



DECEMBER 11, 2017

STORY NEXT 2.0


A TEXT INSIGHTS/VISUALIZATION TOOL

WHAT **STORY** DOES YOUR CONTENT SAY?

<https://github.com/biswarajkar/storynext2>

BISWARAJ KAR
ANDREW KRISCHER
LING ZHANG

PROJECT REPORT FOR **NATURAL LANGUAGE PROCESSING (CS 6120)**
COLLEGE OF COMPUTER AND INFORMATION SCIENCES
NORTHEASTERN UNIVERSITY



INTRODUCTION

There have been multiple avenues where NLP has transcended into our lives, making literary perception and text analysis much easier. Be it automatic News summarization or sentiment analysis of political speeches and news, new NLP methods strive to better our understanding of human ideas. Taking the intuition from the same, we worked on a project that aims to help **content-writers** uncover subtle patterns in their creative work, such as potential biases and overall sentiment.

The **motivation** for the project stemmed from discussions we had with some writers and journalists, and their common curiosity, if technology can be used to make story-writing more introspective and insightful. As a result, we had some **guiding questions**:

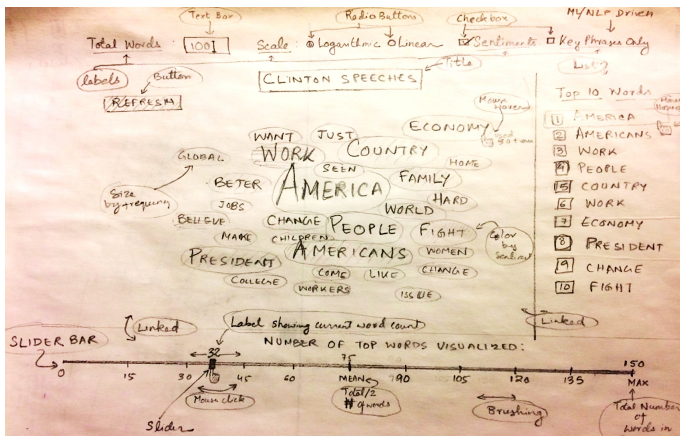
- What if there was a tool that shows what sentiments your writing conveys in its words and sentences, without reading line-by-line of the writing?
- What if we condense a 10-page story into one a one-page interactive visualization, conveying the flow of emotions, sentiments and word usage; all in a succinct manner?

Our project goal was to have a visualization (driven by NLP methods) which would show sentiments in a text in an elegant way so that it gives a clear picture of the polarity/mood of the written text/article/story at a glance and enable the user to get interesting insights just by looking at the visualization.

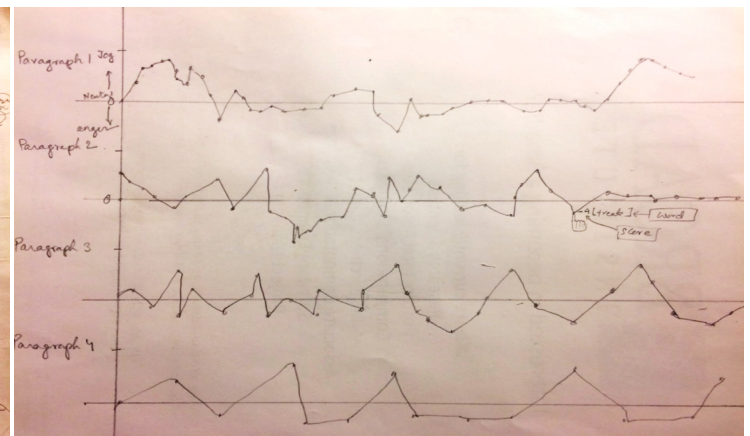
This project studies ways to estimate and visualize sentiment from stories or text snippets input by the user. Sentiment is defined as "an attitude, thought, or judgment prompted by feeling." Our specific goal is a visualization that presents basic emotional properties embodied in the text, specifically in the domains of **News** and **Literature**. Input/Output: *our tool takes a piece of text (articles/stories/news-piece) as input and generates a series of sentiment visualizations as output.*

INITIAL IDEAS/SKETCHES/EXAMPLES

Building up on the ideas and driving questions, we came up with some sketches as to how our project would look:



Can we try showing sentiments in a word cloud, coloring the word cloud by the sentiments we calculate using NLP methods and indicating frequent usage of words by sizing? Illustrated above on the left.



Alternatively, can we try showing the "flow" of sentiments in a long story as the text/narrative progresses? As illustrated above on the right, where the paragraphs in a story moves from top-to-bottom, as they appear in the text.

RELATED WORK

There have been many papers written on sentiment analysis for specific domains of blogs, microblogging and product reviews. ([1]Pang and Lee 2002) give a survey of sentiment analysis. Researchers have also analyzed the brand impact of microblogging (Jansen). Overall, text classification using machine learning is a well-studied field ([3]Manning and Schutze, 1999). ([1]Pang and Lee 2002) researched the effects of various machine learning techniques (Naive Bayes (NB), Maximum Entropy (ME), and Support Vector Machines (SVM)) in the specific domain of movie reviews. They were able to achieve an accuracy of 82.9% using SVM and a unigram model. Researchers have also worked on detecting sentiment in text. ([4]Turney 2002) presents a simple algorithm, called semantic orientation, for detecting sentiment. ([1]Pang and Lee 2004) present a hierarchical scheme in which text is first classified as containing sentiment, and then classified as positive or negative.

We could not find any papers that analyzes NLP techniques in the specific domain of story writing or journalism, probably because digital story-telling and real-time opinion mining is a relatively new field. Essentially, that means we have the following attributes in our project, where we are attempting to do something new or different:

- The domain we are targeting (News & Literature) is very new when it comes to application of sentiment analysis or opinion-mining and is an area of active research. As a result, we were not able to find any annotated datasets for our domains to train our model in. That would mean that we have to create own corpus for training and (or) testing, by manually tagging documents ourselves.
- We found that Interestingly, previous attempts like classifying movie reviews ([1]Pang and Lee 2002) were a relatively a less complex task as the Star Rating indicators could easily give scale to ascertain polarity (e.g. on a 0-9 rating scale, [1-3][4-6][7-9] can easily be tagged as [Neg][Neu][Pos]). Similarly, for tweets or blogs, where the presence of certain types of emoticons can imply sentiment directly ([2]Yang et al.); which doesn't apply in our case and classification is a much more complex task due to size and scope of news-articles/literature.

METHODOLOGY

TRAINING DATA CHOICES & BACKGROUND

Our research revealed that the only reliable pre-annotated large sentiment corpora are available in a handful of domains, like Twitter, Movie Reviews, Product Reviews on Amazon and Blogs based on emoticons. Hence, non-availability of annotated training data is our biggest challenge. After an extensive search of many publications and research corpuses (Stanford, Brown, UPenn, UCI ML Repository, Kaggle etc.), we decided that we will attempt classification using language-models trained from the Movie Corpus of ([1]Pang and Lee 2002). We chose the IMDB Movie Corpus^[14] among all the options to train our model due to some inherent characteristics of the movie review corpus, which we talk about below:

- The movie review corpus has diverse data that is comprised of one or many paragraphs of user opinion; its classification is not bound by size or specificity of features as in the case of Twitter (140 characters) or presence of special features like emoticons.
- The movie review corpus is sufficiently large (5844680 words) to achieve some generality of natural language, given that the reviews are actual human opinions too (much like News & stories). The size and depth of the corpus also gives it a little bit more domain independence.
- Many other successful sentiment analysis mechanisms, like the Stanford NLP ([5]Socher, Richard, et al.) also use the Movie review corpus and yet are more widely applicable to any domain. Though they employ more advanced techniques like *Recursive Neural Networks & Deep Learning*, the inherent base learning model is indeed the same movie review corpus, which hints at some merit for using the corpus.

The movie review dataset has a total of 25000 IMDB reviews for training. The corpus contains already-tagged sentiment data, making it a perfect candidate for training in our early stages of the project.

TECHNIQUES & IMPLEMENTATION

Extending our plan to visualize sentiment, we planned to incrementally try the following techniques to classify the sentiment of our data, starting with the most naïve/simple technique, moving to more complex techniques to get higher accuracy:

1. Multinomial Naïve Bayes
2. Support Vector Machines & Random Forest
3. Recurrent Neural Network with LSTM Units
4. Valence and Arousal Linguistic Model

We used Unigram and Bigram models for feature representation of our data as prior research^{[1]-[10]} suggested that going with higher-order language models have shown either no, or very little improvement in accuracy, at least w.r.t. sentiment analysis. We did not transform the training data too much as the training data was already classified and annotated in the dataset (^[1]Pang and Lee 2002). To clean our data, we first imported the data using the NLTK python library and then removed any empty sentences. We then tabulated unigram and bigram counts and stored them as a dictionary in PKL format. This allows us to easily reload and reuse the preprocessed data for any algorithm we choose in the future. We then ran our models on the hand classified Literature and News articles and evaluated the results.

EVALUATION & TESTING METHODS

We did a mix of Intrinsic and Extrinsic evaluations to measure the success of the project. For the extrinsic part, we tested our classifier on a set of real stories and news articles which will be hand-annotated by us. Additionally, we employed some standard intrinsic techniques like computing accuracy (^[3]Manning and Schutze, 1999) of the classifier on the whole evaluation dataset, defined as:

$$precision = \frac{True\ Positives}{True\ Positives + False\ Positives} , \quad recall = \frac{True\ Positives}{True\ Positives + False\ Negative}$$

Where a *True Positive* implies a text being actually of positive sentiment and the classification also gives a positive sentiment label to the text. *False Positive* implies that a text being actually of negative sentiment and the classification also gives a positive sentiment label to the text. Similarly, the *False Negative* measure was also derived. To measure the performance, we also used F-measure (^[3]Manning and Schutze, 1999), which is the harmonic mean of precision and recall:

$$F = (1 + \beta^2) \cdot \frac{precision \cdot recall}{\beta^2 precision + recall} , \quad F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

We also calculate the Accuracy in some cases to get a more quantitative measure of success tied to results. As we discussed in the previous section, we **hand-annotated 150 documents** with sentiment tags in the domains of **Literature** and **News** for our **cross-domain testing**.

EXPERIMENTS

We were interested in how effective each of our techniques were and the same are detailed each of the sections below. We talk about the intuition for working with the technique, methodology, evaluation and the summary:

MULTINOMIAL NAIVE BAYES

The simplest approach to text classification is using Naïve Bayes. Let $\{f_i, \dots, f_m\}$ be a predefined set of m features that can appear in a document and let $n_i(d)$ be the number of times f_i occurs in document d . It works by assigning to a given document d , the class $c^* = \arg \max_c P(c|d)$. We derive the Naïve Bayes (NB) classifier by first observing that by Bayes rule,

$$P(c | d) = \frac{P(c) \cdot P(d | c)}{P(d)}$$

where $P(d)$ plays no role in selecting c^* . To estimate the term $P(d | c)$, Naive Bayes decomposes it by assuming the f_i 's are conditionally independent given d 's class:

$$P_{NB}(c | d) = \frac{P(c) \cdot (\prod_{i=1}^m P(f_i|c)^{n_i(d)})}{P(d)}$$

Our training method consists of relative-frequency estimation of $P(c)$ and $P(f_i|c)$, using add-one smoothing.

EVALUATION

# of Documents	Predicted Positive	Predicted Negative
Actual Positive	36 (TP)	40 (FN)
Actual Negative	33 (FP)	42 (TN)

Naïve Bayes	Positive Sentiment Test	Negative Sentiment Test
PRECISION	52%	51%
RECALL	47%	56%
F1	49%	53%
ACCURACY	52%	51%

For positive sentiments, NB results in a precision of 52% and recall of 47%, which are rather abysmal results. For negative sentiments, NB results in a precision of 51% and a recall of 56%. Despite its simplicity and the fact that its conditional independence assumption clearly does not hold in real-world situations, Naive Bayes-based text categorization still tends to perform surprisingly well (Lewis, 1998 & Pang & Lee) when the domains of training and test are same, however, not so much in our case.

FEATURE EXTRACTION EXPERIMENTS

- We attempted tagged special tokens like ‘_NOT’ when encountering negation tokens like “n’t”, not, no, never, etc. However, we did not find considerable change in results.
- Attempted Binary classification by checking for presence of words(vs. Frequency) with no improvement.

SUMMARY

Naïve Bayes being the simplest technique gives us our baseline for evaluation of our language models and the poor result was expected, given our cross trained data. The key learnings for Naïve Bayes are:

- + Fastest model to train, computationally inexpensive and easy to implement
- Conditional independence assumption ignores relative context

SUPPORT VECTOR MACHINES (SVM) AND RANDOM FOREST

The next attempt after Naive Bayes took us to another very popular technique employed by many researchers in this area, Support vector machines. SVMs have been shown to be highly effective at traditional text categorization. They are large-margin, compared to probabilistic classifiers (in contrast to Naive Bayes and MaxEnt). In the two-category case (like ours), the basic idea behind the training procedure is to find a hyperplane, represented by vector \vec{w} , that not only separates the document vectors in one class from those in the other, but for which the separation, or margin, is as large as possible. This search corresponds to a constrained optimization problem; letting $c_j \in \{1, -1\}$ (corresponding to positive and negative) be the correct class of document d_j , the solution can be written as:

$$\vec{w} := \sum_j \alpha_j c_j \vec{d}_j, \quad \alpha_j > 0,$$

where α_j 's are obtained by solving a dual optimization problem. Those \vec{d}_j such that α_j is greater than zero are called support vectors, since they are the only document vectors contributing to \vec{w} . Classification of test instances consists simply of determining which side of \vec{w} hyperplane they fall on. Also, note that SVM has inherently a linear decision boundary and hence the natural formulation of SVMs lends itself well for a 2 class classification problem, like ours. Random Forest is intrinsically suited for multiclass problems, while SVM is intrinsically two-class. But we give Random forest a try too.

EVALUATION

SVM	Precision	Recall	F1	# of Documents
Positive Documents	0.58	0.64	0.61	76
Negative Documents	0.60	0.53	0.56	75
Average	0.59	0.59	0.59	151

Random Forest	Precision	Recall	F1	# of Documents
Positive Documents	0.54	0.80	0.65	76
Negative Documents	0.62	0.32	0.42	75
Average	0.58	0.56	0.54	151

FEATURE EXTRACTION EXPERIMENTS

- We attempted one more feature generating method named CountVectorizer. This converts a collection of text documents to a matrix of token counts. This implementation produces a sparse representation of the counts. But the results did not improve considerably.
- Tried working with different parameters for random forest. We tried max_feature and bootstrap parameters and achieved a small upgrade in F1 results to 58% which is no better than SVM.

SUMMARY

Overall, we see that SVM and Random Forest give slightly better results than Naïve Bayes and that was expected, given SVM's larger margin and its linear decision boundary. Though the performance improved, however, the improvement is not enough to be considered 'reliable' when it comes to classifying sentiments. In summary, key learnings for SVM & Random Forests:

- + No conditional independence assumption like in Naïve Bayes, easy to implement
- Accuracy still not high for different training-test domains, possibly due to overfitting

RECURRENT NEURAL NETWORK (RNN) WITH LSTM UNITS

Having tried the few most popular Sentiment Analysis techniques in the previous sections, we notice that we barely see any good increase in accuracy beyond 50-60%. This forces us further to look for something more advanced or more intuitive to capture the true the nature of sentiments in text. Also, it's important to note that while NB and SVM gave pretty good results with most researchers^{[1][3]}, it did not in our case. This is primarily due to the fact that, our training and test data are from different domains and that's what makes our experiments interesting. Hence, we moved to another advanced technique which has proven to show good results in complex domains, namely, **Recurrent Neural Network**.

The unique aspect of NLP data is that there is a **temporal** aspect to it. Each word in a sentence depends a lot on what came before and comes after it. In order to account for this dependency, a recurrent neural network seems to be a perfect fit. In RNNs, each word in an input sequence will be associated with a specific time step. Associated with each time step is also a new component called a hidden state vector h_t . The hidden state is a function of both the **current word vector** and **the hidden state vector at the previous time step**:

$$h_t = \sigma(W^H h_{t-1} + W^X x_t)$$

The 2 W terms in the above formulation represent weight matrices. When the magnitude of W^H is large and the magnitude of W^X is small, we know that h_t is largely affected by h_{t-1} and unaffected by x_t .

Long Short Term Memory Units (LSTMs)

Long Short Term Memory Units are modules that can be inside recurrent neural networks. More generally, they make sure that the hidden state vector h is able to **encapsulate** information about **long term dependencies in the text**. As we saw in the previous paragraph, the formulation for h in traditional RNNs is relatively simple and won't be able to effectively connect together information that is separated by more than a couple time steps. Hence, the addition of LSTM units make it possible to determine the correct and useful information that needs to be stored in the hidden state vector.

Problem Formulation

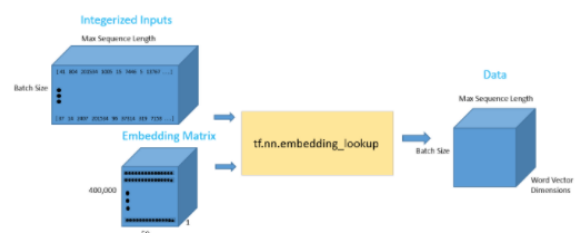
As mentioned above, the task of sentiment analysis involves taking in an input sequence of words and determining whether the sentiment is positive or negative. We can separate this into different components:

- Training a word vector generation model (such as Word2Vec) or loading pre-trained word vectors.
- Creating a matrix representation for our training set
- RNN (with LSTM units) graph creation
- Training the model
- Testing

METHODOLOGY:

Load and Integerize Data:

First, we created our Word Vectors, for which, we used vectors trained using GloVe^[12] dataset. The dataset matrix had 400,000 word vectors, each with a dimensionality of 50. We use this to vectorize all our training data, with a pipeline as shown below for each sentence and review.



We load the 25K reviews IMDB movie training set and integerize it to get a 25000 x 300 matrix (taking reviews of max length 300 words).

Build RNN Model:

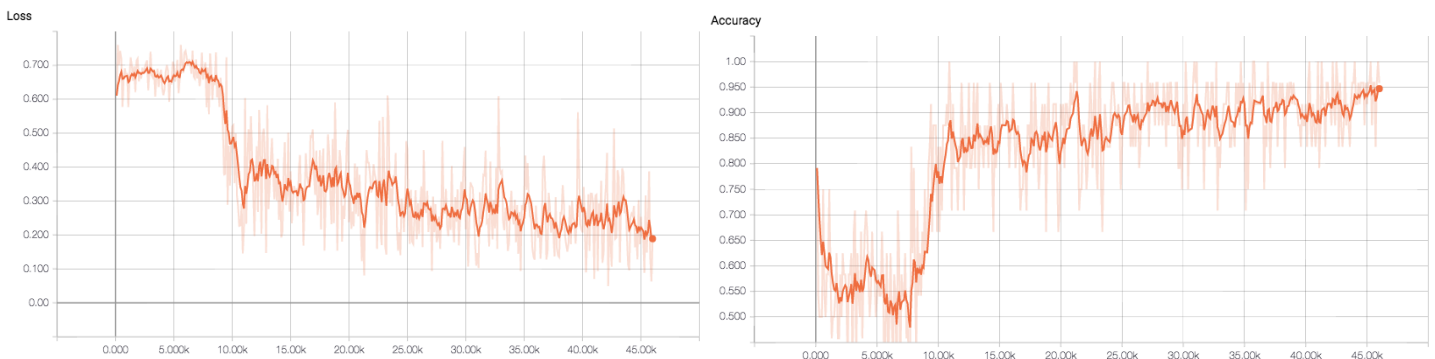
We used the Tensorflow^[13] library to build our RNN; as with all Tensorflow graphs, we specified two placeholders, one for the inputs into the network, and one for the labels. The labels placeholder represents a set of values, each either [1, 0] or [0, 1], depending on whether each training example is positive or negative. Once we had our input data placeholders, we call the `tf.nn.lookup()` function in order to get our word vectors. The call to that function returned a 3-D Tensor of dimensionality batch size by max sequence length by word vector dimensions. Once we have the data in the format that we like, we called the `tf.nn.rnn_cell.BasicLSTMCell()` function, which takes in an integer for the number of LSTM units that we want. Then we wrap that LSTM cell in a dropout layer to help prevent the network from overfitting. For simplicity, we do not stack any LSTM units and the first output of the dynamic RNN function can be thought of as the last hidden state vector. This vector will be reshaped and then multiplied by a final weight matrix and a bias term to obtain the final output values.

HYPERPARAMETER TUNING & EVALUATION

Choosing the right values for our hyper-parameters is a very important part of training deep neural networks effectively. We found that our training loss curves can vary considerably with our choice of optimizer (Adam, Adadelata, SGD, etc), learning rate, and network architecture. We tabulated our few of our top tuning attempts:

Optimizer	Learning Rate	Iterations	Average Accuracy of Test Batches	Peak Accuracy in Test Batches
Adam	0.1	30000	62%	69%
Adam	0.01	10000	67%	71%
Adam	0.01	50000	62%	70%
Gradient Descent	0.1	70000	64 %	75%
Gradient Descent	0.1	50000	72%	79%
Gradient Descent	0.2	50000	68%	79%

As seen above, we found that for the hand-tagged **151 documents** we tested, the best performing model with **Gradient Descent** optimizer on a **learning rate of 0.1**, the loss and accuracy plot for the same is as below:



SUMMARY

Overall, we see that RNN/LSTM give the best results among the language models we tried and can be used as an indicator for the document sentiment. In summary, key learnings for this technique are:

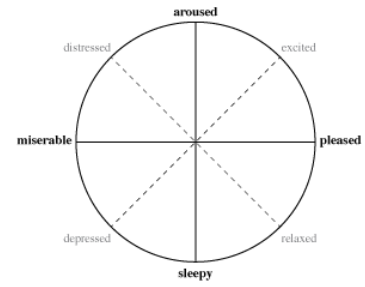
- + High accuracy, RNN+LSTM intuitive choice to capture long dependencies
- + Captures context and higher degree relationships
- Larger risk of overfitting when training and testing are done on different datasets (like ours)
- Very High Training time and computationally extensive

VALENCE & AROUSAL MODEL

Computational methods to estimate sentiment, like few of the ones we evaluated before this section (NB, SVM, RNN), perform a concept-level analysis of the natural language text. One important requirement for these traditional approaches is sufficient *high-quality annotated text* to allow for accurate natural language evaluations. Some researchers argue that this is not necessarily available in short text snippets like tweets, instant messages or emoticons, nor is it feasible in newer domains (like ours) to hand-annotate large corpora of data. Hence, many researchers have suggested an alternative method to the use of **affect dictionaries** that report the sentiment of a set of words along one or more emotional dimensions. We decided to give one of these methods a try, in an attempt to find more effective ways to classify sentiment. Examples of sentiment dictionaries include POMS^[8] and POMS-ex-Profile of Mood States and ANEW-Affective Norms for English Words.

Representing Emotion: Russell's Model of Emotional Affect^[11]

In psychology, emotional models have been proposed to define and compare emotional states which often use emotional dimensions to position emotions on a 2D plane. A simple model proposed in 1980 by J. A. Russell^[11] represents pleasure along a horizontal axis, with highly unpleasant on one end, highly pleasant on the other, and different levels of pleasure in between. He proposed using valence (or pleasure) and arousal (or activation) to build an **emotional circumplex of affect**. Russell applied multidimensional scaling to position 28 emotional states, producing the model shown to the left with valence running along the horizontal axis and arousal along the vertical axes.



Based on Russell's affect model, we decided to use the ANEW, an [extended ANEW dictionary](#)^[6] that was recently built by researchers at McMaster to evaluate the valence and arousal of text. Our sentiment dictionary provides measures of valence and arousal for approximately 13,915 English words. Each word is rated on a nine-point scale ranging from 1 to 9. Words included in the dictionary were selected from the above research that identified them as good candidates to convey emotion. For example, to construct the ANEW dictionary, volunteers were asked to read a text corpus and provide a rating along each dimension for each occurrence of an ANEW-recognized word. Ratings for a common word are combined into a mean rating and a standard deviation of the ratings for each dimension.

Following the research paper (^[6]Warriner et. al.), we do the following to classify a document:

1. Calculate the mean valence and arousal of each sentence
 - a. Identify all words in the sentence that are in ANEW dictionary
 - b. Calculate a sentence-average using probabilistic density function (on the mean of each word)
2. Perform a simple arithmetic mean on each sentence's mean valence and arousal values.
3. If the new average valence is greater than 5, mark the document **positive**. Else, mark it **negative**.

Below are some equations that help summarize the above discussion:

$$(M_v, M_a)_{sentence} = \frac{\sum_{v,a} PDF(mean) * mean}{\sum_{v,a} PDF(mean)}$$

$$(M_v, M_a)_{doc} = mean(M_v, M_a)$$

$$Sentiment(doc) = \begin{cases} Positive & , \text{ if } M_v > 5 \\ Negative & , \text{ otherwise} \end{cases}$$

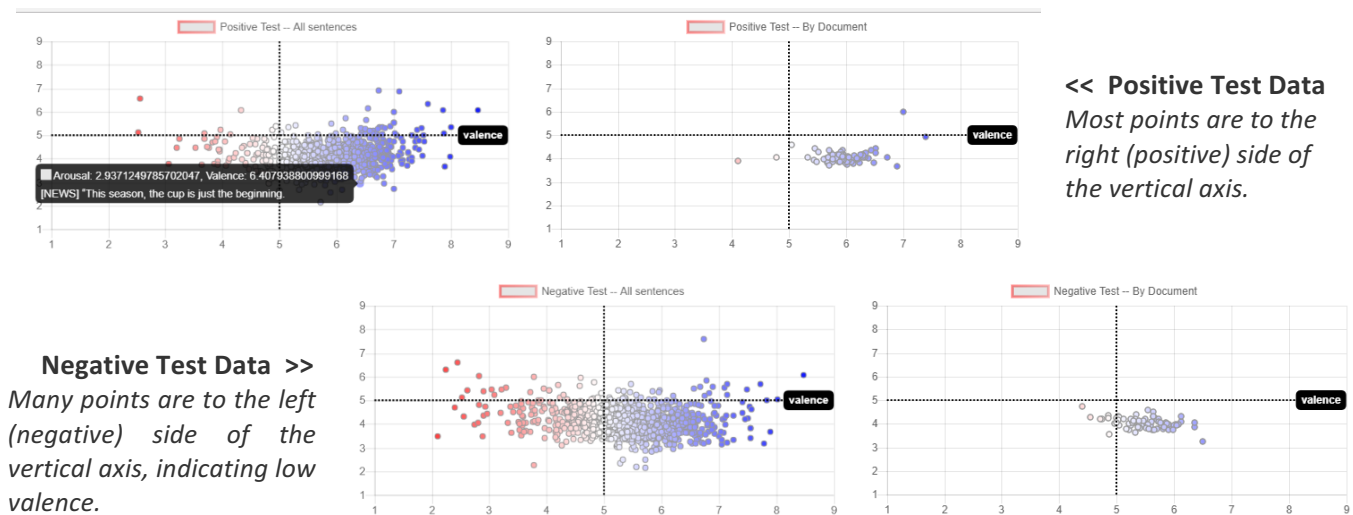
EVALUATION

We ran the classifier on the 150 documents from the 2 domains: News and Literature and we observed the following results, although, intrinsic evaluation like below is not required for this model (since its Linguistic):

# of Documents	Predicted Positive	Predicted Negative
Actual Positive	74 (TP)	2 (FN)
Actual Negative	66 (FP)	9 (TN)

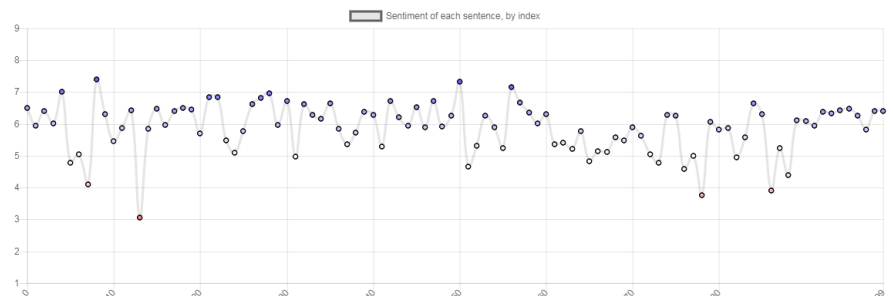
Valence & Arousal Model	Positive Sentiment Test Documents	Negative Sentiment Test Documents
PRECISION	53%	82%
RECALL	97%	12%
F1	69%	21%
ACCURACY	55%	55%

It should be noted that, for this model, intrinsic measures such as Precision, Recall etc. may not be indicative as this method is aimed or designed to be qualitatively evaluated. Hence, for this model, it makes a lot more sense to look at the **Valence and Arousal plot** for the Positive and Negative sentiment tests as shown below:



We can clearly see that the classification works pretty well for most part and gives a nice spread of words across the valence axis. Also note that this view also enables the user to view outliers directly by hovering over the said sentences and words. Another useful evaluation metric is to see the **flow of sentiments** and if it corresponds to the text, as shown below:

Sentiment Flow >
Shows the flow of sentiments (beginning to end) shown as left-to-right to give a perspective of forward motion of ideas and text.



FEATURE EXTRACTION EXPERIMENTS

We tried some tweaking of features as listed below and noted their effects on the outcome/accuracy:

- Using a window of [-1, 0] (checking the previous word), we looked for any negation tokens like “n’t”, “not”, “no”, “never”, etc. When we encountered a negation token in that window, we reverse the valence value of the word token we’re looking at. So, if the valence of ‘good’ is 7, and we encounter ‘not good’, we treat that as a valence of 3. This did not change the results on our test set considerably.
- We also tried weighting words valence values by their frequency in our training set. That is, we had the intuition that a word in ANEW with high frequency (more occurrences) would be more consistent at predicting a sentence’s sentiment than a word with lower frequency in ANEW. This, however, ended up hurting our results (all metrics) instead of improving them.

SUMMARY

Overall, we can note that while the intrinsic evaluations do not yield a very high result for this technique, it can be observed that the Valence and Arousal model does give some good idea into the overall sentiment of the text. It is much easier to understand, interpret and is also free of user-prompting, i.e. that the user is free to interpret the output plots as per their own requirements and there is no hard-definition of what is “positive” or “negative” sentiment. Having said that, the key learnings for this technique are:

- + No Training!
- + Less specificity to domains of data, more generic and tied to linguistics.
- + Qualitative Assessment, open to user interpretation, no hard decision boundary.
- Misses some contextual and structural clues (i.e. grammar, POS etc.).

OUTPUT & USER INTERFACE

The project code is hosted at <https://github.com/biswarajkar/storynext2> and can be accessed using a web browser. The User Interface of the project and the instructions to view the same can be accessed from the URL above as the interface might be updated regularly.

NEXT STEPS

As next steps of this project, these are some of our ideas:

1. Implementing ‘auto-play’ feature which will plot the sentiments automatically in a timeline of the text (beginning to end) so that the user can see the sentiments and words grow with the writing/article.
2. Using MPQA’s (<http://www.aclweb.org/anthology/N/N15/N15-1146.pdf>) opinion corpus’s affect based methods^{[9][10]} instead of B-/Tri-gram language models.
3. Using Topic Detection based on some clustering techniques to further enhance the word-cloud to show word-sentiment clusters; thereby highlighting what words or group of words have large influence on changing the emotion of the whole sentence or the article.
4. Linking words on the word cloud and the points on the plots back to the text, enabling users to click on the plots and see exactly where in text that feature appears.

REFERENCES

Technique/Methodology References:

- [1] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10 (EMNLP '02), Vol. 10. Association for Computational Linguistics, Stroudsburg, PA, USA, 79-86. DOI: <https://doi.org/10.3115/1118693.1118704>
- [2] C. Yang, K. H. Y. Lin and H. H. Chen, "Emotion Classification Using Web Blog Corpora," Web Intelligence, IEEE/WIC/ACM International Conference on, Fremont, CA, 2007, pp. 275-278.
- [3] Christopher D. Manning and Hinrich Schütze. 1999. Foundations of Statistical Natural Language Processing. MIT Press, Cambridge, MA, USA.
- [4] Peter D. Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL '02). Association for Computational Linguistics, Stroudsburg, PA, USA, 417-424. DOI: <https://doi.org/10.3115/1073083.1073153>
- [5] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 conference on empirical methods in natural language processing (pp. 1631-1642).
- [6] Norms of valence, arousal, and dominance for 13,915 English lemmas. Warriner AB, Kuperman V, Brysbaert M. Behav Res Methods. 2013 Dec;45(4):1191-207. DOI: 10.3758/s13428-012-0314-x. PMID: 23404613
- [7] Dodds PS, Harris KD, Kloumann IM, Bliss CA, Danforth CM (2011) Temporal Patterns of Happiness and Information in a Global Social Network: Hedonometrics and Twitter. PLoS ONE6(12): e26752. <https://doi.org/10.1371/journal.pone.0026752>
- [8] L. Curran, Shelly & A. Andrykowski, Michael & Studts, Jamie. (1995). Short form of the Profile of Mood States (POMS-SF): Psychometric information. Psychological Assessment. 7. 80-83. 10.1037/1040-3590.7.1.80.
- [9] Yoonjung Choi and Janyce Wiebe (2014) +/-EffectWordNet: Sense-level Lexicon Acquisition for Opinion Inference, Proc. of EMNLP 2014.
- [10] Janyce Wiebe, Theresa Wilson , and Claire Cardie (2005). [Annotating expressions of opinions and emotions in language](#). *Language Resources and Evaluation*, volume 39, issue 2-3, pp. 165-210.
- [11] Russell, J. A. (1980). A circumplex model of affect. Journal of Personality and Social Psychology, 39(6), 1161-1178. <http://dx.doi.org/10.1037/h0077714>
- [12] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. [GloVe: Global Vectors for Word Representation](#)
- [13] Tensorflow: An open-source software library for Machine Intelligence. <https://www.tensorflow.org/>
- [14] IMDB11 Dataset: 25k Movie Reviews from IMDB. <http://ai.stanford.edu/~amaas/data/sentiment/>