JANUARY 9, 2019

ILLINOIS INSTITUTE OF TECHNOLOGY

RUSSELL CONJUGATION

CS 597: READING & SPECIAL PROBLEMS

PRAGYA MISHRA
ILLINOIS INSTITUTE OF TECHNOLOGY
A20406346

Contents

Objective: 3
Background [1]: 3
Workflow Summary: 3
Step 1: Text Pre-processing 4
Step 2: Identify emotive predicates: 5
Step 3: Identify Russell Conjugates: 5
Method 1: Using Dictionary based Word Sense Disambiguation 5
Method 2: Using Probabilistic language model – N-grams 7
Method 3: Using Word embeddings – Word2vec [9] [10] 8
Comparison of all methods: 10
Result Snapshots: 10
Step 4: Present the identified Russell conjugates as alternatives for emotive predicates 11
Future Research Plan 12
References 13
Appendix 1 14
Programming languages: 14

Objective:

To build and deploy a prototype bias-revealing browser plugin, which will reveal hidden sources of emotive bias in online rhetoric.

Background [1]:

The use of subtly emotive language can bias interpretation of otherwise objective and accurate characterizations of people and events. Speechwriters and rhetoricians have used careful word choice to good effect since time immemorial. Bertrand Russell memorably encapsulated the idea in pseudo-conjugations such as:

- I am firm, you are obstinate, he is a pig-headed fool.
- I am righteously indignant, you are annoyed, he is making a fuss over nothing.
- I have reconsidered the matter, you have changed your mind, he has gone back on his word.

Here, pairs of words such as 'firm', 'obstinate' and 'pigheaded' are known as **Russell or Emotive Conjugates** of each other. As rhetoric in all media, both political and non-, has become increasingly polarized, so has, it seems, the use of such emotive language to pre-emptively destroy one's opponents and prop-up one's heroes.

The long-term goal of this project is to investigate to the extent to which Russell conjugations are used to bias rhetoric, to develop tools to make readers aware of such rhetorical tricks, and to investigate how such tools affect readers' perceptions of bias and evaluation of information. The scope of the current project is to build a browser plug-in that automatically

a) Identifies source of polarization in a rhetoric, i.e., emotive words,
b) Identify and present Russell Conjugates of these emotive words as an option to the user.

Workflow Summary:

The key idea of this project is to identify adjectives, verbs, nouns or phrases with positive or negative emotional bias and present their Russell conjugations as alternative readings within the text. The first step towards that involves finding the emotive predicate. This in turn involves filtering each predicate using a sentiment lexicon. Next step in this process would be to find Russell Conjugates of these emotive predicates. For this, we would need to find set of words that all:

a. Have the same part of speech
b. Have approximately the same denotational meaning,
c. Have differing emotional values, and
d. Are used in the same or similar textual contexts.

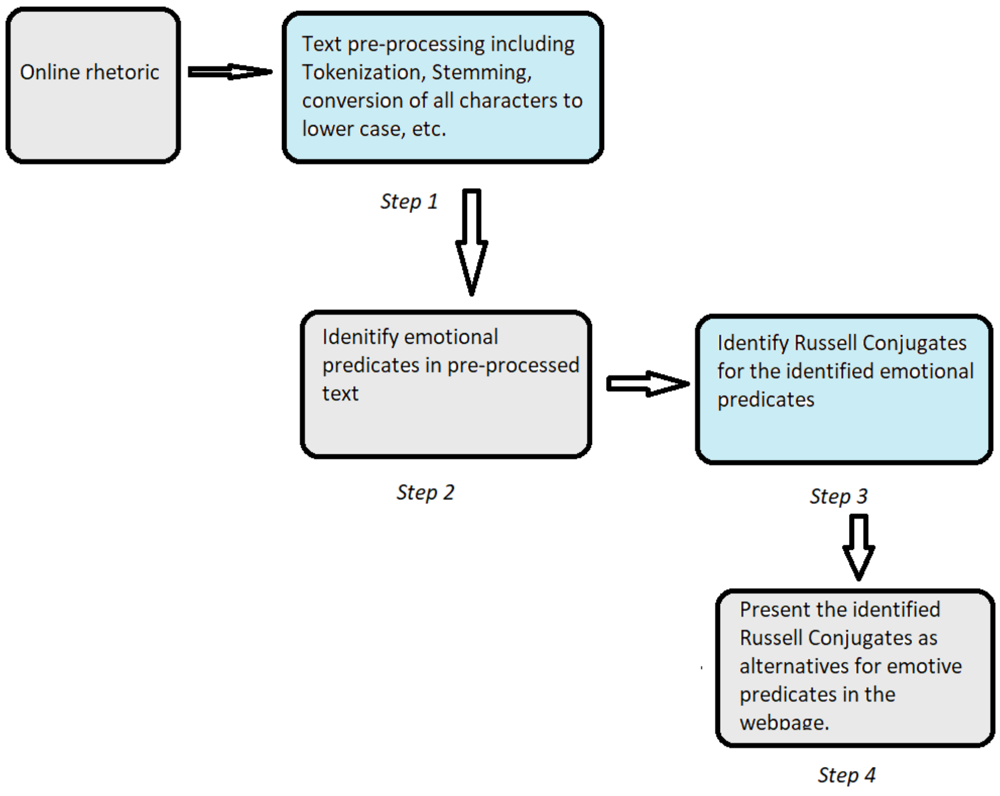

Online rhetoric
Text pre-processing including Tokenization, Stemming, conversion of all characters to lower case, etc.
Step 1
Idenitify emotional predicates in pre-processed text
Step 2
Identify Russell Conjugates for the identified emotional predicates
Step 3
Present the identified Russell Conjugates as alternatives for emotive predicates in the webpage.
Step 4

Figure 1: High Level Work flow

Step 1: Text Pre-processing

Text pre-processing [2] is the first step towards text analysis and involves preparing the text such that further processing can be done on it. We perform tokenization, normalization and POS tagging here.

Tokenization: Tokenization is a process which splits longer strings of text into smaller pieces, or tokens. Larger chunks of text can be tokenized into sentences, sentences can be tokenized into words, etc. [3]

Normalization: Normalization refers to a series of related tasks meant to put all texts on a level playing field and allows processing to proceed uniformly. Some of the normalization tasks used here are normalizing word formats using regular expression and Lemmatization (Reduce inflections or variant forms to base form).

POS tagging: This task assigns the correct part-of-speech to each word (and punctuation) in a text. This will be necessary in our application to understand the part-of-speech of the identified emotive predicates for which the Russell Conjugates would have to be found.

Step 2: Identify emotive predicates:

Emotive predicates here imply the words or set of words that are emotionally biased to have a polarising impact on the reader. The identification of such words through the text is important. This is done by first identifying predicates which in this case (prototype) are adjectives. We can use adverbs/verbs/nouns/phrases in the later versions of this application. Post that, we perform word/feature level sentiment classification using lexicon-based approach [4]. Word/feature level sentiment classification refers to expressing sentiment polarity for each word. Lexicon-based approach relies on a sentiment lexicon, a collection of known and precompiled sentiment terms. The lexicon used here for this purpose is the General Inquirer lexicon.

Step 3: Identify Russell Conjugates:

As discussed earlier, to find Russell Conjugates of the emotive predicates, identified in step 2, we would need to find set of words that all:

a) Have the same part of speech
b) Have approximately the same denotational meaning,
c) Have differing emotional values, and
d) Are used in the same or similar textual contexts.

Multiple approaches were considered and implemented to identify these conjugates:

Method 1: Using Dictionary based Word Sense Disambiguation

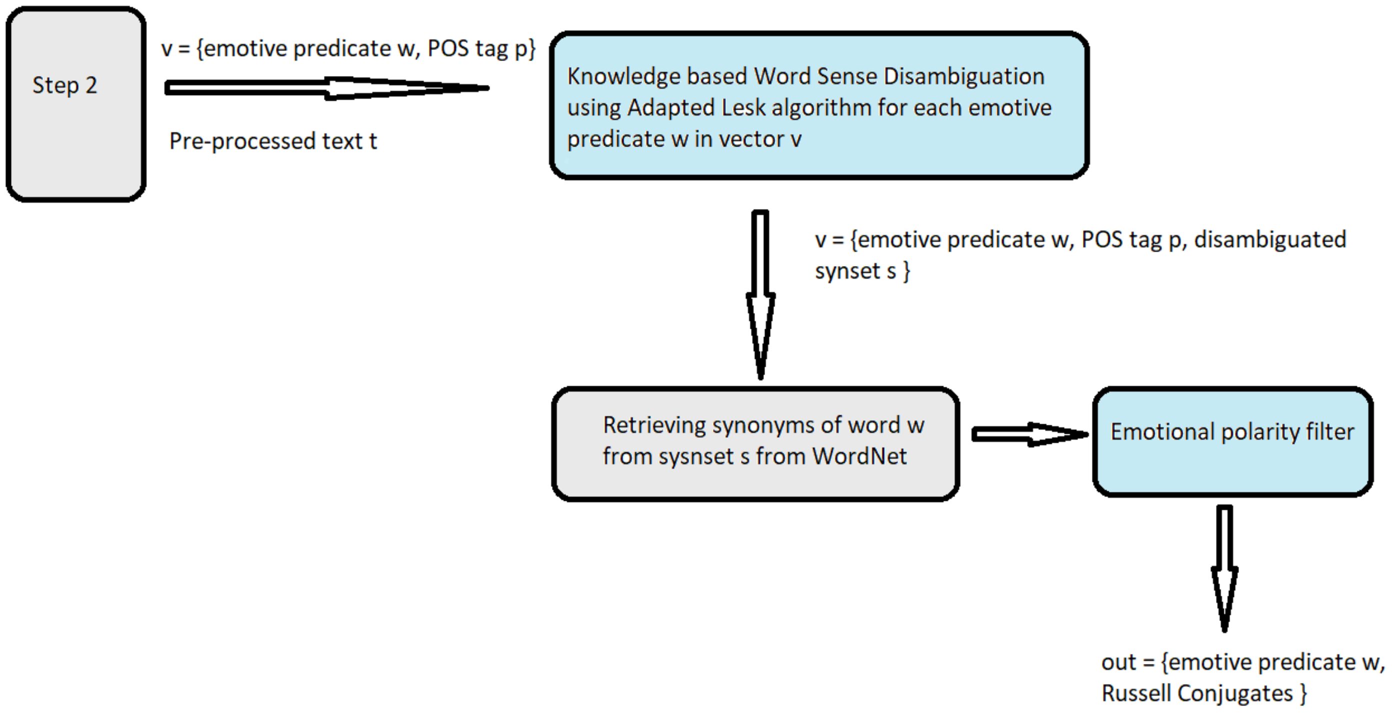

Step 2
v = {emotive predicate w, POS tag p}
Pre-processed text t
Knowledge based Word Sense Disambiguation using Adapted Lesk algorithm for each emotive predicate w in vector v
v = {emotive predicate w, POS tag p, disambiguated synset s }
Retrieving synonyms of word w from sysnset s from WordNet
Emotional polarity filter
out = {emotive predicate w, Russell Conjugates }

Figure 2: Method 1 procedure

Once we have the pre-processed text and identified emotive predicates from step 2, we proceed to identify words with same part of speech, having similar denotational and contexual meaning to the emotive predicate. In method 1, this is done using **Adapted Lesk algorithm** [5] for diambiguating the sense of the emotive predicate and then retreiving the applicable (same sense and POS) synonyms using Wordnet.

Lesk algorithm: The high level idea of the original lesk algorithm is to choose between senses of a word given in a dictionary based on the words in the definitions. Lesk algorithm was introduced by Michael Lesk in his 1986 paper "Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone" [6]. The algorithm is as follows:

- Define $D_i(w)$ as the bag of words in the ith definition for w
- Define E(w) as $\cup_i D_i(w)$
- For all senses s_k of w, do:
    - score(s_k) = similarity($D_k(w)$,[$\cup_{[v \text{ in } c]}$ E(v)])
- Choose s = arg $\max_{sk}$ score(s_k)

The original lesk algorithm disambiguates words in short phrases. The definition, or gloss, of each sense of a word in a phrase is compared to the glossed of every word in the phrase.

But the problem with the original lesk algorithm is that it relies on glosses found in traditional dictionaries such as Oxford Advanced Learner's. Hence, we use a variant of lesk, called the Adapted Lesk, to take advantage of the highly inter-connected set of relations among synonyms that WordNet offers. While lesk algorithm restricts its comparisons to the glosses of words being disambiguated, adapted lesk is able to compare the glosses of the words that are related to the words to be disambiguated. This provides a richer source of information and improves overall disambiguation accuracy. Adapted lesk also introduces a novel scoring mechanism that weighs long sequences of matches more heavily than single words. As mentioned before, the dictionary used here is WordNet.

WordNet [5] is a semantically arranged dictionary, creating an electronic lexical database of nouns, verbs, adjectives and adverbs. Synonyms words are grouped together to form synonym sets, or synsets. A word is polysemous if it occurs in several synsets, where each synset represents a possible sense of the word. The Adapted Lesk algorithm used in this method takes as input an example or instance in which a single target word occurs, and it will output a WordNet sense for that target word based on information about the target word and a few immediately surrounding words that can be derived from WordNet.

In Method 1, as shown in figure 2, word sense disambiguation using adapted lesk algorithm is applied for each emotive predicate using the sentence, in which it appears, as the context. The applicable synonyms are retrieved from the WordNet and are filtered for emotional polarity using sentiment lexicon. The output of this filter are the Russell Conjugates for the emotive predicate w in the given context.

Method 2: Using Probabilistic language model – N-grams

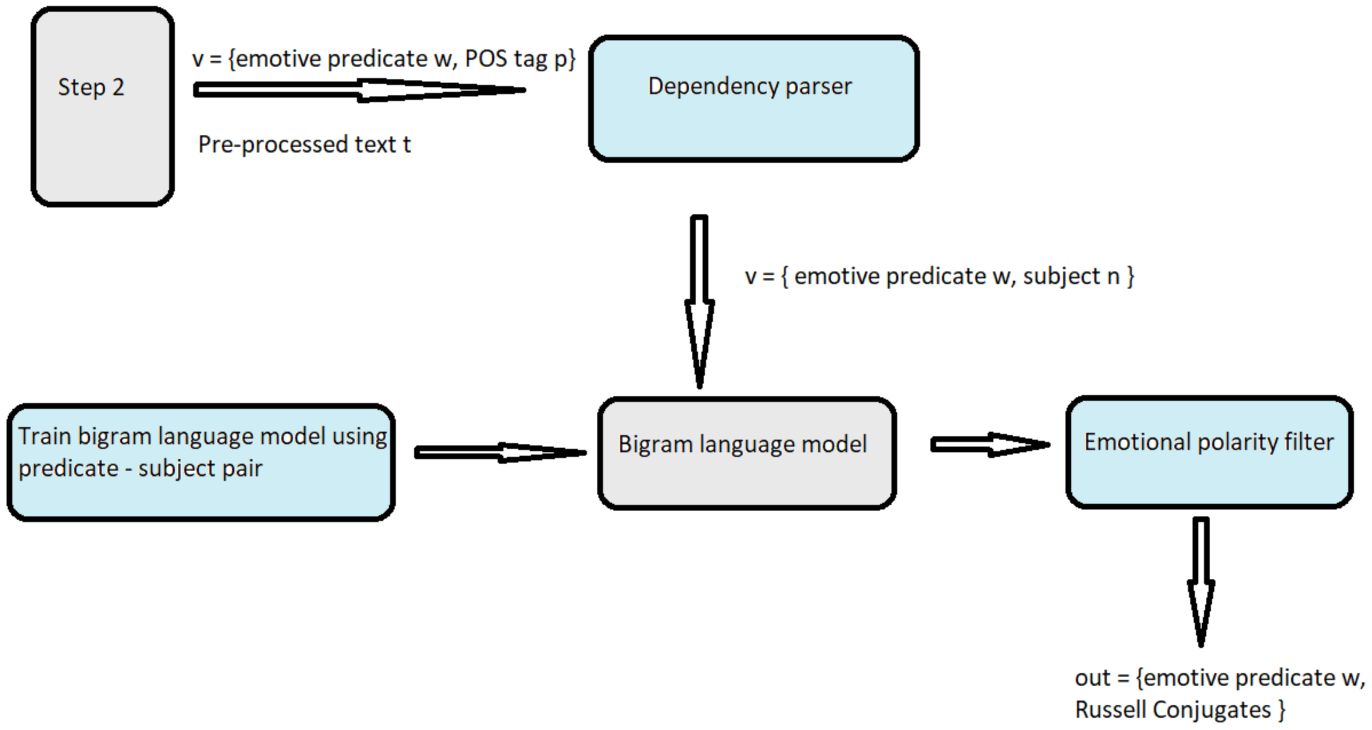

Step 2
v = {emotive predicate w, POS tag p}
Pre-processed text t
Dependency parser
v = { emotive predicate w, subject n }
Train bigram language model using predicate - subject pair
Bigram language model
Emotional polarity filter
out = {emotive predicate w, Russell Conjugates }

Figure 3: Method 2 procedure

Here, we take a different approach from method 1 and use a language model to identify probable alternative predicates as Russell Conjugates. This is done by first using a dependency parser. A parser provides or resolves structural ambiguity in a text in a formal manner. There are two types of parsers, Dependency - which focuses on relations between words and Phase Structure – which focuses on identifying phrases and their recursive structure. Since we need the required output in the form of pairs of emotive predicates w and their subject n, as per the figure 3, we use a **typed dependency parser** on the pre-processed text from step 2 to achieve just that.

Dependency parser [7] – A Dependency parser analyzes the grammatical structure of a sentence, establishing relationships between "head" words and words which modify those heads. In this application, we use the Stanford parser, which is a neural network and transition-based typed dependency parser [8] which accepts word embeddings inputs. This parser builds a parse by performing a linear-time scan over the words of a sentence. At every step it maintains a partial parse, a stack of words which are currently being processed, and a buffer of words yet to be processed. The parser continues to apply transitions to its state until its buffer is empty, and the dependency graph is completed.

After applying dependency parser on the text, we obtain pairs of labeled related words and we extract our emotive predicate and its subject from those pairs as the output. The subject nouns from each pair is then passed to a bigram language model for further processing. Language models are probabilistic models that predict the next word in a sequence of words. An N-gram model [9] is a language model which predicts the next word from the previous N-1 words. An N-gram is a N-token sequence of words: a 2-gram/bigram is a two sequence of words. We built a bigram language model using the news category of BROWN corpus in NLTK package of python. The bigrams used for building this model were filtered to have predicate- subject noun pair, which in this case were adjective-noun pairs. Thus, the model calculated the probability of various adjectives given a noun. Once the subject noun from the dependency parser was passed to the model, the model provided a list of adjectives most probable to be associated with the given noun. These adjectives were then again filtered for emotional valence using General Inquirer lexicon and the output of the same were considered as the Russell Conjugates.

Method 3: Using Word embeddings – Word2vec [10] [11]

The goal of implementing this method was to identify a function f, such that,

f(w) = w's Russell Conjugates

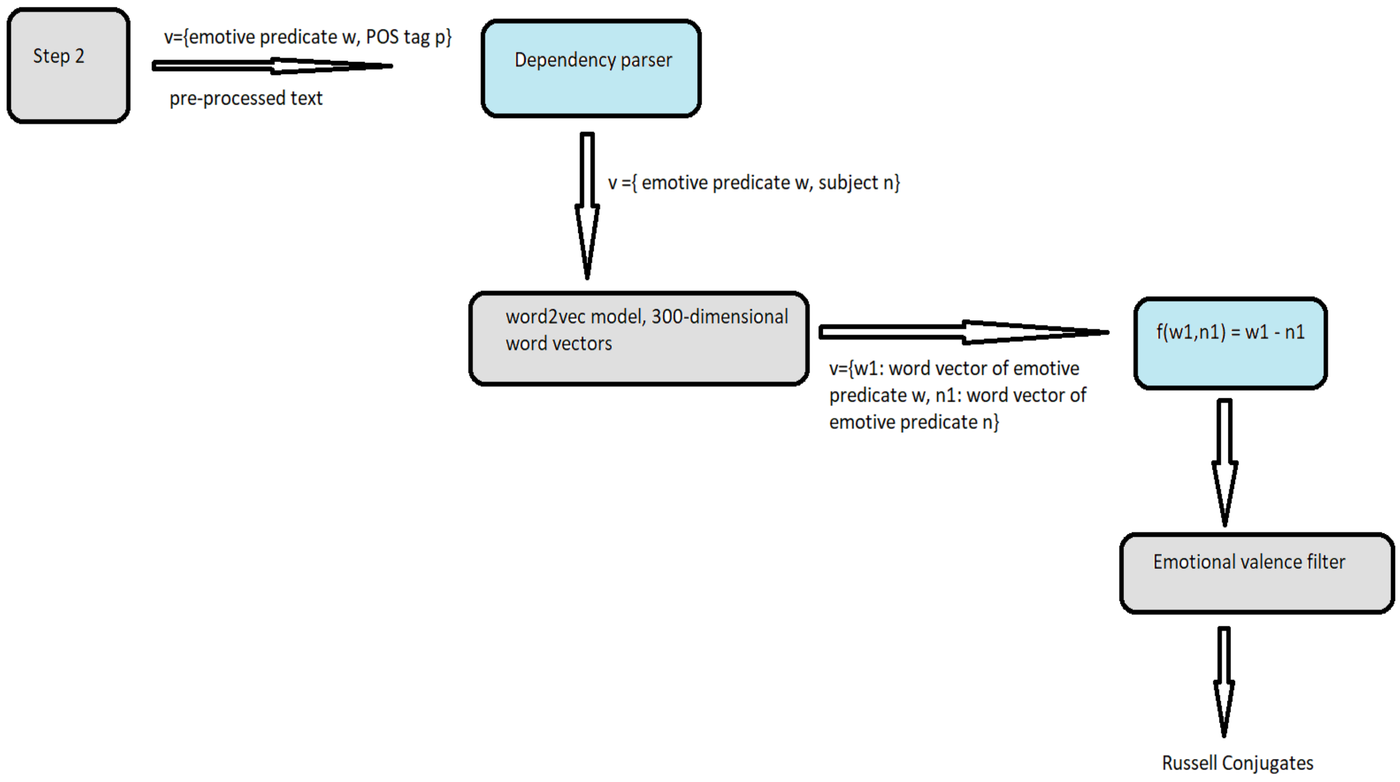

Step 2
v={emotive predicate w, POS tag p}
pre-processed text
Dependency parser
v ={ emotive predicate w, subject n}
word2vec model, 300-dimensional word vectors
v={w1: word vector of emotive predicate w, n1: word vector of emotive predicate n}
f(w1,n1) = w1 - n1
Emotional valence filter
Russell Conjugates

Figure 4: Method 3 procedure

As illustrated in the figure 4, the pre-processed text t is passed through a dependency parser (Stanford parser) to obtain the pairs of emotive predicates and their subject nouns. The procedure for obtaining this is the same as described in the previous method, using the Stanford parser which is a transition-based typed dependency parser. The output pairs are converted into their respective word vectors using Google's Word2vec.

Word2vec is a framework, introduced by Google in 2013, for learning word vectors. The idea behind learning word vectors is to have a large corpus of text for training the vectors. Initially, every word in a fixed vocabulary is represented by a vector. Iterate over each position t in the text, which has a center word c and context (outer/outside) words o and then use the similarity of the word vectors for c and o to calculate the probability of o given c (or vice versa) keep adjusting the word vectors to maximize this probability [12].

There are two variants of the word2vec framework [12]:

1. Skip-grams (SG): The word vectors are trained by maximizing the probability of context words/ outer words "o" given the center word "c"
2. Continuous Bag of Words (CBOW): The word vectors are trained by maximizing the probability of center word "c" given the context/outer words "o"

Let us consider the procedure of training word vectors using skip-gram framework:
For each position t= 1,2,3…,T in the corpus text, we would predict context words within a window of fixed size m, given center word w_t.

$$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \ne 0}} P(w_{t+j} \mid w_t; \theta)$$

$L(\theta)$ is the likelihood function and θ represents all the parameters that need to be optimized. If our vocabulary consists of V number of words and we plan to train N-dimensional vectors, θ will be a matrix of 2DV. This is excluding other parameters like m (size of the context window or any weights given to them). The multiplication by 2 is because each word in the vocabulary will be have two vectors, one as the center word and one as the context word. Having two vectors for each words helps in easier optimization and the resultant to vectors can be averaged at the end for the final vector of that word. The cost function is the negative log likelihood:

$$J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \ne 0}} \log P(w_{t+j} \mid w_t; \theta)$$

Stochastic gradient descent will be used to optimize θ such that $J(\theta)$, cost function, minimizes. The prediction function is calculated in softmax form giving:

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

The numerator in the above equation is a dot product that compares similarity of o and c. Larger the dot product, larger the probability. The denominator, on the other hand, is used for normalizing over the entire vocabulary. Since $J(\theta)$ is the cost function over all windows in the corpus, using gradient descent would be very expensive to compute. The system will take a long time to make even a single update. The solution for this problem will be to use Stochastic gradient descent (SGD) with negative sampling as mentioned in the paper "Distributed Representations of Words and Phrases and their Compositionality" (Mikolov et al. 2013) [11].

Once we have trained our model to generate word vectors, we convert the emotive predicate-subject noun pair, received from the dependency parser, into their respective word vectors for further processing. The next step is identifying a vector f, such that f (w) = Russell Conjugate of w. At this prototype stage, this vector was identified using trial and error on a few combinations of existing Russell Conjugate pairs. The vector f is:

f(w)= v = word vector (emotive predicate w) – word vector (subject noun n)

It can be demonstrated to show that words closest in the vector space to the resultant vector v can be considered as Russell Conjugates of the emotive predicate w in the context of subject noun n. Of course, before the final conjugates are displayed, they would have to be filtered according to their emotional valence using a lexicon-based approach (General Inquirer).

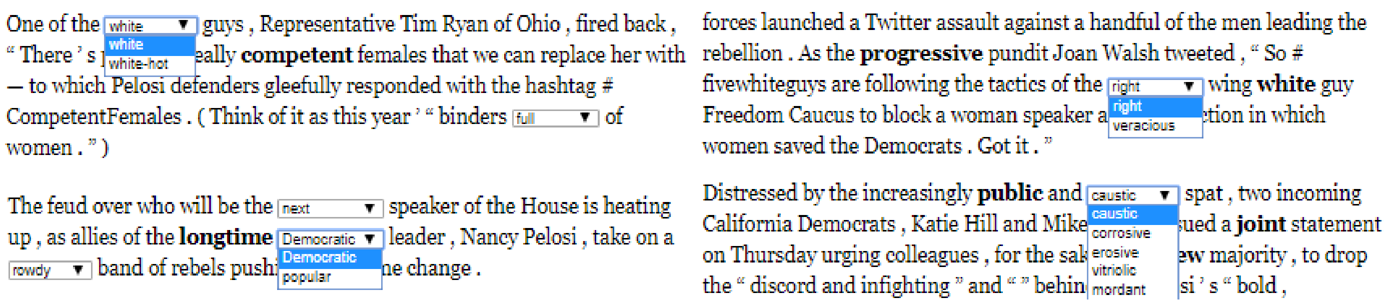
Comparison of all methods:

All the three methods were implemented on 6 news websites. It is observed that Method 3 (word2vec approach) has had the best results among the three.

Sr No.	Factor	Method 1 (Dictionary based WSD)	Method 2 (N-gram LM)	Method 3 (word embeddings)
1	Russell Conjugates with similar denotational meaning	~80%	<10%	~55%
2	Russell Conjugates with similar contextual meaning	<30%	~60%	~50%
3	Overall Accuracy	<30%	<10%	~50%
4	Run-time speed	Fast	Moderate	Low

Result Snapshots:

<u>Method 1:</u>


One of the white guys , Representative Tim Ryan of Ohio , fired back , " There ' s r eally **competent** females that we can replace her with — to which Pelosi defenders gleefully responded with the hashtag # CompetentFemales . (Think of it as this year ' " binders full of women . ")

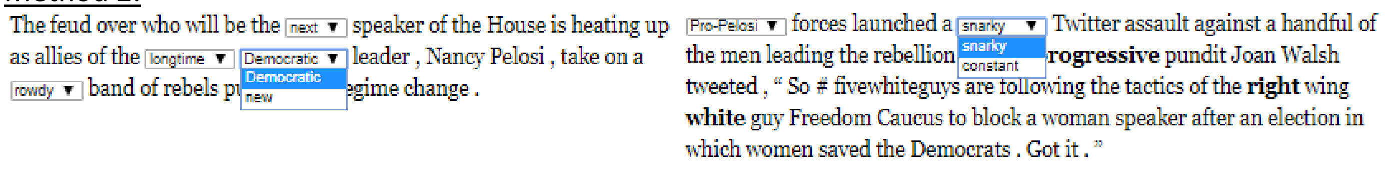
forces launched a Twitter assault against a handful of the men leading the rebellion . As the **progressive** pundit Joan Walsh tweeted , " So # fivewhiteguys are following the tactics of the right wing **white** guy Freedom Caucus to block a woman speaker a tion in which women saved the Democrats . Got it . "

The feud over who will be the next speaker of the House is heating up , as allies of the **longtime** Democratic leader , Nancy Pelosi , take on a rowdy band of rebels pushi ne change .

Distressed by the increasingly **public** and caustic spat , two incoming California Democrats , Katie Hill and Mike sued a **joint** statement on Thursday urging colleagues , for the sak ew majority , to drop the " discord and infighting " and " " behin si ' s " bold ,

Figure 5: Results of Method 1

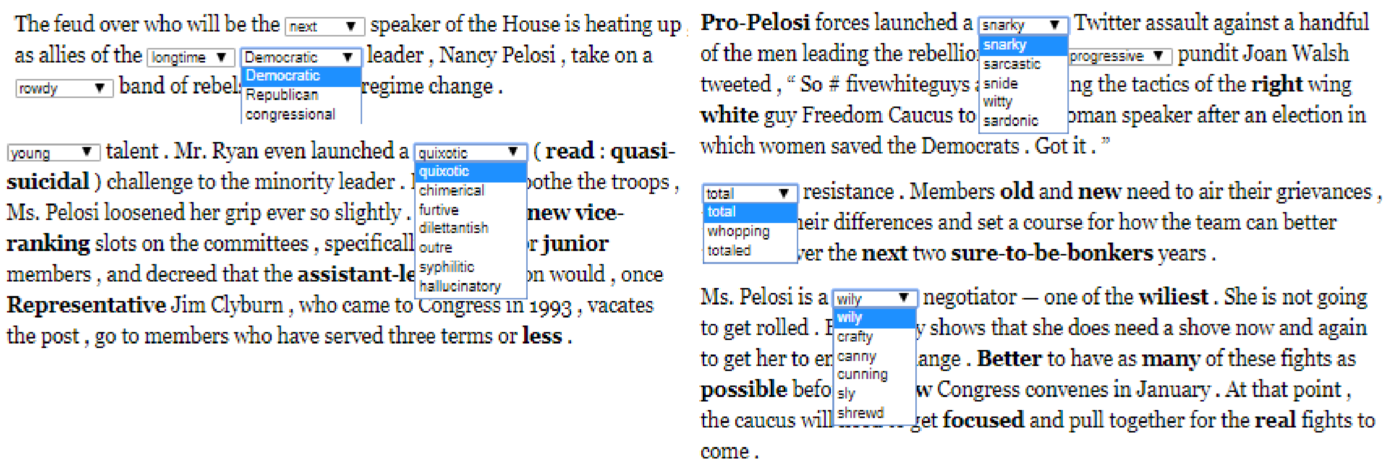
Method 2:


The feud over who will be the next speaker of the House is heating up as allies of the longtime Democratic leader , Nancy Pelosi , take on a rowdy band of rebels pu egime change .

Pro-Pelosi forces launched a snarky Twitter assault against a handful of the men leading the rebellion rogressive pundit Joan Walsh tweeted , " So # fivewhiteguys are following the tactics of the **right** wing **white** guy Freedom Caucus to block a woman speaker after an election in which women saved the Democrats . Got it . "

Figure 6: Results of Method 2

Method 3:


The feud over who will be the next speaker of the House is heating up as allies of the longtime Democratic leader , Nancy Pelosi , take on a rowdy band of rebel regime change .

Pro-Pelosi forces launched a snarky Twitter assault against a handful of the men leading the rebellio progressive pundit Joan Walsh tweeted , " So # fivewhiteguys ng the tactics of the **right** wing **white** guy Freedom Caucus to oman speaker after an election in which women saved the Democrats . Got it . "

young talent . Mr. Ryan even launched a quixotic (**read** : **quasi-suicidal**) challenge to the minority leader . othe the troops , Ms. Pelosi loosened her grip ever so slightly . **new vice-ranking** slots on the committees , specificall r **junior** members , and decreed that the **assistant-le** n would , once **Representative** Jim Clyburn , who came to Congress in 1993 , vacates the post , go to members who have served three terms or **less** .

total resistance . Members **old** and **new** need to air their grievances , eir differences and set a course for how the team can better er the **next** two **sure-to-be-bonkers** years .

Ms. Pelosi is a wily negotiator — one of the **wiliest** . She is not going to get rolled . y shows that she does need a shove now and again to get her to en ange . **Better** to have as **many** of these fights as **possible** befo w Congress convenes in January . At that point , the caucus will et **focused** and pull together for the **real** fights to come .

Figure 7: Results of Method 3

Step 4: Present the identified Russell conjugates as alternatives for emotive predicates

Once the Russell Conjugates for emotive predicates are identified in Step 3, the emotive predicates will be highlighted in the web page and the conjugations would be displayed in one of several ways [1]:

- Click on the element to see a list of alternative words to replace the given word by;
- Show the elements inline as an alternation, e.g.;

  ...controversial/embattled/argumentative Professor Jordan Peterson...
- Rewriting the page with a random choice for each such word – the user can refresh the page to see different choices (or be given a button to do so).

Future Research Plan

The plan for this project is to further research on Method 3 (using word2vec) to automatically identify Russell Conjugates. The research is planned to happen in five phases:

Phase 1: Literature search and creation of training dataset:

1. Literature Search: The idea would be to understand any work that has been done on or around this project, so that we can ensure that our work builds on previous work and not unintentionally duplicate it. This can be done using Google Scholar.
2. Creating Training Dataset: This training dataset would contain pairs of words that are Russell Conjugates, along with their emotional connotations. The expected size of this dataset should be at least 1000 pairs of words. It needs to be created objectively without keeping any procedure in mind. Few approaches for building the dataset:
    a. Take examples that Bertrand Russell gave.
    b. Take other examples available on the internet.
    c. Examine each of the above collected pairs to find other conjugates to them.
    d. Try coming up with a list of at least 100 words and then use WordNet or other thesaurus to identify their conjugates.
    e. Search words (using n-grams for example) that commonly co-occur with the above list of words. Then find other words that co-occur with these co-occurring words. Filter these other words to see which qualifies to be a conjugate. For example, we want to find conjugate for 'controversial'. The words that co-occur with it include 'politician'. The words that co-occur with 'politician' could include 'popular'. Manually identify if 'popular' could be a conjugate for 'controversial'.

    This training set will be reviewed by the team to ensure it is as clean a dataset as possible.

Phase 2: Build software for automatic training and evaluation procedures:

1. Training procedure:
    a. The simplest training procedure, that we can begin with, can be to find the average distance vectors between each pairs of Russell conjugates.
    b. For a more a sophisticated approach, we can do the following. For every word in the training dataset created in phase 1, generate a pair of words that are not Russell conjugates but are a decent alternative. This will be key. These new pair of words will act as bad examples, which means they would look plausible to pass as conjugates, but they should not be. This will lead to us having positive and negative examples, and then the idea would be to find a vector in the 300-dimensional space that is as close as possible to the positive examples and as far away as possible from the negative examples. Support Vector Machines or a similar procedure can be used to identify this target vector.
2. Evaluation procedure: The evaluation procedure decided for this project must be well documented before its usage to have a clear protocol. Types of evaluations that can be done:

a. Cross-validation: the output here will be more in the form of something similar to R-squared or an average distance of the predicted vector from the actual words.
b. Given a word, compare the list of words provided as an output of the above trained model with the actual Russell conjugates of that word.
c. Apply the trained model to the words which are not in the training set. Check if the output are actual Russell conjugates or not. Calculate accuracy.

Phase 3: Analyze Errors:

After implementing training methods and evaluating them, next phase will be analyzing the errors for two primary purposes:

1. To be able to explain the results and the occurrence of the error, which can then lead to addition of some manual filters. For example, if some types of antonyms are considered as errors, we can remove those by adding a filter at the appropriate stage.
2. To look for areas of improvement within word2vec procedure of training words into vectors.

Phase 4: Work on improvement of word2vec procedure:

Post phase 3's error analysis, we can work on modifying the algorithm of word2vec, such that it produces vectors that more actively predict/produce Russell conjugates. Code for word2vec is open source. Illinois Tech's High-Performance Computing team can help in training billions of words in large corpus for achieving better accuracy.

Phase 5: Documentation

References

[1] P. S. Argamon, *Russell Conjugative Debiaser Precis.*

[2] M. J. I. M. N. Dr. S. Vijayarani, "Preprocessing Techniques for Text Mining - An Overview," *International Journal of Computer Science & Communication Networks,Vol 5(1),* pp. 7-16.

[3] KDNuggets, "Text Data Preprocessing," [Online]. Available: https://www.kdnuggets.com/2018/03/text-data-preprocessing-walkthrough-python.html.

[4] V. Y. a. H. E. Prabu Palanisamy, "Serendio: Simple and Practical lexicon based approach to Sentiment," in *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Seventh International Workshop on Semantic*, Georgia, Atlanta, 2013.

[5] T. P. Satanjeev Banerjee, "An adapted Lesk algorithm for word sense disambiguation using WordNet," in *International conference on intelligent text processing and computational linguistics*, 2002.

[6] M. Lesk, "Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone," *SIGDOC '86 Proceedings of the 5th annual international conference on Systems documentation,* pp. 24-26.

[7] The Stanford Natural Language Processing Group, [Online]. Available: https://nlp.stanford.edu/software/nndep.html.

[8] D. C. a. C. Manning, "A Fast and Accurate Dependency Parser Using Neural Networks," 2014.

[9] D. Jurafsky, Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 2008.

[10] K. C. G. C. J. D. Tomas Mikolov, "Efficient estimation of word representations in vector space," *arXiv:1301.3781,* 2013.

[11] I. S. K. C. G. C. J. D. T Mikolov, "Distributed representations of words and phrases and their compositionality," *Neural information processing systems,* 2013.

[12] R. Socher, "Stanford University - CS 224N," 2018. [Online]. Available: http://web.stanford.edu/class/cs224n/index.html#coursework.

[13] L. Tan, "Pywsd: Python Implementations of Word Sense Disambiguation (WSD) Technologies [software]," 2014. [Online].

[14] Analytics Vidhya, "Web Scraping Wikipedia Tables using BeautifulSoup and Python," 1 May 2018. [Online]. Available: https://medium.com/analytics-vidhya/web-scraping-wiki-tables-using-beautifulsoup-and-python-6b9ea26d8722.

[15] S. Banerjee and T. Pedersen, "Computational Linguistics and Intelligent Text Processing - 3rd International Conference, CICLing 2002, Proceedings," pp. 136-145, Jan 1 2002.

[16] A. C. a. C. C. a. D. J. a. O. H. a. L. Brunie, *A Study and Comparison of Sentiment Analysis Methods for Reputation Evaluation.*

Appendix 1

Programming languages:

Python for Natural Language Processing, Javascript and Python for Web App. Python packages:

For NLP tasks:

1. NLTK: The Natural Language Toolkit is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language.

2. Genism: It is a robust open-source vector space modeling and topic modeling toolkit implemented in Python. It uses NumPy, SciPy and optionally Cython for performance.
3. Pywsd: Python Implementations of Word Sense Disambiguation (WSD) technologies [13].

For Web Plugin:

1. BeautifulSoup [14]: Python package for parsing HTML and XML documents. It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping.
2. Flask: It is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself.
3. Requests: Requests is a Python HTTP library, released under the Apache2 License. The goal of the project is to make HTTP requests simpler and more human-friendly.
4. Urllib: It is a package that collects several modules for working with URLs.