# 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 , April 18, 2017

**Given a wiki URL of a person, find out his details like School, Spouse, Coaches, language, alma-meter etc Typically, the wiki page has all this information available but in the free form text. We need to converting it into structured data format so that it can help us analyze the people, from the networks etc We can create a network by navigating the people mentioned in the wiki page.** 

**Keywords:** Cloud, I524

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

## CONTENTS



## 1. INTRODUCTION

Use spark [1] to load the wiki data and create word vectors. Train it using spark ML [2] and then use the model for analytics and prediction. The training set will use Word2Vec. Word2vec [3] is a group of related models that are used to produce word embeddings. Word2Vec is used to analyze the linguistic context of the words. In this project, we created Word2vec model using Wikipedia data. Our focus is people and organization 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 celebraties 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.

| Technology Name | Purpose |
|---|---|
| spark [1] | data analysis |
| sparkML [2] | machine learning |
| python [1] | development |
| ansible [4] | automated deployment |
| collectd [5] | statistics collection for benchmarking |

## 2. PLAN

Following table gives a breakdown of tasks in order to complete the project. Assuming week1 starts after submission of the proposal. These work items are high level breakdown on the tasks and may changes if needed.

| Week | Work Item | Status |
|---|---|---|
| week1 | Basic POC of Word2Vec using Python | planned |
| week2 | Scripts to download Wiki data | planned |
| week3 | Word2Vec Spark program | planned |
| week4 | Training and measuring accuracy | planned |
| week5 | Ansible Deployment script for Spark | planned |
| week6 | Deployment and test on 2 clouds | planned |
| week7 | Performance measurement | planned |
| week8 | Report Creation(parallel) | planned |

## 3. DESIGN

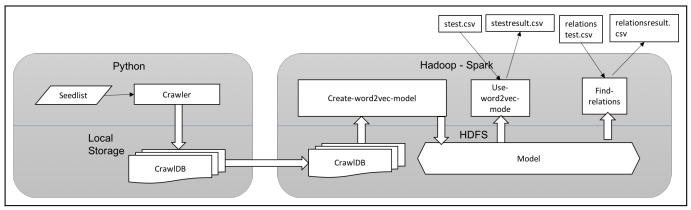


**Fig. 1.** Data Pipeline.

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

- Crawler: Crawler runs in batch mode on a standalone machine. It can download wikipedia data as explained in section 3.1. Crawler creates CrawlDB which is a collection of text files. This crawler can be replaced or augmented with any web-crawler which can download or create the text files.
- create-word2vec-model: 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 3.2 describes this component in detail.
- use-word2vec-model and find-relations: These two components use the precreated word2vec model to find synonym of a word or find the relationships. Section 3.3 describes these components in detail.

### 3.1. Crawler

The 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.

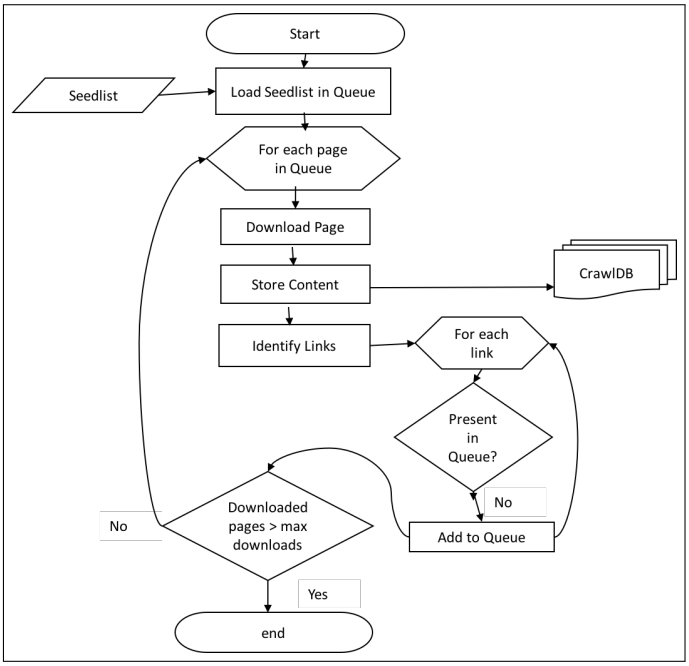


**Fig. 2.** Flowchart of crawler.

### 3.2. Word2Vec Model Creation

### 3.3. Using the Word2Vec Model

## 4. DEPLOYMENT

Solution will be deployed using Ansible [4] playbook. Automated deployment should happen on two or more nodes cluster. Deployment script should install all necessary software along with the project code to the cluster nodes.

## 5. BENCHMARKING

Solution will use collectd [5] to collect statistics. Once the solution is deployed to the cluster. We should benchmark parameters like

- cpu
- memory
- throughput reads/writes

Benchmarking will be done for one or more cloud providers. The deployment scripts should be agnostic to the cloud provider.

## 6. DISCUSSION

TBD

## 7. CONCLUSION

Using this wiki analysis we should be able to build a network based on wiki data.

## 8. ACKNOWLEDGEMENT

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

## 9. APPENDICES

TBD

## REFERENCES

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

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

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

[4] “Ansible, deploy apps. manage systems. crush complexity,” Web Page, accessed: 2017-02-26. [Online]. Available: https://www.ansible.com/

[5] “collectd - the system statistics collection daemon,” Web Page, accessed: 2017-02-26. [Online]. Available: https://collectd.org/