorization (i.e., One-Hot Encoding, TF-IDF and Word Probability); and (4) Text Similarity Dashboard (i.e., Heatmap Tables, Cosine Similarity Matrix, and Euclidean Distance Measurement). Using a variety of interactive visual diagrams with a practical example, novice learners can have a good grasp of the NLP process.

The remainder of the paper is organized as follows. First, we describe our VisNLP framework to show the process of statistical NLP analytics in Section 2. In Section 3, we illustrate our implemented web platform and use the classification of job position advertisements as a pilot example to demonstrate how novice learners can utilize our interactive platform to understand and study statistical NLP analytics step by step and piece by piece. In Section 4, we conclude and briefly outline our future work.

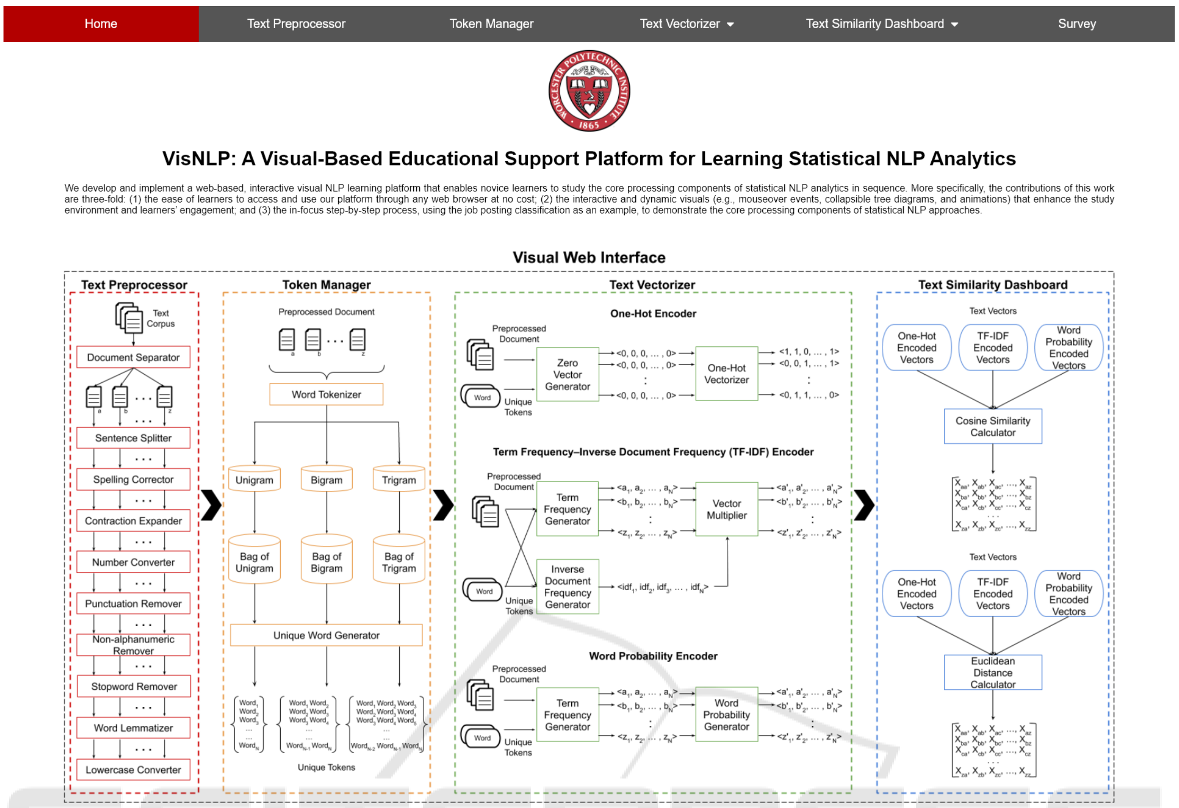
2 VisNLP FRAMEWORK

Fig. 1, home page shows the high-level framework of our VisNLP that consists of five main modules: Text Preprocessor, Token Manager, Text Vectorizer, Text Similarity Dashboard, and Visual Web Interface. Each module has sub-components that manipulate and process texts.

2.1 Text Preprocessor

Text Preprocessor is composed of ten sub-modules that includes Document Separator (DS), Sentence Splitter (SS), Spelling Corrector (SC), Contraction Expander (CE), Number Converter (NC), Punctuation Remover (PR), Non-alphanumeric Remover (NR), Stopword Remover (SR), Word Lemmatizer (WL) and Lowercase Converter (LC). First, the Text Preprocessor takes a text corpus, i.e., a collection of document files from a to z stored in a database of the platform, as an input and the DS module segregates them from the corpus into many individual documents. Each document is then sent to the SS module, which splits the document into individual sentences. The SC module conducts the spell check on each sentence and uses the tokenization approach to replace misspelled words with the highest-probability corrected words. The corrected-word sentences are then sent to the CE module, which expands the contracted form of the words into a longer form, such as ”I'm” to ”I am”, ”You're” to ”You are”, ”It's” to ”It is”, ”S/He isn't” to ”S/He is not”, ”They aren't” to ”They are not” and ”We aren't” to ”We are not”, in each sentence. Then, the NC module substitutes the numeric value for words, such as ”5” to ”Five”, ”11” to ”Eleven” and ”3” to ”Three”.

The subsequent modules, PR, NR, and SR, respectively removes punctuations (e.g., full stop(.), comma (,), and colon (:)), non-alphanumeric characters (e.g., #GoLangCode123!$! to GoLangCode123), and stopwords (e.g., ”ourselves”, ”hers”, ”between”, ”yourself”, ”but”, and ”again”) from expanded sentences.



VisNLP: A Visual-Based Educational Support Platform for Learning Statistical NLP Analytics

We develop and implement a web-based, interactive visual NLP learning platform that enables novice learners to study the core processing components of statistical NLP analytics in sequence. More specifically, the contributions of this work are three-fold: (1) the ease of learners to access and use our platform through any web browser at no cost; (2) the interactive and dynamic visuals (e.g., mouseover events, collapsible tree diagrams, and animations) that enhance the study environment and learners' engagement; and (3) the in-focus step-by-step process, using the job posting classification as an example, to demonstrate the core processing components of statistical NLP approaches.

Figure 1: VisNLP Home Page and Framework.

To shorten the length of each expanded sentence, the WL module groups several forms of the same word together even if their spelling is quite different. For example, "walk", "walked", "walks", and "walking" would be treated the same as "walk". This extensive normalization down to the semantic root of a word is called lemmatization. Finally, the LC module converts each character of a word into a lowercase on lemmatized sentences and then sends the processed sentences to the Token Manager.

2.2 Token Manager

Token Manager is formed by five sub-modules that include Word Tokenizer (WT), three Bag-of-Word (BoW) DBs, and Unique Word Generator (UWG). First, the sentences generated from Text Preprocessor are passed into the WT module that splits each sentence of a document into a collection of vocabularies called tokens (i.e., unigram, bigram, and trigram) that are stored in respective BoW DBs. Given the sentence "I like apple.", the Unigram tokenization generates "I", "like", and "apple" tokens. The Bigram tokenization generates "I like" and "like apple" tokens. The Trigram tokenization generates a "I like apple" token. It is called BoW, as the positional order and the structure of those tokens are discarded but only their occurrence in the document is annotated. The UWG module then removes all the duplicated tokens to generate only the unique tokens (i.e., $Token_1$, $Token_2$, $Token_3$, ..., $Token_N$) among all the documents from the BoW DB.

2.3 Text Vectorizer

Text Vectorizer is divided into three sub-modules: One-Hot Encoder, TF-IDF Encoder and Word Probability Encoder. All three encoders take the two inputs including the unique tokens among all the preprocessed documents and the documents themselves in the text corpus. The unique tokens among all the documents define the dimension of each document vector. For example, if there are N unique tokens in a text corpus, each document vector is a $1 \times N$ one-dimensional vector. The tokens with their occurrence in each document are the components to generate the corresponding vector. Since some corpus tokens do not appear in all the documents, the value of each token may vary according to different encoding schemes.

The One-Hot Encoder is composed of two components: Zero Vector Generator (ZVG) and One-Hot Vectorizer (OHV). The ZVG module generates α zero binary vectors $\langle 0,0,...,0 \rangle$, where α is the total number of documents in the text corpus and the size of each binary vector is N. And then, the OHV module generates α non-zero binary vectors (e.g., $\langle 1,1,0,...,1 \rangle$, $\langle 0,0,1,...,1 \rangle$,..., $\langle 0,1,1,...,0 \rangle$) (Goldberg, 2017, p. 212) for all the documents when a document contains the unique token in a specific position, i.e., a one (1) means on or hot and a zero (0) means off or absent. This collection of vectors is called One-Hot Encoded Vectors.

The TF-IDF Encoder consists of three submodules: Term Frequency Generator (TFG), Inverse Document Frequency Generator (IDFG), and Vector Multiplier (VM). The TFG module generates α token frequency (TF) vectors (e.g., $\langle a_1,a_2,...,a_N \rangle$, $\langle b_1,b_2,...,b_N \rangle$,...,$\langle z_1,z_2,...,z_N \rangle$), where $a_i,b_i,...,z_i$ are the non-negative numeric values β_i, which counts the frequency of a token appeared in a document, for $0 \leq i \leq N$. The IDFG module generates one non-negative numeric (IDF) vector $\langle idf_1,idf_2,...,idf_N \rangle$, where $idf_i = log\frac{N+1}{D_i+1}+1$ and D_i is the total number of documents containing a token, for $D_i \leq N$. The VM module takes the two vectors, i.e., one TF vector and the IDF vector, as inputs and computes the multiplication $\beta_i \times idf_i$ for each token to generate a TF-IDF vector (e.g., $\langle a'_1,a'_2,...,a'_N \rangle$, $\langle b'_1,b'_2,...,b'_N \rangle$,..., $\langle z'_1,z'_2,...,z'_N \rangle$) for a document.

The Word Probability Encoder contains Term Frequency Generator (TFG) and Word Probability Generator (WPG). Similar to the TF-IDF encoder, the TFG module generates the α token frequency vectors. Then, the WPG module takes the TF vectors as the input and computes the probability token $p(t_j)$ of each N-gram document vector by $\frac{f(t_j)}{\sum_{j=1}^{D} f(t_j)}$, where $f(t_j)$ is the frequency of term t in document j and D is the total number of documents.

2.4 Text Similarity Dashboard

After the process of Text Vectorizer, One-Hot, TF-IDF, and Word Probability encoded vectors are the inputs of Text Similarity Dashboard, which computes and generates four types of heatmap tables, including (1) ”Token vs. Document”, (2) ”Text Vectorizer vs. Document”, (3) Cosine Similarity Score (CSS) and (4) Euclidean Distance Score (EDS) (Pattnaik, 2019; Liu, 2019). The Cosine Similarity Score (CSS) is calculated for any two document vectors $(\vec{DV}_i, \vec{DV}_j)$, where $CSS = \frac{\vec{DV}_i \cdot \vec{DV}_j}{\|\vec{DV}_i\| \|\vec{DV}_j\|}$, for $\|\vec{DV}_i\|$ and $\|\vec{DV}_j\|$ are the length of vectors. The Euclidean Distance Score is computed for any two document vectors $(\vec{U}, \vec{V})$, where $EDS = \sqrt{\sum_{i=1}^{n}(u_i - v_i)^2}$, $\vec{U} = \langle u_1,u_2,...u_n \rangle$ and $\vec{V} = \langle v_1,v_2,...v_n \rangle$, for u_i is the i-th token value of $\vec{U}$ and v_j is the j-th token value of $\vec{V}$.

2.5 Visual Web Interface

Finally, to provide the educational support to novice learners to understand the statistical NLP analytics, we develop and implement a highly interactive visual web interface that enables our learners to study the entire process through the step-by-step guidance from our interactive visual diagrams that are described in Section 3.

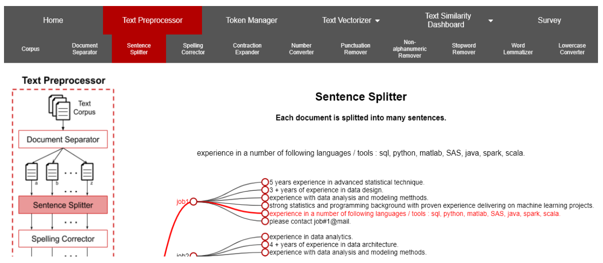
3 VisNLP PLATFORM

In this section, we use the job posting advertisements, which are collected from Monster (Monster, 2017), as an example to describe and explain how learners can use our VisNLP platform with the well-developed interactive D3.js visuals (Bostock, 2020) to learn and study the statistical NLP analytics. Fig. 1 shows the Home page of our platform.

After learners use a web browser to access our VisNLP platform, they see the five main module tabs, including Text Preprocessor, Token Manager, Text Vectorizer, Text Similarity Dashboard and Survey, at the top of the page, the summary contributions of our VisNLP and the framework at the bottom of the page. Note that the ”Survey” tab shown on the platform is only for learners to give us the feedback after using the platform that is not part of our paper scope. Due to the large number of visual diagrams developed in our platform, we selectively choose some representative diagrams in each module and make a short video (https://youtu.be/ZbJuajHgAYQ) to introduce our platform. To begin with our VisNLP, learners can start selecting the Text Preprocessor tab.

3.1 Text Preprocessor

The Text Preprocessor (TP) interface consists of the ”Corpus” and other ten sub-module tabs that we describe in Section II. Each tab contains both the TP module and the tree diagrams to show how each highlighted sub-module impacts on the text. Specifically, the process starts with showing one node of the tree diagram named ”Corpus” to learners, as the ”Corpus” tab is selected by default at the top bar of the page. The TP module also highlights the ”Text Corpus”. When learners move the cursor over the ”Corpus” node, the node turns the color to be red. The corpus description, at the same time, is displayed to explain the meaning of a corpus. As our text corpus is



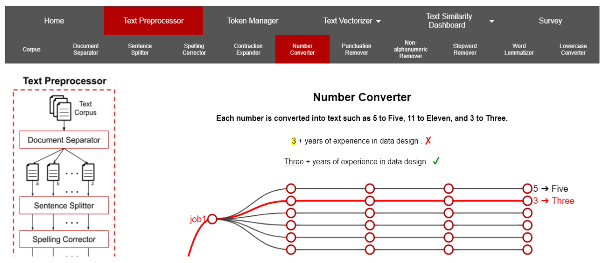
Sentence Splitter

Each document is splitted into many sentences.

experience in a number of following languages / tools : sql, python, matlab, SAS, java, spark, scala.

5 years experience in advanced statistical technique.
3 + years of experience in data design.
experience with data analysis and modeling metthods.
strong statistics and programming background with proven experience delivering on machine learning projects.
experience in a number of following languages / tools : sql, python, matlab, SAS, java, spark, scala.
please contact job#1@mail.
experience in data analytics.
4 + years of experience in data architecture.
experience with data analysis and modeling methods.

(a) Sentence Splitter



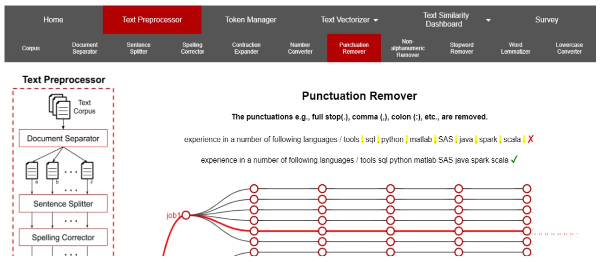
Number Converter

Each number is converted into text such as 5 to Five, 11 to Eleven, and 3 to Three.

3 + years of experience in data design . ✗

Three + years of experience in data design . ✓

(b) Number Converter



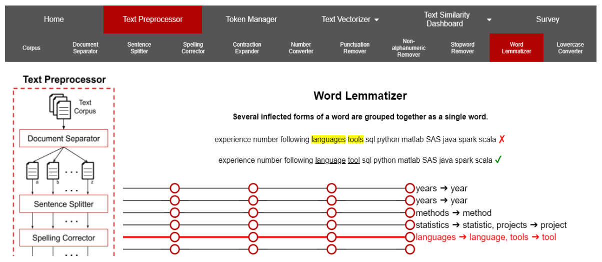
Punctuation Remover

The punctuations e.g., full stop(.), comma (,), colon (:), etc., are removed.

experience in a number of following languages / tools : sql , python , matlab , SAS , java , spark , scala . ✗

experience in a number of following languages / tools sql python matlab SAS java spark scala ✓

(c) Punctuation Remover



Word Lemmatizer

Several inflected forms of a word are grouped together as a single word.

experience number following languages tools sql python matlab SAS java spark scala ✗

experience number following language tool sql python matlab SAS java spark scala ✓

(d) Word Lemmatizer

Figure 2: Sub-modules in Text Preprocessor.

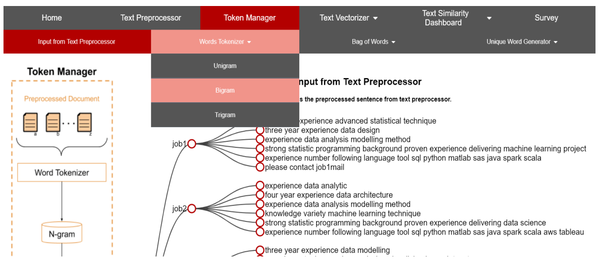
related to job posting advertisements, learners should see the ”Corpus is a collection of text documents. In our example, there are eight job posting documents.” on top of the page.

After the DS module tab is clicked, the TP module highlights the ”Document Separator” and the tree diagram shows up and branches out to eight leaf nodes. Each leaf node represents a text of job posting advertisements in the corpus and is composed of many sentences. In Fig. 2a, the SS sub-module branches out the tree from each job posting node to many sentence leaf nodes that are depended on the number of sentences in each job posting advertisement. For instance, the job posting node No. 1 is split into six sentences. When learners move the cursor over a sentence node, the node and its parent turn the color to be red. The link associated with that sentence node is also changed the color to be red and show the path up to the corpus node. The corresponding sentence is also displayed at the top of the page.

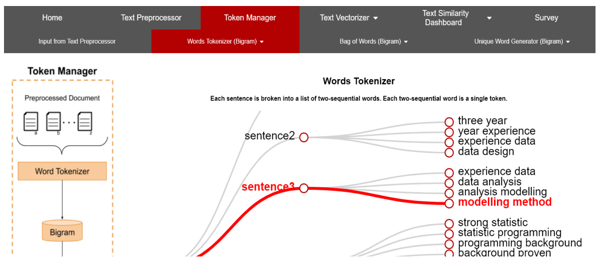
The SC module then takes every single word in a sentence as an input and replaces any misspelled word with the highest-probability corrected word. For example, the misspelled word ”metthods” is replaced by the corrected word ”methods”, i.e., ”metthods $\rightarrow$ methods”, shown at the end of the node. Similarly, the selected node shows the red link path up to its parent and ancestor. By moving the cursor over the node, learners can also see the correction on top of the page, i.e., the ”before” sentence with a red cross mark and the ”after” sentence with a green check mark. The misspelled word is highlighted and the corrected word is underlined. For a sentence that does not contain any misspelled words, it is shown with a green check mark only. In the CE module, at the end of the leaf node, learners can see some words that are expanded. For instance, ”You’re” is highlighted and changed to ”You are” with the underline. In the NC module, a standalone numeric value is replaced by its English word. For instance, in Fig 2b, ”3” is replaced by ”Three” and ”5” is replaced by ”Five”. Both before and after sentences are displayed in the same format as that of the SC module on top of the page.

The next three modules are PR, NR, and SR that take a sentence from the NC module and remove some characters, symbols, and words from the sentence. The PR module removes a punctuation in the sentence. Learners can see all the removed punctuations, such as ”full stop (.)”, ”hyphen (-)”, ”exclamation mark (!)”, and ”question mark (?)”, highlighted at the end of the node. For example, in Fig 2c, the PR module takes a sentence ”experience in a number of following languages/tools: sql, python, matlab, SAS, java, spark, scala.” and then removes ”comma (,)”, “colon (:)”, and ”full-stop (.)” to generate the output sentence ”experience in a number of following languages / tools sql python matlab SAS java spark scala”.

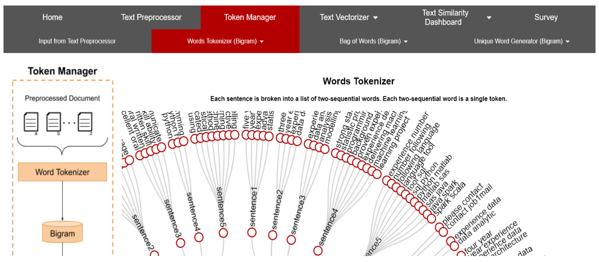
Likewise, the NR and SR modules remove “/¡¿@#$%&*” and stopwords (e.g., a, an, the, is, am, and are) respectively in any sentence. Any characters, symbols, numbers, and words that get removed are highlighted in the original sentence. For example, the NR module takes the sentence ”please con-


Home
Text Preprocessor
Token Manager
Text Vectorizer
Text Similarity Dashboard
Survey
Input from Text Preprocessor
Words Tokenizer
Bag of Words
Unique Word Generator
Unigram
Bigram
Trigram
Token Manager
Preprocessed Document
Word Tokenizer
N-gram
Input from Text Preprocessor
the preprocessed sentence from text preprocessor.
job1
job2
experience advanced statistical technique
three year experience data design
experience data analysis modelling method
strong statistic programming background proven experience delivering machine learning project
experience number following language tool sql python matlab sas java spark scala
please contact job1mail
experience data analytic
four year experience data architecture
experience data analysis modelling method
knowledge variety machine learning technique
strong statistic programming background proven experience delivering data science
experience number following language tool sql python matlab sas java spark scala aws tableau
three year experience data modelling

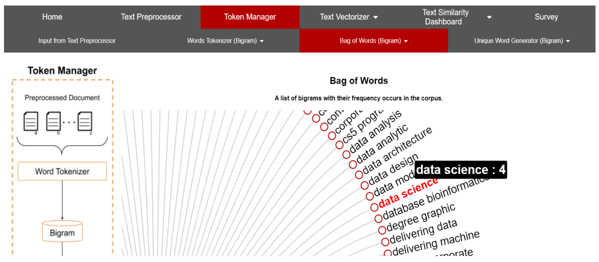
(a) Input from Text Preprocessor



(b) Words Tokenizer in Vertical Tree



(c) Words Tokenizer in Radial Tree



(d) Bag of Words (Bigram)

Figure 3: Sub-modules in Token Manager.

tact job#1@mail" as the input and removes some non-alphanumeric characters "hashtag (#)" and "at (@)" to get the output sentence "please contact job1mail". After passing the sentence through the SR module, for example, the sentence "experience in a number of following languages / tools sql python matlab SAS java spark scala" is changed to "experience number following languages tool sql python matlab SAS java spark scala" by removing the "in", "a", and "of" stop-words. All the above corrections in a sentence are shown on the top of the page as well by moving the cursor over the node.

The last two modules are WL and LC, which are to simplify the words. The WL module takes a sentence from the SR module and converts all the words to their semantic roots, for example, "help", "helping", "helps", and "helper" transformed to "help". In Fig. 2d, the input sentence from the SR module is "experience number following languages tools sql python matlab SAS java spark scala". The WL module takes that input and lemmatizes the word "languages" to "language" and "tools" to "tool". The changes are then shown as "languages $\rightarrow$ language and tools $\rightarrow$ tool" at the end of the node. Likewise, the LC module takes the sentence from the WL module and converts each character of a word into a lowercase. For instance, the LC module takes the output from the WL module, changes the word "SAS" to "sas", and shows "SAS $\rightarrow$ sas" at the end of the node. Thus, the final result of this sentence is "experience number following language tool sql python matlab sas java spark scala". After the LC module returns all the output sentences, the sentences are sent to Token Manager to create the Bag of N-grams DBs.

3.2 Token Manager

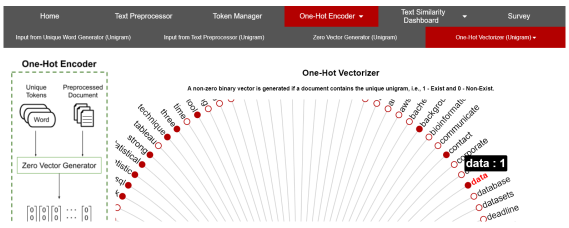
The Token Manager (TM) interface consists of the "Input from Text Processor" and other sub-module tabs that we describe in Section 2. Likewise, each tab contains both the TM module and the tree diagrams to show how each highlighted sub-module impacts on the text. The TM module starts with the input sentences generated from the TP module shown in Fig. 3a. The WT module takes these inputs and splits each sentence into a collection of single vocabularies called tokens. Each token can be Unigram, Bigram, or Trigram depended on the learners' selection. In this tokenization process, learners can select which N-gram token should be used as a vocabulary. Specifically, when learners click the WT tab and the "Bigram" option, the tree diagram branches out each sentence node to its token nodes shown as a huge zoomable vertical tree diagram in Fig. 3b. For instance, if the input sentence is "experience data analysis modeling method", after the bigram tokenization, "experience data", "data analysis", "analysis modeling", and "modeling method" are generated. As there is a large number of tokens in the text corpus and the vertical tree is very lengthy, learners can zoom and drag to the bottom to see the rest of the tree. By clicking the tree diagram, the vertical tree can also be transformed to the radial tree shown in Fig. 3c. Based upon the input tokens with their occurrence from the WT module, the BoW (Bigram) tab shows the radial tree, in Fig. 3d., with two layers, i.e., corpus and tokens, which displays all the bigrams with their total occurrence in the corpus. Learners can see the detail by moving the cursor over the node. For example, when learners move the cursor over the token node ”data science”, it displays the node in the red color and shows the text ”data science : 4”, which means that this token ”data science” occurs four times in this corpus. Finally, the UWG module takes the result from the BoW (Bigram) DB and generates the unique tokens. Token Manager then sends those unique tokens from the UWG module to Text Vectorizer.

3.3 Text Vectorizer

In Text Vectorizer, there are three options for learners to select: One-Hot Encoder, TF-IDF Encoder, and Word Probability Encoder. Each encoder has the three Bag of N-grams to be chosen. All encoders take the same inputs from the UWG module and the TP module respectively to generate the vectors.

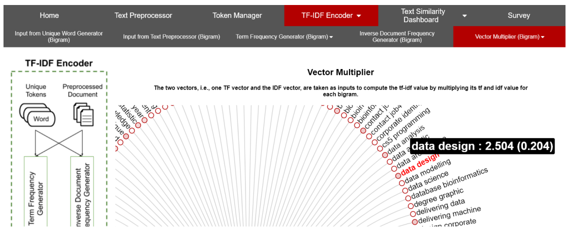
If the One-Hot Encoder is selected, the ZVG module takes the unique token from the UWG module as an input and generates the radial tree diagram with the white color nodes and the corresponding words for each job posting advertisement. The diagram is a zero-value vector of each document since each job document starts with all zero values and the length of each document vector is the same due to the number of unique tokens in the entire corpus. Learners can see that when the word ”data” is selected, the node shows ”data : 0”. There is a color scheme at the bottom of the page. The white color means a zero value. To visualize One-Hot encoded vectors, the OHV module takes the zero vector and the corresponding preprocessed job posting advertisement as the inputs to display the vector as a radial tree, in which the red node indicates the value ”1” and the white node means the value ”0”. If learners move the cursor over the node, the node is enlarged and displays the detail of the token. For example, in Fig. 4a, the word ”data : 1” in the red node is shown up. It means this word appears in this job posting advertisement.

If the TF-IDF Encoder is chosen, the TFG module takes the same inputs as that of the ZVG module and shows the radial tree diagram with different color intensity of nodes from white (0) to red (1) depending on the normalized color value of each token. For example, when the cursor is moved over the word ”data design”, the node is enlarged to display ”data design : 1 (0.16)”. The value of 1 means ”data design” appears only one time in this job posting advertisement and its normalized color intensity value is 0.16. The


Home
Text Preprocessor
Token Manager
One-Hot Encoder
Text Similarity Dashboard
Survey
Input from Unique Word Generator (Unigram)
Input from Text Preprocessor (Unigram)
Zero Vector Generator (Unigram)
One-Hot Vectorizer (Unigram)
One-Hot Encoder
Unique Tokens
Preprocessed Document
Word
Zero Vector Generator
One-Hot Vectorizer
data : 1

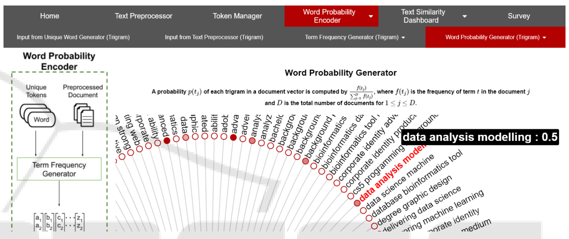
A non-zero binary vector is generated if a document contains the unique unigram, i.e., 1 - Exist and 0 - Non-Exist.

(a) One-Hot Encoder


Home
Text Preprocessor
Token Manager
TF-IDF Encoder
Text Similarity Dashboard
Survey
Input from Unique Word Generator (Bigram)
Input from Text Preprocessor (Bigram)
Term Frequency Generator (Bigram)
Inverse Document Frequency Generator (Bigram)
Vector Multiplier (Bigram)
TF-IDF Encoder
Unique Tokens
Preprocessed Document
Word
Term Frequency Generator
Inverse Document Frequency Generator
Vector Multiplier
data design : 2.504 (0.204)

The two vectors, i.e., one TF vector and the IDF vector, are taken as inputs to compute the tf-idf value by multiplying its tf and idf value for each bigram.

(b) TF-IDF Encoder


Home
Text Preprocessor
Token Manager
Word Probability Encoder
Text Similarity Dashboard
Survey
Input from Unique Word Generator (Trigram)
Input from Text Preprocessor (Trigram)
Term Frequency Generator (Trigram)
Word Probability Generator (Trigram)
Word Probability Encoder
Unique Tokens
Preprocessed Document
Word
Term Frequency Generator
Word Probability Generator
data analysis modelling : 0.5

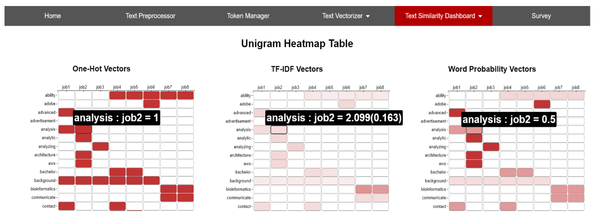
A probability $p(t_j)$ of each trigram in a document vector is computed by $\frac{f(t_j)}{\sum_{j=1}^{D} f(t_j)}$, where $f(t_j)$ is the frequency of term t in the document j and D is the total number of documents for $1 \leq j \leq D$.

(c) Word Probability Encoder

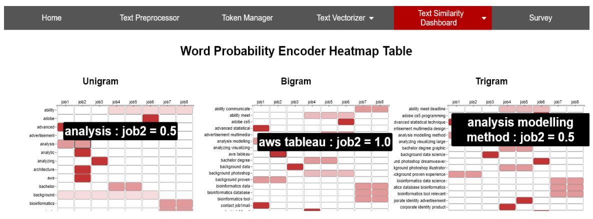
Figure 4: Text Vectorizer.

IDFG module calculates idf_i based upon the equation shown on the web page. For instance, as there are eight job posting advertisements in the corpus, N = 8. Because the token ”data design” appears in one documents, D = 1. By substituting these two values into the equation, the idf_i value of the word ”data design” is 2.504. Finally, the VM module takes both computed results from the TFG and IDFG modules as the inputs and performs the multiplication for each token to generate a TF-IDF vector, which is shown as a radial tree diagram in Fig. 4b. ”data design: 2.504 (0.204)” is displayed with the tf-idf value (2.504) and the normalized color intensity value (0.204).

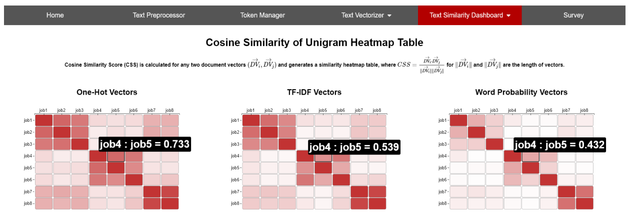
If the Word Probability Encoder is chosen, the TFG module generates the TF vector as the same way as that of TF-IDF Encoder. For example, the radial tree diagram shows the trigram ”data analysis modeling : 1 (0.186)”, which means ”data analysis modeling” appears only one time in job1. The WPG module then (1) counts the number of times of a token appeared in a specific document, (2) computes the total frequency of that token appeared in all the document in the entire corpus, and (3) divides these two values to get the token probability in the specific document.



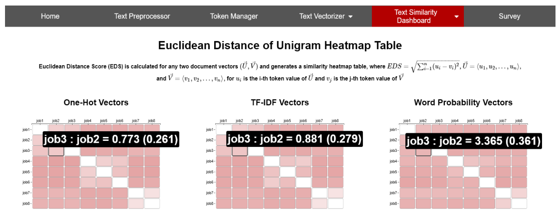
(a) Unigram Heatmap Table



(b) Word Probability Heatmap Table



(c) Cosine Similarity Heatmap Table


Home
Text Preprocessor
Token Manager
Text Vectorizer
Text Similarity Dashboard
Survey
Euclidean Distance of Unigram Heatmap Table
One-Hot Vectors
TF-IDF Vectors
Word Probability Vectors
job3 : job2 = 0.773 (0.261)
job3 : job2 = 0.881 (0.279)
job3 : job2 = 3.365 (0.361)

(d) Euclidean Distance Heatmap Table

Figure 5: Text Similarity Dashboard.

In Fig. 4c, the probability of the token “data analysis modeling” is 0.5, which means this token is rare and can be found in only two documents.

3.4 Text Similarity Dashboard

After the Text Vectorizer generates all the One-Hot, TF-IDF, and Word Probability vectors, they are sent to Text Similarity Dashboard for similarity evaluations. In the Text Similarity Dashboard tab, learners can select: (1) N-gram Heatmap Table, (2) Vectorizer Heatmap Table, (3) Cosine Similarity Matrix, and (4) Euclidean Distance Matrix. For the N-gram Heatmap Table, learners can choose unigram, bigram or trigram. For example, when learners select ”Unigram”, shown in Fig. 5a., the table for each vectorization approach has the rows as the tokens and the columns as the jobs. In the table of One-Hot vectors, if the token appears in a specific document, it is displayed with a red color box and the value ”1”, or else it shows a white color box with the value ”0”. When the cursor is moved over the token ”analysis” in the second column, it displays ”analysis : job2 = 1”, i.e., the token ”analysis” appears in the job2. In the table of TF-IDF vectors, learners can see different color intensity of mixing red and white due to their tf-idf values. For instance, ”analysis : job2 = 2.099 (0.163)” means the tf-idf value of the word ”analysis” is 2.099 and its normalized color intensity value is 0.163. The table of Word Probability vectors shows “analysis : job2 = 0.5”, which means the probability of token “analysis” appears in job 2 document is 0.5. Similar to the N-gram Heatmap Table, the Vectorizer Heatmap Table display the ”Text Vectorizer vs. Document” among all the N-grams in Fig. 5b.

The text similarity among the job posting advertisements can be compared by computing the CSS score and EDS score. The Cosine Similarity Calculator takes the job posting vectors and substitutes them into the given formula to generate the heatmap tables for each vectorization approach. For the One-Hot vectors, the same job posting advertisement is displayed with a red color and the value ”1” shown in the diagonal of the tables, as they are the same job posting. In the table, learners can see that there are three red color clusters, including (1) Job 1, 2, and 3, (2) Job 4, 5, and 6, and (3) Job 7 and 8. It means that those jobs in their own clusters are very similar. Job 1, 2, and 3 are about data scientists; Job 4, 5, and 6 are about graphic designers; and Job 7 and 8 are about mechanical engineers. When learners move the cursor over a red cell, they can see the detail of it. For example, in Fig. 5c, it shows ”job4 : job5 = 0.733”. This is the value of the CSS score between Job 4 and Job 5. Learners can apply the same value scheme to view the CSS score of TF-IDF and Word Probability between any two jobs. The Euclidean Distance Calculator also takes the same inputs to compute the distance value. However, the scores show in the Euclidean Distance Heatmap Table are different from those shown in the Cosine Similarity Heatmap Table. For instance, in Fig. 5d, it shows ”job3 : job2 = 0.733(0.261)” which are the value of the EDS score between Job 3 and Job 2 and the normalized color value. Likewise, learners can apply the same value scheme to view the EDS score of TF-IDF and Word Probability between any two jobs. Note that the reason why these three tables in each vectorization approach look quite similar to one another is because these three approaches have quite similar vectors with different scaling values only, i.e., 0 or 1 for One-Hot encoding, ≥ 0 for TF-IDF, and 0$\sim$1 for Word Probability. Due to the large number of tokens in our corpus, most of the vectors are sparse vectors (i.e., a lot of zero values in each vector), as many tokens do not appear in each job posting advertisement. Hence, both scores of all three methods are very close to one another.

4 CONCLUSIONS AND FUTURE WORK

To our best knowledge, this is the first paper to introduce a visual-based educational support platform for learning statistical NLP Analytics. Specifically, we develop and implement a web-based, interactive visual NLP learning platform that enables novice learners to study the core processing components of statistical NLP analytics in sequence. The contributions of this work are three-fold: (1) the ease of learners to access and use our platform through any web browser at no cost; (2) the interactive and dynamic visuals (e.g., mouseover events, collapsible tree diagrams, and animations) that enhance the study environment and learners' engagement; and (3) the in-focus step-by-step process, using the job posting classification as an example, to demonstrate the core processing components of statistical NLP approaches. However, there is still a lack of many important research questions, e.g., how the 3D diagrams should be designed and implemented to strengthen the interactivity between novice learners and the platform, how neural network-based methods should be developed visually to deliver the knowledge and to be a contrast with statistical-based approaches, and how the platform can be further enhanced to visualize a NLP pipeline to solve real-world problems.

REFERENCES

Bengfort, B., Bilbro, R., & Ojeda, T. (2018). Applied Text Analysis with Python: Enabling Language-Aware Data Products with Machine Learning (1st ed.). O'Reilly Media.

Bostock, M. (2020). D3.js - Data-Driven Documents. D3JS. https://d3js.org/

Data Science. (2020a). WPI Data Science Program. https://www.wpi.edu/academics/departments/data-science

Data Science. (2020b). Harvard University Data Science Program - The Graduate School of Arts and Sciences. https://gsas.harvard.edu/programs-of-study/all/data-science

Deng, L., & Liu, Y. (2018). Deep Learning in Natural Language Processing (1st ed. 2018 ed.). Springer.

Goldberg, Y. (2017). Neural Network Methods for Natural Language Processing. Synthesis Lectures on Human Language Technologies, 10(1), 1–309. https://doi.org/10.2200/s00762ed1v01y201703hlt037

Google Translate. (2020). Google Translation Website. https://translate.google.com/

Grammarly. (2020). Write your best with Grammarly. https://www.grammarly.com/

Hastie, T., Tibshirani, R., & Friedman, J. (2020). The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics) (2nd ed.). Springer.

Kamath, U., Liu, J., & Whitaker, J. (2020). Deep Learning for NLP and Speech Recognition (1st ed. 2019 ed.). Springer.

Lane, H., Hapke, H., & Howard, C. (2019). Natural Language Processing in Action: Understanding, analyzing, and generating text with Python (1st ed.). Manning Publications.

Liu, Y., Xu, Q., & Tang, Z. (2019). Research on Text Classification Method Based on PTF-IDF and Cosine Similarity. 2019 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS). doi:10.1109/iciibms46890.2019.8991542

Monster. (2017, May 26). Job Search, Find Job Openings Monster.com. https://www.monster.com/jobs/search/

Pattnaik, S., & Nayak, A. K. (2019). Summarization of Odia Text Document Using Cosine Similarity and Clustering. 2019 International Conference on Applied Machine Learning (ICAML). doi:10.1109/icaml48257.2019.00035

Protege. (2020). Protege. https://protege.stanford.edu/

Reese, M. R., & Bhatia, A. (2018). Natural Language Processing with Java: Techniques for building machine learning and neural network models for NLP, 2nd Edition (2nd Revised edition). Packt Publishing.

Rehman, Z., & Kifor, S. (2015). Teaching Natural Language Processing (NLP) Using Ontology Based Education Design. Balkan Region Conference on Engineering and Business Education, 1(1), 206–214. https://doi.org/10.1515/cplbu-2015-0024

Rodrigues, M., & Teixeira, A. (2015). Advanced Applications of Natural Language Processing for Performing Information Extraction (SpringerBriefs in Speech Technology) (2015th ed.). Springer.

Shaalan, K., Hassanien, A. E., & Tolba, F. (2017). Intelligent Natural Language Processing: Trends and Applications (Studies in Computational Intelligence (740)) (1st ed. 2018 ed.). Springer.

Statistics and Data Science MicroMasters. (2020). Statistics and Data Science MicroMasters. https://micromasters.mit.edu/ds/

Text Analysis. (2020). Text Analytics Website MonkeyLearn. https://monkeylearn.com/