Analysis Of People Relationship Using Word2Vec on Wiki Data

ABHISHEK GUPTA[1,*] AND AVADHOOT AGASTI[1,**]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: abhigupt@iu.edu
[**] Corresponding authors: aagasti@iu.edu

project-1: Data mining for a wiki url , July 22, 2017

Wikipedia pages of famous personalities contain details like school, spouse, coaches, languages, alma-meter etc. This information is in free form text. In this project, we extract the people information from the Wikipedia page and to establish relationships between them. Specifically, we use the data of known relationships to derive the newer relationships.

Keywords: Cloud, I524, Chemeleon, Word2Vec, Jetstream, Cloudmesh, RAM

Report: https://github.com/cloudmesh/sp17-i524/blob/master/project/S17-IR-P005/report/report.pdf
Code: https://github.com/cloudmesh/cloudmesh.word2vec

CONTENTS

1. INTRODUCTION

Word2Vec [1] is a group of related models that are used to produce word embedding. Word2Vec is used to analyze the linguistic context of the words. In this project, we created Word2vec model using Wikipedia data and news articles. Our focus is people names occurring in the Wikipedia data and to see if Word2vec can be used to understand relationship between people. Typically Wikipedia page for people and celebrities contain the entire family and friends, colleagues information. Our idea is to use Word2vec to see if using a smaller training set of known relationships whether we can derive similar relationship for anyone who has presence on Wikipedia. This mechanism can be then used to convert the data hidden in textual format to more structured data.

We used spark [2] to load the wiki data and create word vectors. We then used vector manipulations to derive the relationships.

2. DESIGN

Figure 1 shows the overall data pipeline for the project. The data pipeline has three important stages:

- Wiki crawler: Wiki crawler runs in batch mode on a standalone machine. It can download wikipedia data as explained in section 2.1. Crawler creates CrawlDB which is a collection of text files. This crawler can be replaced or

Name	Purpose
spark [2]	data analysis
sparkML [3]	machine learning
python [2]	development
ansible [4]	automated deployment

Table 1. Technology Name and Purpose

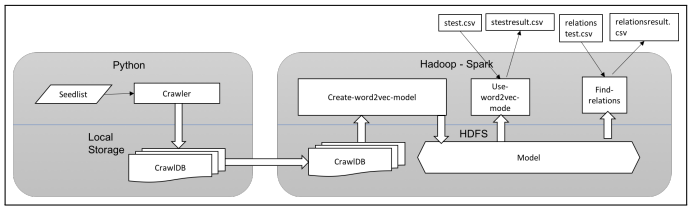

Fig. 1. Data Pipeline.

augmented with any web-crawler which can download or create the text files.

- News crawler: News crawler is responsible for downloading the news articles.
- CreateWord2VecModel: This component is responsible for creating the Word2Vec model for the text files in the CrawlDB. This model runs on spark and stores the model on HDFS. Section 2.3 describes this component in detail.
- UseWord2VecModel and FindRelations: These two components use the pre-created Word2Vec model to find synonym of a word or find the relationships. Section 2.4 describes these components in detail.

2.1. Wiki crawler

The Wiki Crawler component is useful to download the data from web. We implemented a simple crawler using Python which can deep traverse the wikipedia pages and download the text from it. In our crawler implementation, a user can specify the seed pages from wikipedia. User can also specify the maximum number of pages that are required to be downloaded. The crawler first downloads all the pages specified in the seedlist. It then extract the links from each wikipedia page and puts it in a queue which is internally maintained by the crawler. The crawler then downloads the the linked pages. Since this logic is implemented in recursive manner, the crawler can potentially download all the wikipedia pages which can be reached from the pages in the seedlist.

We followed the seedlist based crawler approach so that we can retrieve domain specific web pages. A well chosen seedlist can fetch large number of relevant web pages.

Figure 2 is the flowchart of the crawler implementation.

2.2. News crawler

News crawler is crawler implemented in Python. It executes in batch mode and download latest news article related to the topics configured in its seedlist. The news crawler uses Google APIs [5] to search the topics configured in the seedlist. Then it iterates over the result of each search, and downloads the original HTML page contents. The textual portion of the HTML is extracted using goose python library [6]

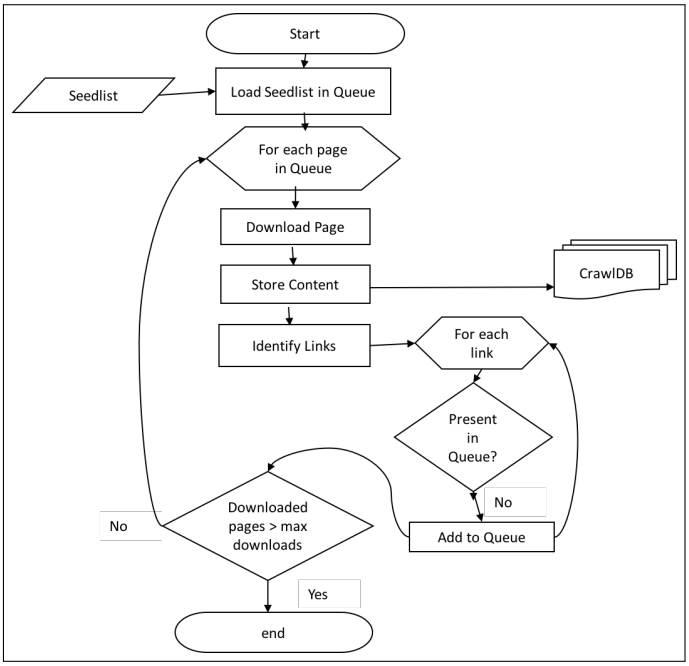

Fig. 2. Flowchart of crawler.

2.3. Word2Vec model creation

CreateWord2VecModel is Spark application implemented in Python. This application is responsible for creating the Word2Vec model and storing it for later use. Figure 3 explains the steps involved in the Word2Vec model creation. We used Spark Feature Extraction [7] for implementing the steps involved in the Word2Vec model creation. These steps are explained below:

- Read crawled documents from HDFS
- For each crawled document, remove special characters from the text
- Tokenize text to create list of words
- Remove the stop words
- Create Word2Vec model
- Store the model on HDFS

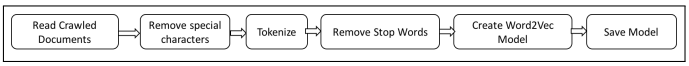

Fig. 3. Steps involved in Word2Vec model creation.

2.4. Using the Word2Vec model to find synonyms and relations

The pre-created Word2Vec model can be queried to find the synonyms or relations between the words. In the context of Word2Vec, synonym of the word is the word that co-occur in similar context [8]. The *UseWord2VecModel* finds the synonyms

for the words provided in the *stest.csv* file. The results are stored in *sresults.csv* file.

The Word2Vec model is also used to find the relationships. [9] explains how the vector operations can be performed on word vectors to derive relationships. The *FindRelations* spark application performs the vector operations to find relationships. The relationtest.csv is input to the *FindRelations* application. The relationtest.csv has 3 words in each row. The application predicts the fourth word which has same relation to the thrid word as the second word related to first word. In the example below

Sachin,Anjali,Sourav

If *Anjali* is spouse of *Sachin*, then *FindRelations* application is expected to predict the first name of the person who is spouse of Sourav. The result of *FindRelations* is saved in relationsresult.csv.

3. DEPLOYMENT

The deployment on the cluster can be accomplished using 2 steps, assuming the cluster is up and running

- Step1: update hosts file *ansible-word2vec/hosts* with the IP of the master. The first node on the cluster becomes the master node.
- Step2: run the script *ansible-word2vec/run.sh*. This script will run the ansible playbooks to accomplish stage1 through stage4 of the deployment process.

Figure 4 shows the deployment stages. The two steps accomplish deployment in multiple stages as discussed in the sections below.

3.1. Stage1

As pre-requisite, we need to create a cluster with 1 or more nodes. We created a 3 node cluster using Cloudmesh [10] command line interface(CLI). Cloudmesh[10] CLI allows you to orchestrate virtual machines(VM) in a cloud environment. For this project, we have used Chameleon and Jetstream cloud providers to orchestrate the VMs using cloudmesh[10] CLI. We can orchestrate a 3 node cluster using following CLI:

```
cm reset
pip uninstall cloudmesh_client
pip install -U cloudmesh_client
cm key add --ssh
cm refresh on
cm cluster define --count 3 \
--image CC-Ubuntu14.04 --flavor m1.medium
cm hadoop define spark pig
cm hadoop sync
cm hadoop deploy
cm cluster cross_ssh
```

We are using Ubuntu14.04 image with m1.medium which comes with 2 CPU, 4GB memory. Also, the nodes created are having hadoop and spark add-ons. We can test the deployment by checking hdfs and spark-submit CLI work fine.

```
ssh cc@<cluster-ip>
sudo su - hadoop
hdfs
spark-submit
```

At this stage our cluster is ready for further deployments.

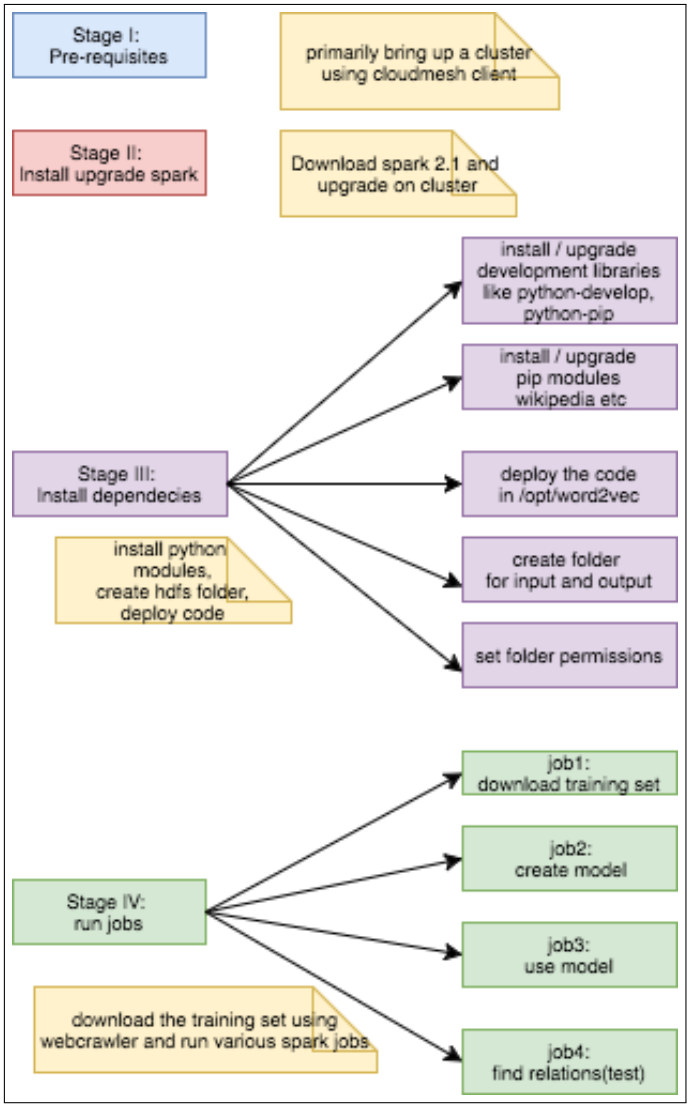

Fig. 4. Deployment stages.

3.2. Stage2

By default, cloudmesh installs spark 1.6 but our word2vec solution requires spark 2.1. We need to upgrade spark on the cluster. In order to do so we can run *install_upgrade_spark.yaml* ansible[4] playbook. This will download and unpack spark2.1 tar ball and further update the softlink to point to spark 2.1 folder.

3.3. Stage3

In this step, we upgrade the development libraries for python, and pip, install python modules like wkipedia, request etc, download the code from git repo and install it in */opt/word2vec* folder, set the folder permissions for the */opt/word2vec* folder so that it can be executed by hadoop user. These steps can be achieved using *word2vec_setup.yaml* playbook. After completing this stage, we are ready for running our word2vec solution on the cluster.

3.4. Stage4

This stage primarily deals with submitting the jobs for various purpose. Before we submit the jobs, we need to make sure input folder are created on hadfs. First, we run the crawler to download the training set and upload the data on hdfs. Further we run various jobs to created model and find relations. Along with these jobs we also run some monitoring jobs. The monitoring job queries spark metrics using

```
http://${spark_master}:4040
```

Stage4 steps can be accomplished using *word2vec_execute.yaml* playbook.

At the end of stage4 we also fetch the execution results from the cluster along with the metrics of execution times at various stages. The output files are fetched into */tmp/word2vec_results*

```
ls -1t /tmp/word2vec_results
jobs.csv
executors.csv
app.csv
stest.csv
relationstest.csv
stestresult.csv
relationsresult.csv
```

Files *jobs.csv, executors.csv, and app.csv* collect the execution time for various jobs. File *relationsresult.csv* file collects the results for sample relations corresponding to *relationstest.csv*. Similarly *stestresult.csv* collects the results corresponding to *stest.csv*.

3.5. Execution

3.5.1. Cleanup

We can execute the run from local system using *run.sh* located inside *ansible-word2vec*. Run script executes all stages sequentially on the remote system. To rerun word2vec, run the cleanup playbook *word2vec_cleanup.yaml* located inside *ansible-word2vec*. Cleanup remove the spark 2.1 binary and the soft link, remove */opt/word2vec* folder as well as any temporary files created during setup step.

3.5.2. Page size

The crawler downloads pages from wikipedia based on the config page count. To modify the page count, we can edit *ansible-word2vec/setupvariables.yml* and set the *max_pages* to desired page size. The crawler downloads individual pages and then combines the pages into a single file before submitting the spark jobs.

3.5.3. Test Results

Test results are downloaded to local machine in */tmp/word2vec_results* folder. Ansible execution log is saved in */tmp/word2vec-logfile.txt*. The log gets appended each time you execute the run script.

3.5.4. Troubleshooting

1. If the installation *run.sh* script fails in middle due to some reason, execute the cleanup script before re-triggering run script again. The run script may fail due to variety of reasons like failed to shh, hadoop not available etc
2. If the run script fails due to spark memory errors, you can modify the spark memory setting in *code/config.properties* push the code to a git feature branch for example spark_test. Modify *word2vec_setup.yaml* git section *version=master* to point to spark_test branch and execute the run script.
3. If hadoop goes into safe mode, goto the cluster namenode and execute the following

   ```
   /opt/hadoop$ bin/hadoop dfsadmin -safemode leave
   ```

   This will remove the cluster from safe mode.

4. BENCHMARKING

We used datasets of 2 different sizes to perform the benchmarking of the application. Table 2 shows the details of the two datasets. The CreateWord2VecModel spark application is most complex and time consuming application. We used this application for the benchmarking. We deployed the application on Chameleon cloud and Jetstream cloud. Table 3 shows the details for the cluster configurations on Chameleon and Jetstream clouds.

Table 2. Dataset Used for Performance Measurement

Parameter	Dataset1	Dataset2
Size of crawldb	1.4MB	7.4MB
Count of files in crawldb	100	500
Source	Wikipedia	Wikipedia

Table 3. Dataset Used for Performance Measurement

Parameter	Chameleon Cluster	Jetstream Cluster
Cluster name	cluster-005	cluster-010
Nodes	2	2
OS	Ubuntu 14.04	Ubuntu 14.04
Flavor	m1.medium	m1.medium
Secgroup	default	default
Assign floating IP	True	True
Cloud	chameleon	iujetstream

Figure 5 shows the total time taken by CreateWord2Vec application for Dataset1 and Dataset2 on the Chameleon and Jetstream cloud environments.

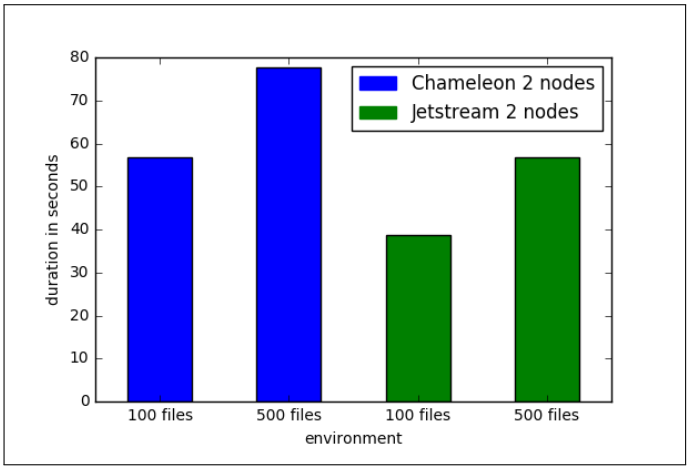

Fig. 5. Time taken by CreateWord2Vec

4.1. Working with large dataset

There are several configuration parameters added in the application to fine tune the behavior of the spark applications. When working with larger datasets, the spark applications can go out of memory. Following parameters can be configured in the *ansible-word2vec/setupvariables.yml* to handle such situation.

```
spark_executor_memory = <memory given to executor>
spark_driver_memory = <memory given to driver>
max_result_size = <maximum result size>
```

Table 4 show cluster configuration of Chameleon cloud used for 1k files dataset

Table 4. Cluster configuration used for performance measurement

Parameter	Chameleon Cluster
Cluster name	cluster-006
Nodes	4
OS	Ubuntu 14.04
Flavor	m1.medium
Secgroup	default
Assign floating IP	True
Cloud	chameleon

We tried running out solution on a 4 node cluster as described in **Table 4** with 1000 files on the crawlDB and the create model script failed with out of memory exceptions.

```
["OpenJDK 64-Bit Server VM warning:
INFO: os::commit_memory(0x00007ff67449f000, 12288, 0)
failed; error='Cannot allocate memory' (errno=12)",
"OpenJDK 64-Bit Server
VM warning: INFO:
os::commit_memory(0x00007ff6746a1000, 12288, 0)
failed; error='Cannot allocate memory' (errno=12)",
"#",
"# There is insufficient memory for the Java
Runtime Environment to continue.",
"# Native memory allocation (malloc) failed to allocate
12288 bytes for committing reserved memory.",
"# An error report file with more information is saved as:",
"# /home/hadoop/hs_err_pid25854.log"], "warnings": []}
```

We need to tune spark memory parameters here, since the words in model become too high and we need as much memory(RAM) to execute the solution or else it goes out of memory(OOM). For example for min_word_count = 5, the number of words in the model were 20839 and executor_memory of 4GB was enough. But for min_word_count of 3 which resulted in 30194 words in model, we have to give 6GB RAM. After making these changes to spark executor and driver memory we were able to run the solution successfully with 1000 files.

Figure 6 shows the total time taken by CreateWord2Vec application varying min word count on the Chameleon cloud with 4 nodes

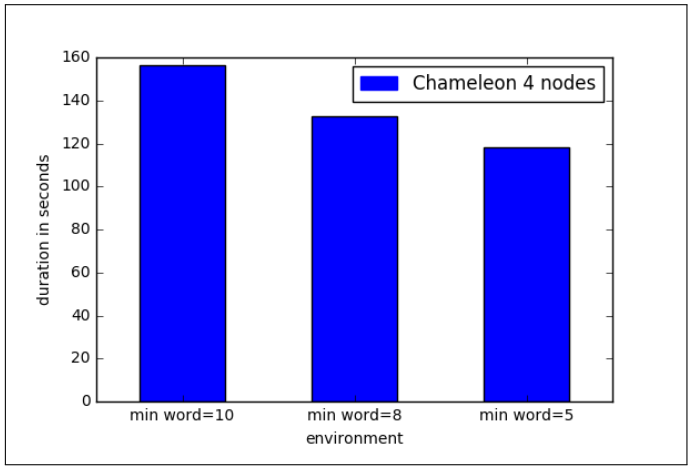

Fig. 6. Time taken on 4 node Chameleon cloud

5. DISCUSSION

The analysis of the results provided interesting insights. Section 5.1 provides key insights on our experiments with Word2Vec on Wikipedia data. Section 5.2 provide key insights on our experience of deployment and execution of Word2Vec application on Chameleon and Jetstream cloud.

5.1. Word2Vec app - key insights

We trained the Word2Vec model with the wikipedia data of Indian Cricket team members. We used *List of India ODI cricketers* as seedlist and downloaded 500 pages from Wikipedia. With this dataset, we observed that the *synonym* or correlated words were pretty accurate. Some interesting examples and its explanation:

sachin -> tendulkar

world -> cup

Sachin Tendulkar [11] is famous Indian cricket player. Naturally, the words *sachin* and *tendulkar* are highly correlated. The Word2Vec model was able to find this correlation.

Cricket World Cup [12] is one of the most viewed sporting event and considered as the flagship event of the international cricket. Hence the words *world* and *cup* are highly correlated in the context of cricket. The Word2Vec model was able to find this correlation.

However, when we tried to find the people relationships using the wikipedia dataset, we could not get good accuracy.

For example, *Anjali Tendulkar* is wife of *Sachin Tendulkar* while *Dona Ganguly* is wife of *Sourav Ganguly* who is another famous Indian Cricket player [13]. When we provided the test record of

sachin,anjali,sourav

we did not get good results.

After observing the wikipedia data, we concluded that there is lot of literature in wikipedia about the cricketers. However, there were very few mentions of their family members, coaches, schools etc. Due to this, we were not getting good results on the people relationships.

To augment the Wikipedia data, we decided to crawl news data which provide large amount of articles containing people and their relationships. We implemented the news crawler as explained in the section 2.2. After downloading about 200 news articles of 20 cricketers, our relationships discovery improved a lot. Some interesting examples are given below: *sachin,anjali,sourav,dona*

sachin,anjali,dhoni,sakshi

sachin,cricket,amitabh,hero

In first two examples, the Word2Vec model was able to identify the people-people relationships, while in third example, the Word2Vec model was able to identify the people-profession relationships.

5.2. Deployment of Word2Vec app on Chameleon and JetStream cloud - key insights

Both Chameleon and Jetstream run on openstack and work seamlessly using cloudmesh and ansible. There is a difference in the flavor for Chameleon and Jetstream, where m1.medium on Chameleon is different than m1.medium on Jetstream.

6. CONCLUSION

Using this project we conclude that we can use Word2Vev model on Wikipedia and news data to find the relationships between the people.

We further conclude that Word2Vec based analytics can be performed on public cloud systems like Chameleon cloud and Jetstream cloud. Our deployment automation, which is implemented using Cloudmesh and Ansible technology demonstrates the power of these technologies to achieve one touch deployment and execution of applications across multiple clouds.

7. ACKNOWLEDGEMENT

We acknowledge our professor Gregor von Laszewski and all associate instructors for helping us and guiding us throughout this project.

8. APPENDICES

Appendix A: Work Distribution The co-authors of this report worked together on the design of the technical solutions, implementation, testing and documentation. Below given is the work distribution

- Avadhoot Agasti
  - Implementation of wiki crawler and news cralwer in Python.
  - Implementation of CreateWord2VecModel in Spark.
  - Implementation of UseWord2VecModel and FindRelations in Spark.
  - Implementation of Python script MonitorSparkApp.
  - Analyzing the Word2Vec model results.
  - Testing of end to end flow on Chameleon cloud.
  - Performance testing and bug fixing in the spark application.
  - Writing related sections in this report.
- Abhishek Gupta
  - Implementation of Ansible scripts for deployment of Spark 2.1 which is required for the spark application.
  - Implementation of Ansible scripts for deployment.
  - Implementing the changes in the spark applications to get it working on HDFS.
  - Setting up and testing the end to end flow on Chameleon cloud.
  - Setting up and testing the end to end flow on Jetstream cloud.
  - Testing the crawler and Word2Vec applications for semantic correctness.
  - Gathering the perfomance statistics for comparison.
  - Writing related sections in this report.

REFERENCES

[1] "Word2Vec, learning vector representation of words," Web Page, accessed: 2017-02-26. [Online]. Available: https://en.wikipedia.org/wiki/Word2vec

[2] "Spark Python API (PySpark)," Web Page, accessed: 2017-02-26. [Online]. Available: https://spark.apache.org/docs/0.9.1/python-programming-guide.html

[3] "Spark ml programming guide," Web Page, accessed: 2017-02-26. [Online]. Available: https://spark.apache.org/docs/1.2.2/ml-guide.html

[4] Red Hat, Inc., "Ansible documentation," Web Page, Jan. 2015, accessed 2017-01-13. [Online]. Available: https://docs.ansible.com/ansible/index.html

[5] "Google custom search," Web Page. [Online]. Available: https://developers.google.com/custom-search/

[6] "Python-goose - article extractor," Code Repo. [Online]. Available: https://github.com/grangier/python-goose

[7] "Extracting, transforming and selecting features," Web Page, accessed: 2017-04-26. [Online]. Available: https://spark.apache.org/docs/latest/ml-features.html

[8] Y. Goldberg and O. Levy, "word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method," *CoRR*, vol. abs/1402.3722, 2014.

[9] "Vector representations of words," Web Page, accessed: 2017-04-26. [Online]. Available: https://www.tensorflow.org/tutorials/word2vec

[10] "Cloudmesh client," Code Repo. [Online]. Available: https://github.com/cloudmesh/client

[11] "Sachin tendulkar," Web Page, accessed: 2017-04-15. [Online]. Available: https://en.wikipedia.org/wiki/Sachin_Tendulkar

[12] "Cricket world cup," Web Page, accessed: 2017-04-15. [Online]. Available: https://en.wikipedia.org/wiki/Cricket_World_Cup

[13] "Sourav ganguly," Web Page, accessed: 2017-04-15. [Online]. Available: https://en.wikipedia.org/wiki/Sourav_Ganguly