Word Proximity Integration in Word Cloud

Ansuya Ahluwalia[*], Chaitanya[**], Shailendra Singh[***]
(Department of Computer Science, PEC University of Technology, Chandigarh-12)

ABSTRACT

Word clouds are used for giving a quick overview of textual content based on the frequency of the terms that occur. The conventional visualization displays the most frequent words by reflecting it in the font sizes. In this project, we introduce a new methodology for calculating the semantic proximity between frequently occurring words by considering the horizontal radial distance of the high frequency word pairs. The proximity counts are then integrated in the word cloud visualization by varying the color intensity of the close-distanced words.

Keywords - ***colour intensity, term frequency, term proximity, word cloud***

1. Introduction

A word cloud is a visual representation for text data, typically used to depict a focused view of a document by displaying words that are most often used in the entire document. Tag clouds have completely redefined the orthodox wisdom about how visualizations ought to work. [1].These high frequency single-word tags highlight the importance of each word/tag by reflecting it in the font size or color. Word clouds have been subject of investigation in several usability studies [2] [3] [4].

In this paper, we have focused on how to utilize both the font size as well as color to get a deeper insight into the association of words in an input document. We integrate other novel aspects to our word cloud algorithm such as visualization of co-occurring or closely located high frequency terms based on how far apart they are in the input corpus; they commensurate with each other in visual parameters.

Some work has been done in this field as well. One such method of integrating the semantic proximity of words in a document is by using a tree cloud [5]. Here, the words are arranged on a tree to reflect their semantic proximity according to the text. In contrast, our methodology focuses on integrating the semantic counts in the conventional word cloud visualization i.e. a bubble looking figure integrating term frequencies by varying font size [6]. Doing so we are exploiting the potential of the conventional visualization and varying color shades to reflect word proximities.

Other exploitations of word cloud have also been made such as establishing relations among tags by taking into consideration the textual content of documents [7]. Such approaches are based on qualitative rather than quantitative analysis of the text.

2. Methodology

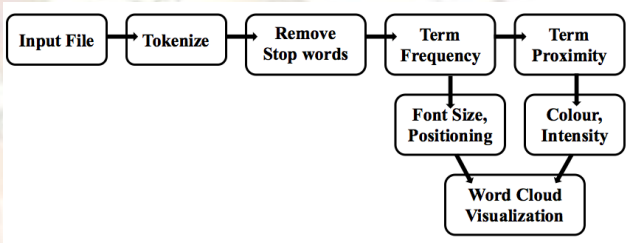

Figure 1. Methodology

Fig.1 depicts a flowchart of the methodology we have employed. We start off by reading an input file containing text using Python's input functions [8], subsequently tokenization of the text using regular expressions is done [9], followed by removal of stop words [10] from the list of tokens generated.

Term frequency is calculated for each of the tokens and saved in a Python dictionary in the manner *token: frequency*. We then calculate the term proximity of the high frequency terms. These weighted lists of terms are further fed into our Java program to generate a visualization that depicts the high frequency terms by adjusting the font sizes, and the proximity between these terms by adjusting the color shade intensities.

For programming the graphics, prerequisite knowledge is required in the areas of Object Oriented Programming, in particular the concepts of inheritance and polymorphism for designing classes, GUI and custom graphics programming [11], graphics programming using Java 2D [12], 2-Dimesional representation or coordinate representation and orientation of terms – angles, size, etc.

3. Weighted List of Tokens

Concepts of Natural Language Processing and statistics were highly crucial for tokenizing text into sentences and sentences into words, and subsequent generation of weighted frequency and proximity count lists. The following concepts were implemented in Python:

a) *Tokenization*: the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens [13].
b) *Stop words:* are words that are filtered out prior to, or after, processing of natural language data. Any group of words can be chosen as the stop words for a given purpose. For e.g. – 'is', 'that', 'which', 'the', etc. [10].
c) *Term Frequency:* (TF) is a measure of how often a term is found in a collection of documents or a specific document.
d) *Co-occurrence of tokens:* in this linguistic sense can be interpreted as an indicator of semantic proximity or an idiomatic expression.
e) *Term proximity measures:* to measure how close two terms are to each other. This is determined by either the number of characters or the number of words present between those two terms.
f) *Regular Expressions*: are required to perform tokenization of words in the input corpus [9]. Regex is implemented using methods given in re python library.
g) *Dynamic data structures:* such as dictionaries are needed for storing terms and their frequencies. Python libraries like *defaultdict* are used for working with dictionaries and *Counter* are used for calculating term frequencies for the terms listed in the dictionary [8].
h) Sorting techniques

3.1 Tokenization and Frequency Count

To calculate the frequencies of the terms, we first tokenized the entire text corpus using the regular expression "[\\w-]*\\w" which parses the text to take out terms that start with any character from the set {A-Z, a-z, 0-9, _} or contains a dash, repeating for as many multiple occurrences and terminates with any character from the former set. Then we remove the *stop words* in this list of terms [10] and sort the terms in descending order, based on the number of times they occur in the corpus. We use at least 50 high frequency words (arbitrary) in our visualization if they contain at least 5 different frequencies in the list, up to at most 70 high frequency words if the number of different frequencies is less than 5 in the initial list of 50.

3.2 Proximity Count of Tokens

We came up with a novel algorithm for calculating word proximity between two high frequency words. We calculate the proximity between two words based on the radial distance between them (default is taken as 2). If a word is immediately adjacent to the word in consideration, then its proximity value is taken as 20. Alternatively if a word is one hop away from the word in consideration, then its proximity value is taken as 10. Any words out of this range are not considered. Also, even if a word is one hop or two hops away from the word in consideration but is present in the next sentence, then its proximity count is not considered. Summarily this algorithm only applies to high frequency words that are present in the same sentence [13]. For each distinct pair, we calculate the term proximity as;

$$T.P.(\tau_1, t_2) \quad \sum t.p.(\tau_1, t_2) \quad (1)$$

where $t.p.(t_{..}, \tau_2) <$ 20 and $t.p.(t_{..}, \tau_2)$ is the proximity of terms τ_2 to τ_1 calculated by the number of tokens present between terms τ_1 and τ_2. If two terms are adjacent to each other then they are assigned a value of 20; each increasing distance is denoted by decreasing multiples of ten; if a term is two terms away from the concerned term, then the proximity value is taken as 10. For all terms that fall outside the horizontal radial distance of the concerned term, their proximity count is taken as zero.

$T.P.(\tau_1, t_2)$ is the aggregate of all these values for all values less than 30. The pair of terms having high $T.P.$ values are generally present in close proximity in the corpus and has one similar visual attribute to represent them as closely occurring terms. An aggregate of all the proximity counts between all the word pairs gives us an overall idea of how frequently and closely the two words in each pair occur in the piece of text.

We have also removed duplicate considerations of the same word pairs in a sentence, based on their positioning in the sentence and inclusion in the high frequency word list. Therefore, for a list of *n* high frequency terms, there are *n*(n-1)/2* word pairs that are examined for proximity calculation.

The result is stored as a list of terms tab spaced with their frequencies in a .txt file. This file is utilized in the Java code for creating the world cloud visualization.

4. Visualization of Close Distanced Terms

Visualization for the word cloud is based on the frequencies and proximities given as the weighted list in the .txt file. The weighted lists are stored as

HashMaps [14]. Following concepts are crucial for designing the word cloud visualization [15]:

a) *Tag size*: Large tags attract more user attention than small tags.
b) *Scanning*: Users scan rather than read tag clouds. Larger font size helps to determine the title/story/topic of the document and related color intensity symbolizes the close relation of the words.
c) *Position*: The upper left quadrant receives more user attention than the others as observed in the western reading habits.
d) *Exploration*: Tag clouds also provide suboptimal support when searching for specific tags, which do not have a very large font size.
e) *Integrating word proximity using color*: A *HashMap* of colors is maintained; every element of this *HashMap* contains the list of hex codes of colors of same configurations but with varying intensities. Standard deviation is used to calculate the cut off proximity value.

The visualization is based on font size, font, word positioning and color.

4.1 Word Font Size

As mentioned before, large tags attract more user attention than small tags. The font size of the word is determined by the word frequency given by equation (2), accessed from the *HashMap*;

$$Font\ Size \propto Term\ Frequency \qquad (2)$$

For the less frequent words, resulting font size comes out to be very less. For more readability of the low frequency words, a constant is also added to the above equation. The font of the word is set as Calibri by default. User can select the word font as per their requirements.

4.2 Word Positioning

Positioning of every term is done by setting (x, y) coordinates for them. The word frequency and the length of the word determine coordinates [15]. Different algorithms have been used to calculate the x-coordinate and y-coordinate.

X-coordinate directly depends on the frequency of occurrence and the number of alphabets in the particular word. The space occupied by the word is directly proportional to the word length and to the word font size that is determined by the word frequency. We are using equation (3) for determining x-coordinates for the terms;

$$x \quad coordinate \quad t.l.* t.f.* \alpha + t.l.* \beta \qquad (3)$$

where $t.l.$ is the term length, $t.f.$ is the term frequency, and α is taken as 3 and β as 4.5.

Y-coordinate for every single row is adjusted as the maximum height of the word in that row; height of the word directly depends on its frequency of occurrence in the specified text; hence y-coordinate is determined by the word frequency using equation (4);

$$y \quad coordinate \quad Max\ (t.f.* \gamma) \qquad (4)$$

where γ is taken as 5.

4.3 Word Integrating Word Proximity using Color

Word proximity integration is done by reading the weighted proximities list stored in a .txt file. *word_proximity_map* is a *HashMap* that maintains the information about term proximities. list_of_colors is a *HashMap* that maintains the list of colors to be used. Every element of this *list_of_colors* is mapped onto the list of hex codes of colors of same configurations but with varying intensities in the increasing order. The *word_proximity_map* is accessed sequentially.

For every term *t* that has not been assigned the color code yet, a color *c* is selected from the *list_of_colors* and assigned to the word *t;*

$$P_t \quad \{(t(i).x(i)) \mid 0 < i < n\} \qquad (5)$$

where *t(i), x(i)* denotes the term and its proximity with the concerned term *t;*

$$stdDev \quad \sqrt{\{\sum_{i=1}^{n}[(x(i) \quad x)^2/n^2\}}$$
(6)

where *stdDev* is the standard deviation of the term proximities with term t.

Standard deviation shows how much variation or dispersion exists form the mean; low standard deviation indicates that the data points tend to be very close to the mean; high standard deviation indicates that data points are spread out over a range of values [16]. Hence, standard deviation is used to calculate the cut off proximity value. For every *x(i) > stdDev, t(i)* is assigned a color from the color set. Color intensity varies proportionally with the proximity value as specified in the equation below; greater proximity value means greater intensity and lower the proximity value, lower is the intensity. Higher the word proximity, higher is the color intensity. The color codes are arranged in the increasing order of their intensity in the color set.

Hence, the index selected from the color set is determined by the proximity value.

For other terms that have already been assigned the same color, we calculate the standard deviation of the term proximities with the concerned term as described and assign them a color again using the above stated formulae.

5. Results

Figure 2. Word cloud visualization

Fig.2 represents the visualization of the 70 most high frequency words by adjusting it in the corresponding font sizes. The Fig.2 also demonstrates the proximity between all these terms by adjusting the color shades and intensities. In Fig.3 are specified few of the most frequent terms from the example document. Taking an example, consider the term *princeton*. It has a fairly decent frequency count as shown in Fig.3 and has a fairly decent font size corresponding to it. The term *attend* has a low frequency count and thus appears smaller in the visualization.

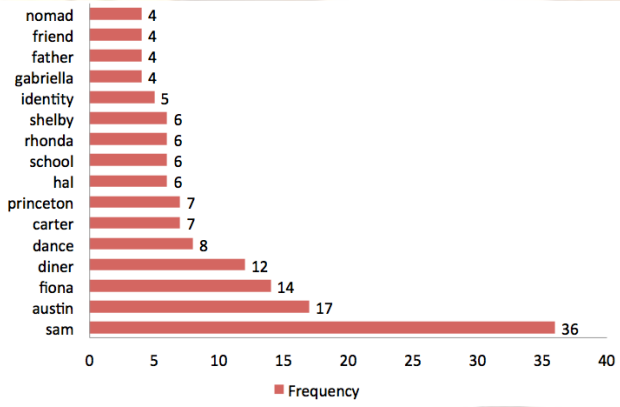

Figure 3. Few high frequency terms

However, the terms *princeton* and *attend* have the highest proximity among all high frequency term pairs, with a value of 60 as shown in Table 1. Thus, we have considered an arbitrary, relatively dark color to represent this proximity from our list of colors. Furthermore, after separating all the terms that have high and low proximity with the concerned term i.e. *princeton* in this case, we allot corresponding, varying color intensities to the different terms that pair with it. Different colors appear in the visualization, as we have not considered terms paired with the concerned term, that fall too below the calculated standard deviation, otherwise the entire visualization would have been of different shade intensities of a single color.

Table 1. Few Proximity Counts

Term 1	Term 2	Proximity
princeton	attend	60
sam	austin	50
sam	finds	40
school	football	30
identity	austin	30
sam	rhonda	30
sam	confronts	30
sam	sees	30
sam	reluctant	30
fiona	dance	20
girl	diner	20
fiona	save	20
carter	rhonda	20
sam	day	20
named	fiona	20
girl	named	20

6. Conclusion

We have successfully implemented Natural Language Processing techniques for parsing text and calculated the term proximity values successfully by devising a novel algorithm. Finally, we successfully integrated these proximity values into our word cloud visualization by using a color intensity variation approach.

The visualization is extremely useful in understanding the overview of documents and demonstrating the semantic closeness between important words found in the text. However, the visualization technique used can achieve higher resolution and optimal space coordination by employing sophisticated mathematical equations like spiral equations for better emphasis of the high frequency words [17] [18] [19]. Also, the variation in color shades is not as prominent due to the choice of color shades. One can refer to a better-resolved color shades template for showing the decreasing proximity between words. This way closely related words would appear highly distinct and discernable. Although our algorithm does succeed in computing

semantic proximity count between high frequency terms and integrates it in the visualization, more rigorous methods as suggested can be employed for designing the visualization.

REFERENCES

[1] Fernanda B. Viégas, Martin Wattenberg, Timelines tag clouds and the case for vernacular visualization, *Magazine interactions – Changing energy use through design Interactions, ACM,* New York, USA, 2008, vol. 15, issue 4, 49-52.

[2] Seifert, C., Kump, B., Kienreich W., Granitzer, G., Granitzer, M., On the Beauty and Usability of Tag Clouds, *Information Visualisation, 2008. IV '08. 12th International Conference* , 17-25.

[3] Cui, Weiwei, Yingcai Wu, Shixia Liu, Furu Wei, Michelle X. Zhou, Huamin Qu, Context preserving dynamic word cloud visualization, *Pacific Visualization Symposium (PacificVis), IEEE 2010,* , 121-128.

[4] Cidell, Julie, Content clouds as exploratory qualitative data analysis, 2010, Area 42.4: 514–523.

[5] Gambette, Philippe, Jean Véronis., Visualising a text with a tree cloud, *Classification as a Tool for Research*, 2010, 561-569.

[6] Wordle - Beautiful Word Clouds, http://www.wordle.net/, 22-01-2013.

[7] Zubiaga, Arkaitz, et al., Content-based clustering for tag cloud visualization, *Social Network Analysis and Mining, ASONAM'09, International Conference on Advances in IEEE,2009,* 316-319.

[8] Python Documentation contents, http://docs.python.org/2/contents.html, 22-01-2013.

[9] Regular Expression Tutorial - Learn How to Use Regular Expressions, http://www.regular-expressions.info/tutorial.html, 22-01-2013.

[10] English Stopwords, http://www.ranks.nl/ resources/stopwords.html, 22-01-2013.

[11] Custom Graphics Programming - Java Programming Tutorial, view-source:http://www3.ntu.edu.sg/home/ehchua/programming/java/J4b_CustomGraphics.html, 22-01-2013.

[12] 2D Graphics; Java2D, http://www3.ntu.edu.sg/ home/ehchua/programming/java/J8b_Game_2DGraphics.html, 22-01-2013.

[13] Natural Language Toolkit; NLTK 2.0 documentation, http://nltk.org/, 22-01-2013.

[14] Java HashMap- Java Tutorial, http://www.rose india.net/javatutorials/javahashmap.shtml, 23-01-2013.

[15] Bateman, Scott, Carl Gutwin, Miguel Nacenta, Seeing things in the clouds: the effect of visual features on tag cloud selections, *Proceedings of the nineteenth ACM conference on Hypertext and hypermedia, ACM,* New York, NY, USA,*2008, 193-202.*

[16] Standard Deviation, http://en.wikipedia.org/wiki/ Standard_deviation, 23-01-2013.

[17] Spirals, http://www.mathematische-basteleien.de /spiral.htm, 22-01-2013.

[18] Mathematica, http://mathematica.stackexchange .com/questions/2334/how-to-create-word-clouds /2360#2360, 22-01-2013.

[19] WhyDoIDoIt.com, http://whydoidoit.com/, 22-01-2013.