**Machine Learning Engineer Nanodegree**

Capstone Project

*“Semantic similarity extraction using word vectors in Mahabharata dataset”*

Tilak D

March 23rd, 2017

**Definition**

***Project Overview***

**Natural language processing (NLP)** is a field of [computer science](https://en.wikipedia.org/wiki/Computer_science), [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence), and [computational linguistics](https://en.wikipedia.org/wiki/Computational_linguistics) concerned with the interactions between [computers](https://en.wikipedia.org/wiki/Computer) and [human (natural) languages](https://en.wikipedia.org/wiki/Natural_language) and, in particular, concerned with programming computers to fruitfully process large [natural language corpora](https://en.wikipedia.org/wiki/Corpus_linguistics).

Challenges in natural language processing frequently involve [natural language understanding](https://en.wikipedia.org/wiki/Natural_language_understanding), [natural language generation](https://en.wikipedia.org/wiki/Natural_language_generation) (frequently from [formal, machine-readable logical forms](https://en.wikipedia.org/wiki/Formal_language)), [connecting language and machine perception](https://en.wikipedia.org/wiki/Symbol_grounding_problem), [managing human-computer dialog systems](https://en.wikipedia.org/wiki/Dialog_system), or some combination thereof.

The [Mahabharata](https://en.wikipedia.org/wiki/Mahabharata) is one of the two major [Sanskrit](https://en.wikipedia.org/wiki/Sanskrit_literature) [epics](https://en.wikipedia.org/wiki/Indian_epic_poetry) of [ancient India](https://en.wikipedia.org/wiki/History_of_India). The Mahabharata is an epic narrative of the [Kurukshetra War](https://en.wikipedia.org/wiki/Kurukshetra_War) and the fates of the [Kaurava](https://en.wikipedia.org/wiki/Kaurava) and the [Pandava](https://en.wikipedia.org/wiki/Pandava) princes. It also contains [philosophical](https://en.wikipedia.org/wiki/Hindu_philosophy) and devotional material, such as a discussion of the four "goals of life" or [purusharthas](https://en.wikipedia.org/wiki/Purusharthas). Among the principal works and stories in the Mahabharata are the [Bhagavad Gita](https://en.wikipedia.org/wiki/Bhagavad_Gita), the story of [Damayanti](https://en.wikipedia.org/wiki/Damayanti), an abbreviated version of the Ramayana, and the [Rishyasringa](https://en.wikipedia.org/wiki/Rishyasringa), often considered as works in their own right.

The Mahabharata is the longest known epic poem and has been described as "the longest poem ever written" Its longest version consists of over 100,000 [shloka](https://en.wikipedia.org/wiki/Shloka) or over 200,000 individual verse lines (each shloka is a couplet), and long prose passages. About 1.8 million words in total, the Mahabharata is roughly ten times the length of the [Iliad](https://en.wikipedia.org/wiki/Iliad) and the [Odyssey](https://en.wikipedia.org/wiki/Odyssey) combined, or about four times the length of the Ramayana, which makes it a huge dataset for using NLP.

By utilizing NLP, we can organize and structure knowledge of the huge Mahabharata to perform tasks such as automatic summarization, translation, named entity recognition, relationship extraction, sentiment analysis and topic segmentation, which will be helpful for extracting quick, short and concise answers.

Word2vec ([Ref[1]](http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf)) is a group of related models that are used to produce [word embeddings](https://en.wikipedia.org/wiki/Word_embedding). These models are shallow, two-layer [neural networks](https://en.wikipedia.org/wiki/Neural_network) that are trained to reconstruct linguistic contexts of words. Word2vec takes a large corpus of text as input and produces a [vector space](https://en.wikipedia.org/wiki/Vector_space), typically of several hundred [dimensions](https://en.wikipedia.org/wiki/Dimensions), with each unique word in the [corpus](https://en.wikipedia.org/wiki/Corpus_linguistics) being assigned a corresponding vector in the space ([Ref[2]](https://arxiv.org/pdf/1301.3781.pdf)). [Word vectors](https://en.wikipedia.org/wiki/Word_vectors) are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space.

***Problem Statement***

In ancient times this knowledge used to pass along generations, but in this fast moving world, everyone needs answers easily and to be in their fingertips. Most of the relationships between characters in lengthy novels are hard to remember for general public, here NLP’s Semantic similarities come into play.

This corpus of data, of about 29100 words will be fed as input to the model to create word vectors and with the help of word2vec, we would analyze semantic similarities between characters, i.e. word vector representation will be created with neural network and by measuring similarity between similar words through cosine similarity to find the similarity between words. This will be used to answer questions related to relationships between characters in Mahabharata. For example, Arjuna was the son of Indra- the king of celestials and Krishna was son of Vasudeva. If an input is given as Arjuna, Indra and Krishan, system should be capable to provide an answer as Vasudeva, based on the knowledge learnt using NLP.

***Metrics***

As explained in the previous section, the result obtained can only be objectively compared with the real facts. An evaluation metrics can only be a percentage of correct semantic obtained, which will be obtained through a sizable number of inputs. The model will be tested with an input containing a list of all the semantics obtained from Mahabharata in the below format.

{A} is related to {B}, as {C} is related to {D}

Here A, B and D would be the inputs and C will the output provided by the model, based on the learning. Below is a subset of the compiled test semantics (Red words are the expected outputs),

Dhritarastra is related to Pandu, as Sahadeva is related to Nakula

Bhima is related to Arjuna, as Ambalika is related to Ambika

Pandu is related to Kunti, as Dhritarashtra is related to Gandhari

Bhima is related to Draupadi, as Arjuna is related to Chitrangada

Karna is related to Kunti, as Duryodhana is related to Gandhari

Bhima is related to Draupadi, as Arjuna is related to Subhadra

Yudhisthira is related to Kunti, as Duryodhana is related to Gandhari

Bhima is related to Kunti, as Nakula is related to Madri

Bhima is related to Draupadi, as Arjuna is related to Ulupi

Vichitravirya is related to Ambalika, as Vichitravirya is related to Ambika

Bhima is related to Ghatotkacha, as Arjuna is related to Abhimanyu

Bhima is related to Draupadi, as Arjuna is related to Draupadi

The output provided by the model is compared to the actual semantics and a percentage accuracy is calculated, i.e. (Total number of right answers given)/(Test size).

**Analysis**

***Data Exploration***

Dataset is a set 18 text file, where in each text file is Parva (Which means book in Sanskrit). Figure 1 shows information of all 18 books.

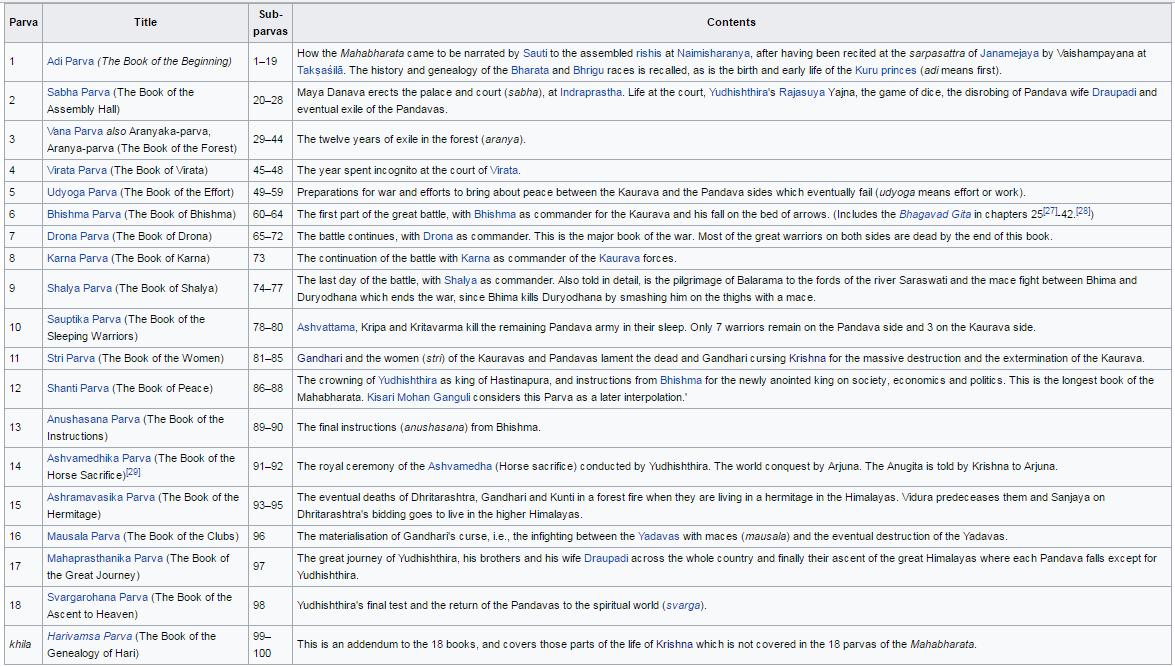


Figure 1: Title and contents of all 18 books of Mahabharata

Dataset was obtained from an online library, [Nitaaiveda](http://nitaaiveda.com/All_Scriptures_By_Acharyas/Historical_Works/Mahabharata.htm). Figure 2 indicates the statistics of all the books combined, having approximately 291,000 words in total.



Figure 2: Statistics of Mahabharata

This corpus of data, of about 29100 English words will be fed as input to the model to create word vectors using word2vec ([Ref[3]](http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf)), and with the help of NLTK, we would analyze semantic similarities between characters. Below is a small snippet from the first book of Mahabharata (One of eighteen).

According to the historical records of this earth, there once lived a King named Maharaja Shantanu, the son of Pratipa, who took his birth in the solar dynasty and was considered naradeva, the manifest representative of the Supreme Lord on earth. His fame and rule extended to all parts of the world. The qualities of self-control, liberality, forgiveness, intelligence, modesty, patience and power always resided this exalted emperor. His neck was marked with three lines like a conchshell, and his shoulders were broad. In prowess He resembled a maddened elephant. Above all these qualities, he was a devoted servant of Lord Vishnu, and therefore he was given the title, "King of kings".

Once when Maharaja Shantanu, that bull among men, was wandering in the forest, he came upon a place frequented by the Siddhas and Charanas (a class of heavenly demigods). There he saw an angelic woman who appeared like the goddess of fortune herself. In truth, she was the personification of the river Ganges. She was glancing at the monarch with her youthful longing eyes, and Maharaja Shantanu became attracted to her. He then approached her inquiring, "O beautiful woman, are you from the race of the Gandharvas, Apsaras, Yakshas, Nagas or the human race? As yet I have no queen, and your birth appears divine. Whatever your origin, O celestial beauty, I request you to become my wife."

***Algorithms and Techniques***

As described above the corpus of words will be used as an input to create word vectors ([Ref[4]](https://arxiv.org/pdf/1301.3781.pdf)) using word2vec, with the help of t-SNE ([Ref[5]](http://jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf)), reduce the dimensions of the word vectors and finally use cosine similarity to analyze semantic similarities, i.e. to answer relationship questions based on the learning. The end solution of this project will be to analyze relationships and logics in the dataset.

Skeleton of the approach will be,

1. Create a dataset by converting corpus into sentences in turn into a bag of words.
2. Improve the dataset by removing the words and symbols that does not have meanings.
3. Build model by training word2vec and build a vocabulary.
4. The trained word vectors will be in a high dimension, example more than 200 dimension. Using t-distributed stochastic neighbor embedding or t-SNE to reduce this higher dimension to a feasible, analyzable dimension size. Train the above dimensionality reduction algorithms to create a lower dimension dataset. Plot and analyze it for semantics.
5. For further analysis and to answer the problem statement, cosine similarity is used to assess similarities between 2 word vectors, to answer similarity questions on the 3rd word vector.
6. Calculate accuracy.

***Benchmark***

The problem which is being solved can only benchmarked based on the real info based on the book. As described in an example in Problem Statement, Arjuna was the son of Indra- the king of celestials and Krishna was son of Vasudeva. If an input is given as Arjuna, Indra and Krishna, system should be capable to provide an answer as Vasudeva, based on the father son relation knowledge learnt using NLP. This result can only be objectively compared with the real facts.

The real facts about the data set already exists, to benchmark the model I have compiled 144 unique relationship facts, such as father-son, mother-son, siblings and spouse. These 144 unique relationships are used in different combinations to generate 1551 relations. For example,

1. Finding Son given Father, based on an example Son-Father relationship
2. Finding Father given Son, based on an example Father-Son relationship
3. Finding Son given Mother, based on an example Son-Mother relationship
4. Finding Mother given Son, based on an example Mother-Son relationship
5. Finding Husband given Wife, based on an example Husband-Wife relationship
6. Finding Wife given Husband, based on an example Wife- Husband relationship
7. Finding Sibling, based on a Sibling example.

Below are a few of the real data used to benchmark the model.

Dhritarastra is related to Pandu, as Sahadeva is related to Nakula

Bhima is related to Arjuna, as Ambalika is related to Ambika

Pandu is related to Kunti, as Dhritarashtra is related to Gandhari

Bhima is related to Draupadi, as Arjuna is related to Chitrangada

Karna is related to Kunti, as Duryodhana is related to Gandhari

.

.

**Methodology**

***Data Preprocessing***

Initial part of the data preprocessing is done by removing all the *Stop words* from the corpus. Stop words are extremely common words, such as the, at, a, an; which would appear to be of little value to the meaning of the sentence, so that we can focus on the important words instead.

For example, consider the first sentence from first book of Mahabharata,

According to the historical records of this earth, there once lived a King named Maharaja Shantanu, the son of Pratipa, who took his birth in the solar dynasty and was considered naradeva, the manifest representative of the Supreme Lord on earth.

This can be reduced to a sentence as shown below and still carry majority of the intended meaning.

According historical records earth lived King named Maharaja Shantanu son Pratipa took birth solar dynasty considered naradeva manifest representative Supreme Lord earth.

Next part of the preprocessing step is done by *tokenizing* all the sentences from 18 books into words for further word analysis. Tokenization is the task of chopping sentences up into pieces, called tokens, also at the same time throwing away certain characters, such as punctuation. Figure 3 shows the input and tokenized output of an example sentence.

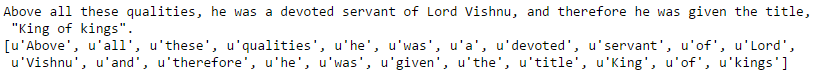


Figure 3: Tokenizing

***Implementation***

The main steps involved in the developed model are listed below along with code snippets,

1. Corpus to bag of words – All the 18 books of Mahabharata is merged to make a huge corpus. Paragraphs are split into sentences and in turn this corpus is chopped into words with the help of split() function.
2. Stop word and punctuation removal - Improve the dataset by removing the stop words and symbols that does not have meanings. This is done using stopwords and punkt packages of NLTK library.
3. Train word2vec - Train word2vec on this processed corpus, thereby converting words into vectors. This is done using Gensim’s word2vec library. Below is a code snippet.

mahabharata2vec = w2v.Word2Vec(

sg = 1,

seed = 1,

workers = multiprocessing.cpu\_count(),

size = 500,

min\_count = 7,

window = 25,

sample = 1e-3

)

mahabharata2vec.train(sentences)

1. The trained word vectors will be in a high dimension, in the above code snippet number of dimentions is equal to 500. With the help of t-distributed stochastic neighbor embedding or t-SNE reduce this higher dimension to a feasible, analyzable dimension size. T-SNE is impemented using sklearn. Below is a code snippet of the implementation.

tsne =sklearn.manifold.TSNE(n\_components=3,perplexity=50.0,n\_iter=9000,random\_state=0)  
all\_word\_vectors\_matrix = mahabharata2vec.wv.syn0

Train the above dimensionality reduction algorithms to create a lower 3 dimension dataset.

all\_word\_vectors\_matrix\_3d = tsne.fit\_transform(all\_word\_vectors\_matrix)

The x, y and z coordinates can be extracted for plotting purpose and semantic analysis.

points = pd.DataFrame(

[

(word, coords[0], coords[1], coords[2])

for word, coords in [

(word, all\_word\_vectors\_matrix\_2d[mahabharata2vec.wv.vocab[word].index])

for word in mahabharata2vec.wv.vocab

]

],

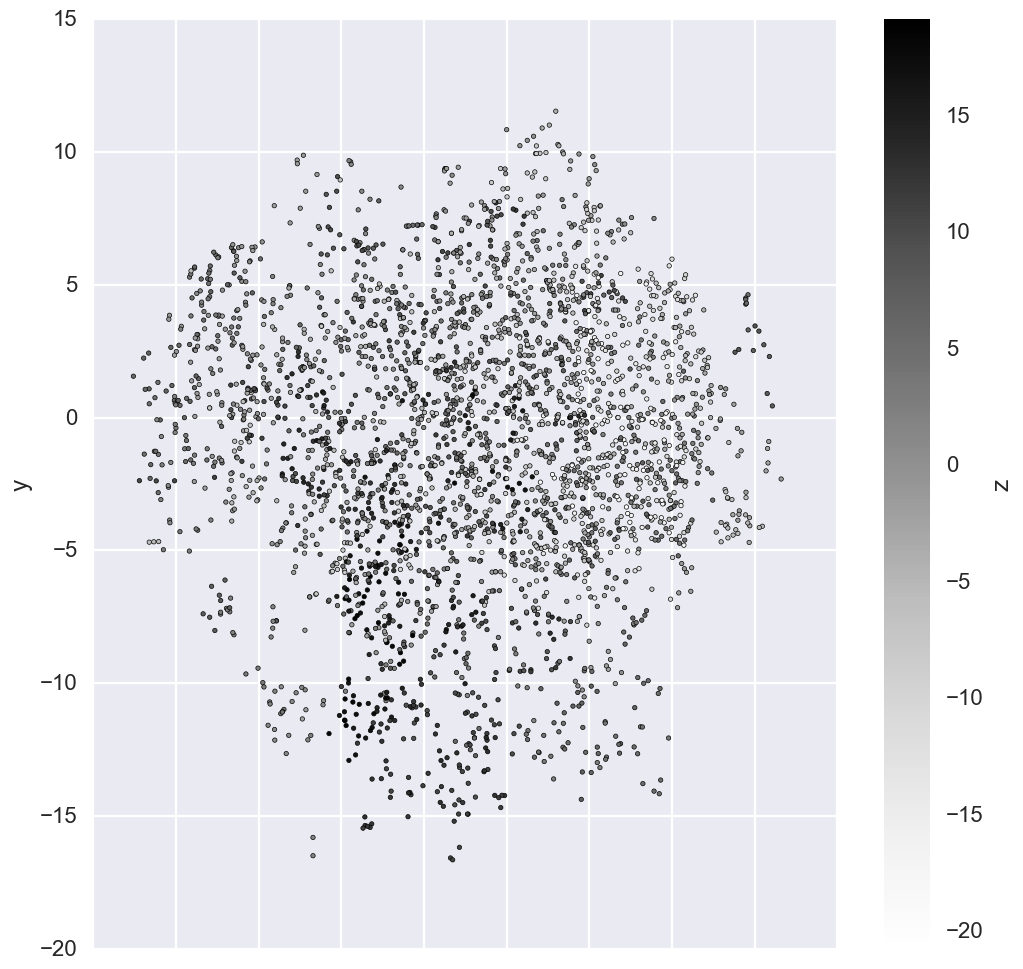
columns=["word", "x", "y", "z"]

)

Coordinates of the first few words are a follows,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | word | x | y | z |
| 0 | raining | 0.781272 | -4.57775 | 6.318752 |
| 1 | yellow | 0.704123 | 0.076027 | 2.908227 |
| 2 | four | -3.05614 | -1.01975 | 2.592682 |
| 3 | woods | 2.339273 | -4.82267 | 4.904764 |
| 4 | hanging | -0.47257 | -3.03956 | 6.286031 |
| 5 | looking | 2.380621 | -1.55693 | 3.994533 |
| 6 | granting | 3.470153 | -5.0568 | 3.453314 |
| 7 | eligible | -1.1368 | -7.32454 | 1.005169 |
| 8 | Kundadahara | 1.235221 | -3.27123 | 7.61231 |
| 9 | lord | 4.241032 | -3.02878 | 0.147951 |
| 10 | sinking | 0.687666 | -2.61176 | 5.224991 |

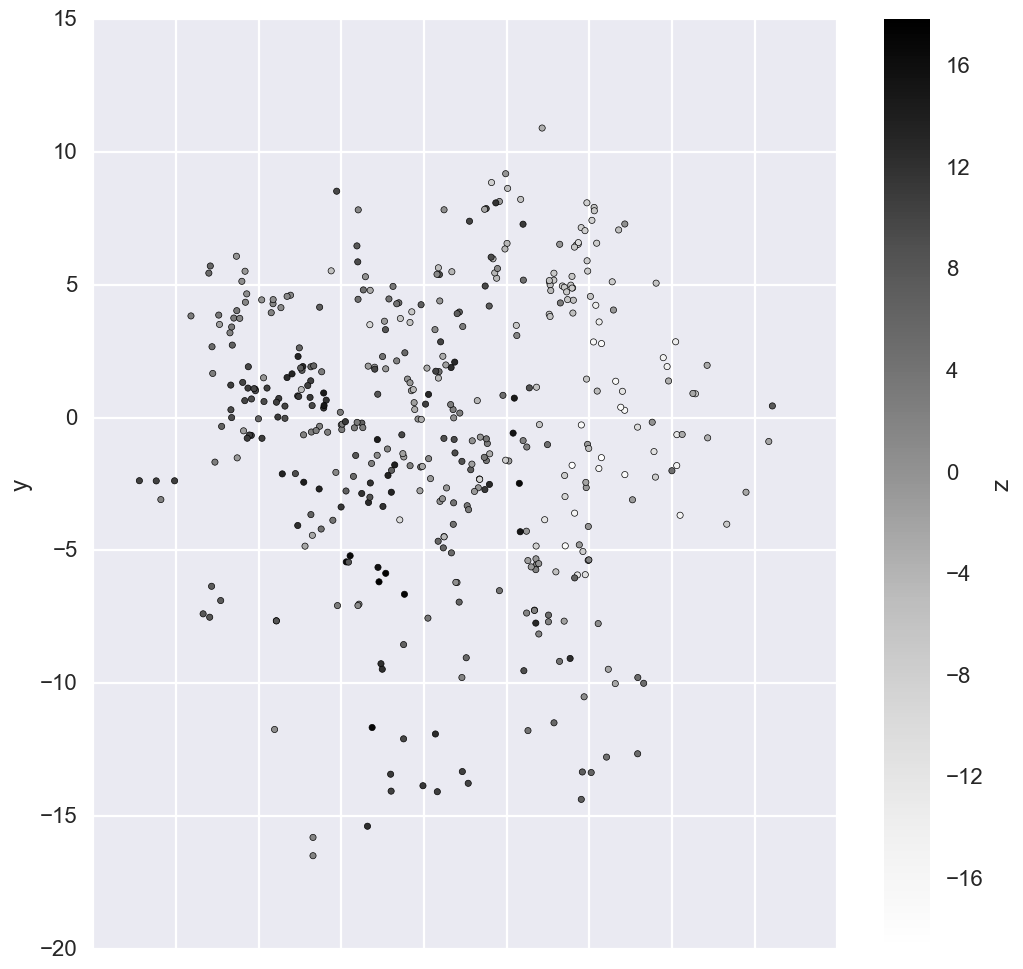
Based on the extracted coordinates, plot scatter plot for visual analysis.



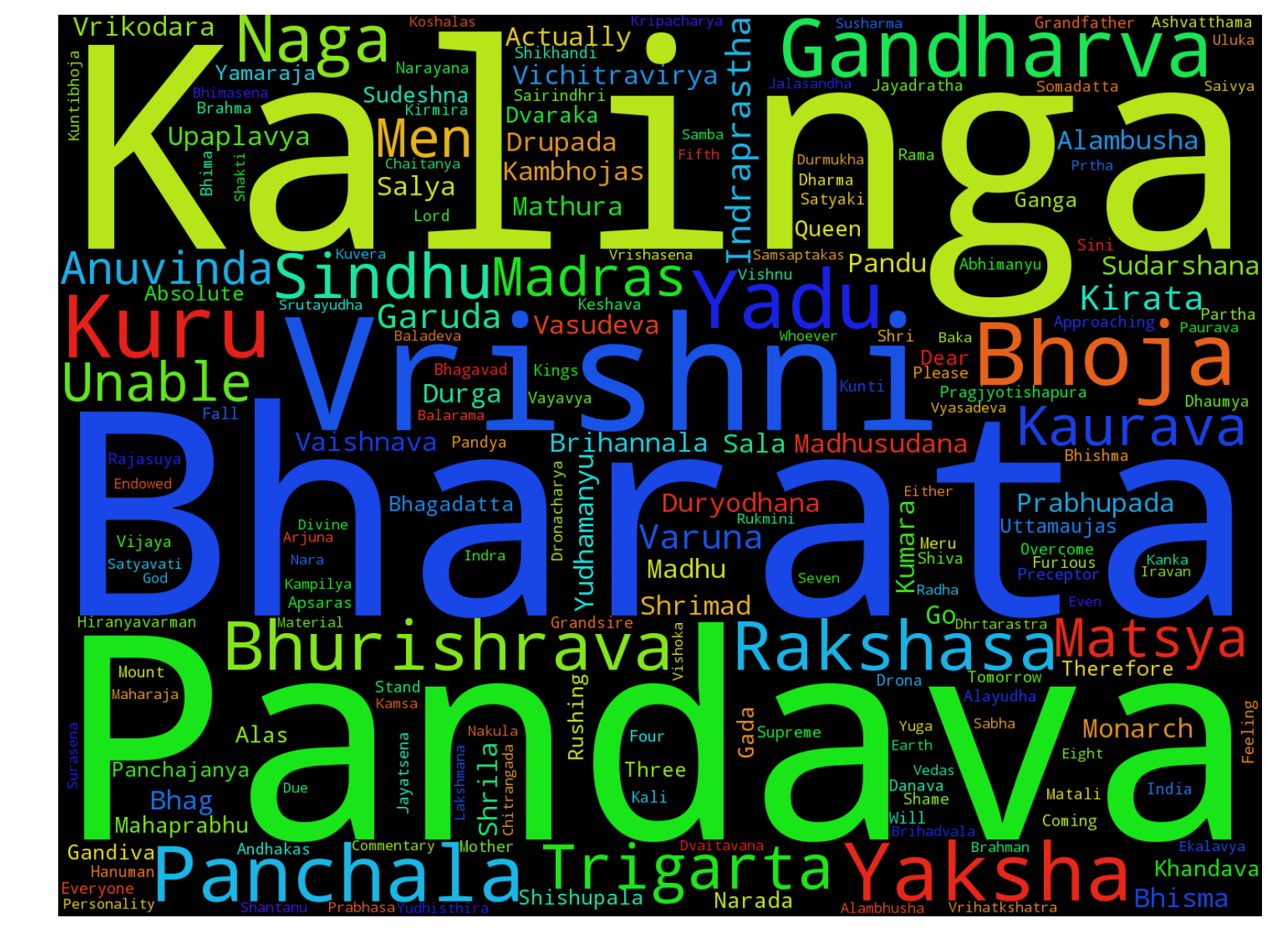
1. As our final goal is to identify relationships between people, extraction of Proper nouns from the above step is a right way to reduce noice. This is done with the help of NLTK’s pos\_tag. Proper nouns from the above scatter plot is shown below.

tagged = nltk.pos\_tag(tokens)

nouns = [word for word,pos in tagged if (pos == 'NNP') or (pos == 'NNPS')]



Output nouns can be shown as word cloud,



1. For further analysis and to answer the problem statement we use cosine similarity, to assess similarities between 2 word vectors, to answer similarity questions on the 3rd word vector.

Firstly, lets see all the similar words, when given a character’s name. Lets see all the similar words related to Arjuna, 83% similar word is Partha, which is another name of Arjuna.

mahabharata2vec.most\_similar("Arjuna")

Output:

[(u'Partha', 0.8320261240005493),

(u'Daruka', 0.8209834694862366),

(u'Satyaki', 0.7644478678703308),

(u'Pradyumna', 0.7604053020477295),

(u'Karna', 0.759215772151947),

(u'Bhima', 0.7575901746749878),

(u'Kuvera', 0.7552096843719482),

(u'Shishupala', 0.7415941953659058),

(u'Subhadra', 0.738818883895874),

(u'wanting', 0.7378615736961365)]

Second part of the problem statement of answering relationship questions is done with the help of Cosine Similarity. Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them.

def nearest\_similarity\_cosmul(start1, end1, end2):

similarities = mahabharata2vec.most\_similar\_cosmul(

positive=[end2, start1],

negative=[end1]

)

start2 = similarities[0][0]

print("{start1} is related to {end1}, as {start2} is related to {end2}".format(\*\*locals()))

return start2

When the system is asked to provide answer for the below question, i.e. Dhritarastra and Pandu are brothers and what is for Nakula, system provides below Sahadeva, who is brother of Sahadeva.

nearest\_similarity\_cosmul("Dhritarastra" ,"Pandu", "Nakula")

Output: Dhritarastra is related to Pandu, as Sahadeva is related to Nakula

***Refinement***

Refinement was done in 3 stages,

1. word2vec model creation
2. t-SNE parameters
3. Proper Noun extraction for final relation analysis.

Lets analyze each of the above.

1. **word2vec model creation**: word2vec has main 7 parameters, viz. *seed, workers, size, min\_count, window and sample*. In any dataset high frequency words often provide little information. Words with frequency above a certain threshold may be subsampled to increase training speed, this can be done with the help of *min\_count*. Quality of word embedding increases with higher dimensionality. But after reaching some point, marginal gain will diminish. Typically, the dimensionality of the vectors is set to be between 100 and 1,000. The size of the context window determines how many words before and after a given word would be included as context words of the given word. According to the authors' note, the recommended value is 5 to 10. As Mahabharata data set has lengthy sentences, I have chosen context window size as 25.

mahabharata2vec = w2v.Word2Vec(

sg=1,

seed=seed,

workers=num\_workers,

size=num\_features,

min\_count=min\_word\_count,

window=context\_size,

sample=downsampling

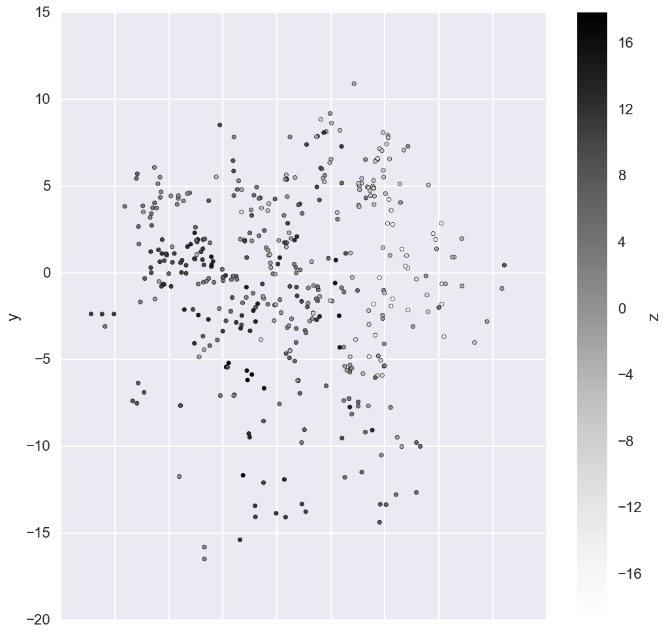
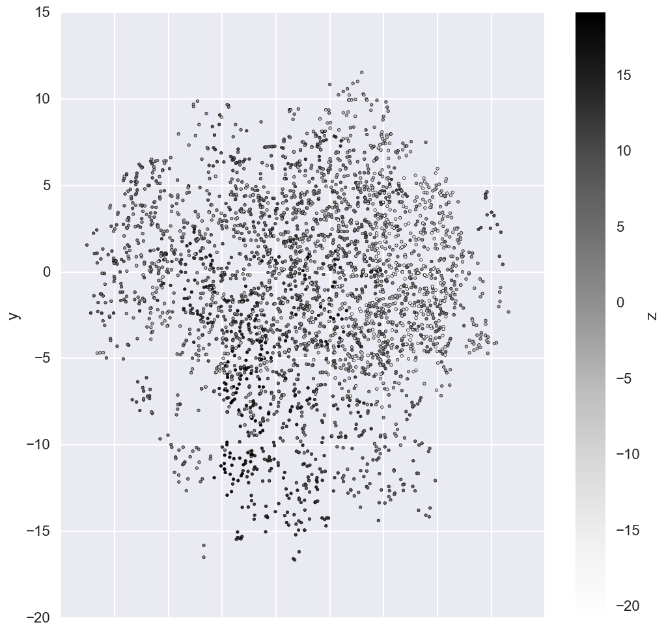
)

1. **t-SNE parameters**: t-SNE has 3 important parameters, viz. *n\_iter, n\_components and perplexity*. Perplexity is a measure for information that is defined as 2 to the power of the Shannon entropy. The perplexity of a fair die with k sides is equal to k. In t-SNE, the perplexity may be viewed as a knob that sets the number of effective nearest neighbors. It is comparable with the number of nearest neighbors k that is employed in many manifold learners. The performance of t-SNE is fairly robust under different settings of the perplexity. The most appropriate value depends on the density of your data. Loosely speaking, one could say that a larger / denser dataset requires a larger perplexity. Typical values for the perplexity range between 5 and 50.

Ref: http://distill.pub/2016/misread-tsne/

tsne =sklearn.manifold.TSNE(n\_components=3,perplexity=50.0,n\_iter=9000,random\_state=0)

1. Next step of refinement is done by extracting only proper nouns from the t-SNE output and using it for further analysis, as our main aim is relationship extraction. This can be seen by the number of points reduction in the below plot (Left is t-SNE for all word and right is t-SNE only for Proper Noun)



**Results**

***Model Evaluation and Validation***

In this section, the final model and any supporting qualities should be evaluated in detail. It should be clear how the final model was derived and why this model was chosen. In addition, some type of analysis should be used to validate the robustness of this model and its solution, such as manipulating the input data or environment to see how the model’s solution is affected (this is called sensitivity analysis). Questions to ask yourself when writing this section:

- \_Is the final model reasonable and aligning with solution expectations? Are the final parameters of the model appropriate?\_

- \_Has the final model been tested with various inputs to evaluate whether the model generalizes well to unseen data?\_

- \_Is the model robust enough for the problem? Do small perturbations (changes) in training data or the input space greatly affect the results?\_

- \_Can results found from the model be trusted?\_

***Justification - YET TO BE DONE***

In this section, your model’s final solution and its results should be compared to the benchmark you established earlier in the project using some type of statistical analysis. You should also justify whether these results and the solution are significant enough to have solved the problem posed in the project. Questions to ask yourself when writing this section:

- \_Are the final results found stronger than the benchmark result reported earlier?\_

- \_Have you thoroughly analyzed and discussed the final solution?\_

- \_Is the final solution significant enough to have solved the problem?\_

**Conclusion**

\_(approx. 1-2 pages)\_

***Free-Form Visualization - YET TO BE DONE***

In this section, you will need to provide some form of visualization that emphasizes an important quality about the project. It is much more free-form, but should reasonably support a significant result or characteristic about the problem that you want to discuss. Questions to ask yourself when writing this section:

- \_Have you visualized a relevant or important quality about the problem, dataset, input data, or results?\_

- \_Is the visualization thoroughly analyzed and discussed?\_

- \_If a plot is provided, are the axes, title, and datum clearly defined?\_

***Reflection - YET TO BE DONE***

In this section, you will summarize the entire end-to-end problem solution and discuss one or two particular aspects of the project you found interesting or difficult. You are expected to reflect on the project as a whole to show that you have a firm understanding of the entire process employed in your work. Questions to ask yourself when writing this section:

- \_Have you thoroughly summarized the entire process you used for this project?\_

- \_Were there any interesting aspects of the project?\_

- \_Were there any difficult aspects of the project?\_

- \_Does the final model and solution fit your expectations for the problem, and should it be used in a general setting to solve these types of problems?\_

***Improvement - YET TO BE DONE***

In this section, you will need to provide discussion as to how one aspect of the implementation you designed could be improved. As an example, consider ways your implementation can be made more general, and what would need to be modified. You do not need to make this improvement, but the potential solutions resulting from these changes are considered and compared/contrasted to your current solution. Questions to ask yourself when writing this section:

- \_Are there further improvements that could be made on the algorithms or techniques you used in this project?\_

- \_Were there algorithms or techniques you researched that you did not know how to implement, but would consider using if you knew how?\_

- \_If you used your final solution as the new benchmark, do you think an even better solution exists?\_

-----------

\*\*Before submitting, ask yourself. . .\*\*

- Does the project report you’ve written follow a well-organized structure similar to that of the project template?

- Is each section (particularly \*\*Analysis\*\* and \*\*Methodology\*\*) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification?

- Would the intended audience of your project be able to understand your analysis, methods, and results?

- Have you properly proof-read your project report to assure there are minimal grammatical and spelling mistakes?

- Are all the resources used for this project correctly cited and referenced?

- Is the code that implements your solution easily readable and properly commented?

- Does the code execute without error and produce results similar to those reported?