International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 09, September - 2017

# Comparison of Semantic Similarity Determination using Machine Learning

Leena Giri G., Manjula S. H., Venugopal K R
Department of Computer Science and Engineering
University Visvesvaraya College of Engineering
Bangalore University, Bengaluru – 560 001

Karthik Karanth
Department of Computer Science and Engineering,
Dr. Ambedkar Institute of Technology,
Bengaluru-560056

***Abstract*— Evaluating the semantic similarity of two terms is a task central to automated understanding of natural languages. The challenge of "semantic similarity" lies in determining if two chunks of text have very similar meanings or totally different meanings. The amount of research on semantic similarity has increased greatly in the past 5 years, partially driven by the annual SemEval competitions. In this work, to compute the similarity between terms we consider the WordSim and SimLex data set , compare the results obtained between Neural network, Support Vector Machine and Linear Regression machine learning techniques and evaluate the results obtained against the M&C data set. For the data set considered, the Neural Network Model gave the best results, the Linear Regression method fared better than the Support Vector Machine with Regression.**

***Keywords- Component; Semantic Similarity; Neural Networks, Linear Regression; Support Vector Machines.***

## I. INTRODUCTION

The main challenge of natural language processing is to understand the meaning of a piece of text. Judging whether a computer program "understands" a piece of text is an ambiguous task, as a piece of text could be interpreted in different ways by humans too. The challenge of "semantic similarity" lies in determining if two chunks of text have very similar meanings or totally different meanings. Although the problem of semantic similarity has a very simple statement, it has broad applications.

Evaluating the semantic similarity of two terms is a task central to automated understanding of natural languages. Researchers consider neural networks as an effective method for advancing computational understanding of semantic similarity. Bollegala et. al., [1], describe a two-class SVM they trained using those features extracted for synonymous and non-synonymous word pairs selected from WordNet synsets. Experimental results on three benchmark data sets showed that their method outperforms various baselines as well as previously proposed web-based semantic similarity measures.

The amount of research on semantic similarity has increased greatly in the past 5 years, partially driven by the annual SemEval [2] competitions (Jurgens 2014). Results for the 2015 SemEval tasks were published on June 5, 2015. Results and models of SemEval 2015 Task 1 for semantic similarity of twitter messages are described in (Xu et al. 2015). The highly successful ASOBEK [3] system for semantic similarity (Eyecioglu and Keller, 2015) uses a SVM classifier with simple lexical word overlap and character n-grams features. The MITRE system (Zarrella et al., 2015) uses a recurrent neural network augmented with string matching features. Many other systems use a variety of supervised models using features such as n-gram overlap, word alignment, edit distance, cosine similarity of sentence embeddings.

### *A. Motivation*

Measuring the semantic similarity between named entities is vital in many applications such as query expansion, entity disambiguation (e.g., namesake disambiguation), and community mining. Since most named entities are not covered by WordNet, similarity measures that are based on WordNet cannot be used directly in these tasks..

### *B. Contribution*

In this work, to compute the similarity between terms we have used the WordSim dataset and WordNet. WordSim contains **353** pairs of words and their expected term similarity. For the training set, we use **200** of the words in WordSim. M&C (Miller-Charles), a benchmark data set and a subset of WordSim, is used to evaluate the results. Manually maintaining an up-to-date taxonomy of named entities is costly, if not impossible. The proposed semantic similarity measure is appealing for these applications because it does not require precompiled taxonomies.

### *C. Paper Organization*

Section 2 presents a discussion on research related work, section 3 defines the problem and the evaluation of semantic similarity between a pair of words and section 4 briefly concludes the work.

## II. RELATED WORK

Bollegala et. al.,[1] propose a novel pattern extraction algorithm and a pattern clustering algorithm to identify the numerous semantic relations that exist between two given words.. The optimal combination of page counts-based co-occurrence measures and lexical pattern clusters is learned using support vector machines. The proposed method outperforms various baselines and previously proposed web-based semantic similarity measures on three benchmark data sets showing a high correlation with human ratings. Moreover, the proposed method significantly improves the accuracy in a community mining task. Sahami and Heilman [4] measured semantic similarity between two queries using snippets returned for those queries by a search engine. For each query, they collect snippets from a search engine and represent each snippet as a TF-IDF-weighted term vector. Each vector is L2 normalized and the centroid of the set of vectors is computed. Semantic similarity between two queries is then defined as the inner product between the

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 09, September - 2017

corresponding centroid vectors. They did not compare their similarity measure with taxonomy-based similarity measures.

In query expansion [5], a user query is modified using synonymous words to improve the relevancy of the search. One method to find appropriate words to include in a query is to compare the previous user queries using semantic similarity measures. If there exists a previous query that is semantically related to the current query, then it can be either suggested to the user, or internally used by the search engine to modify the original query.

## III. PROBLEM DEFINITION

Semantic similarity is computed between terms in the M&C data set. This data set was used to train the Neural Network model and compare it with supervised and unsupervised machine learning techniques. For the data set considered, the Neural Network Model gave the best results, the Linear Regression method fared better than the Support Vector Machine with Regression.

### *A. Implementation Details*

In order to train the neural network, we used a combination of two data sets, the WordSim data set with 353 word pairs and the SimLex data set with 999 word pairs. These data sets include word pairs and their similarity as judged by a human panel. The M&C data set is used to validate the results.

Obtaining a vector between the words

The following procedure is employed to obtain a vector to represent the word pair:

1) Let the words be word1 and word2
2) Find the list of synsets for both the words
3) synsets1 = synsets(word1)
4) synsets2 = synsets(word2)
5) Find the pair (syn1, syn2) where syn1∈synsets1 and syn2∈synsets2 such that path_similarity(syn1, syn2) is maximised.
6) Find the lch, the lower common hypernym between the words.
7) Set lch1=path_similarity(syn1, lch) and lch2=path_similarity(syn2, lch) if lch exists. Else, lch1=lch2=0
8) Obtain the minimum and maximum depths for both the synsets.
9) Find the shortest path distance between the synsets.
10) The final tuple is of the form (path_sim, lch1, lch2, min_depth1, min_depth2, max_depth1, max_depth2, shortest)

If no synset is found for the word, a similarity score of 0.0 is automatically assigned. For the training step, such word pairs are discarded.

Table 1: Examples of vectors generated

| Vectors Generated | |
|---|---|
| *Word Pairs* | *Vectors Generated* |
| (car, automobile) | (1.0, 1.0, 1.0, 10, 10, 11, 11, 0.0) |
| (food, rooster) | (0.0625, 0.25, 0.07692, 4, 13, 4, 13, 15) |
| (glass, magician) | (0.125, 0.25, 0.2, 4, 5, 4, 8, 7) |
| (car, automobile) | (1.0, 1.0, 1.0, 10, 10, 11, 11, 0.0) |

### *B. Neural Network Approach*

In the first approach, we used a Neural Network(Multilayer Perceptron Regressor) to train the model to the data. The three parameters used are:

- Activation Function: Regularized Linear Unit a = max(0, z)
- Middle Layer 1: 50 units
- I. Middle Layer 2: 20 units
- Learning Rate: 0.008

The input to the first layer is set using the generated tuple. The middle layer is obtained by multiplying the first layer with a weight matrix.

Let

X = First layer, a Nx8 matrix, where N is the number of samples in the current training batch

Y = Output from dataset, a vector of order N, where each element is the similarity between the two words
$M_1$ = First Middle Layer, a Nx50 matrix
$M_2$ = Second Middle Layer, a Nx20 matrix
O = Output Layer, a vector of order N
$W_1$ = Weight matrix between $M_1$ and X, a 50x8 matrix
$W_2$ = Weight matrix between $M_1$ and $M_2$, a 20x50 matrix
$W_3$ = Weight matrix between $M_2$ and O, a 20x1 matrix
$B_1$ = Bias between X and $M_1$, a vector of order 50
$B_2$ = Bias between $M_1$ and $M_2$, a vector of order 20
$B_3$ = Bias between $M_2$ and O, a vector of order 1

The feed-forward phase is as follows:

$$M_1 = X \times W_1^T + B_1 \quad \text{----------} \quad (1)$$
$$M_2 = M_1 \times W_2^T + B_2 \quad \text{----------} (2)$$
$$O = M_2 \times W_3^T + B_3 \quad \text{----------} (3)$$

The loss is determined using the squared loss error function:
$L = (1/N) \times \Sigma[(Y-O)^2]$

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 09, September - 2017

The derivative of the loss with respect to the weights and biases is calculated and the weights and biases are appropriately changed before the next iteration. The above equations are standard neural networks equations. In the implementation, we have used a Python library 'sklearn' to run the neural network.

The neural network is trained using the above hyper-parameters. The training phase is run for 200 iterations. Once it is done, it is tested on the M&C dataset by computing the vectors for every pair of words. In our results, we found that the network did well for word pairs such as (car, automobile).

Table 2: Comparison of Miller-Charles and Neural Network Results

| word1 | word2 | M&C | NN |
|---|---|---|---|
| car | automobile | 0.89 | 0.86 |
| gem | jewel | 0.90 | 0.74 |
| journey | voyage | 0.93 | 0.66 |
| boy | lad | 0.88 | 0.63 |
| coast | shore | 0.91 | 0.70 |
| asylum | madhouse | 0.89 | 0.66 |
| magician | wizard | 0.90 | 0.75 |
| midday | noon | 0.93 | 0.84 |
| furnace | stove | 0.88 | 0.43 |
| food | fruit | 0.75 | 0.32 |
| bird | cock | 0.71 | 0.66 |
| bird | crane | 0.74 | 0.58 |
| tool | implement | 0.65 | 0.66 |
| brother | monk | 0.63 | 0.57 |
| crane | implement | 0.27 | 0.44 |

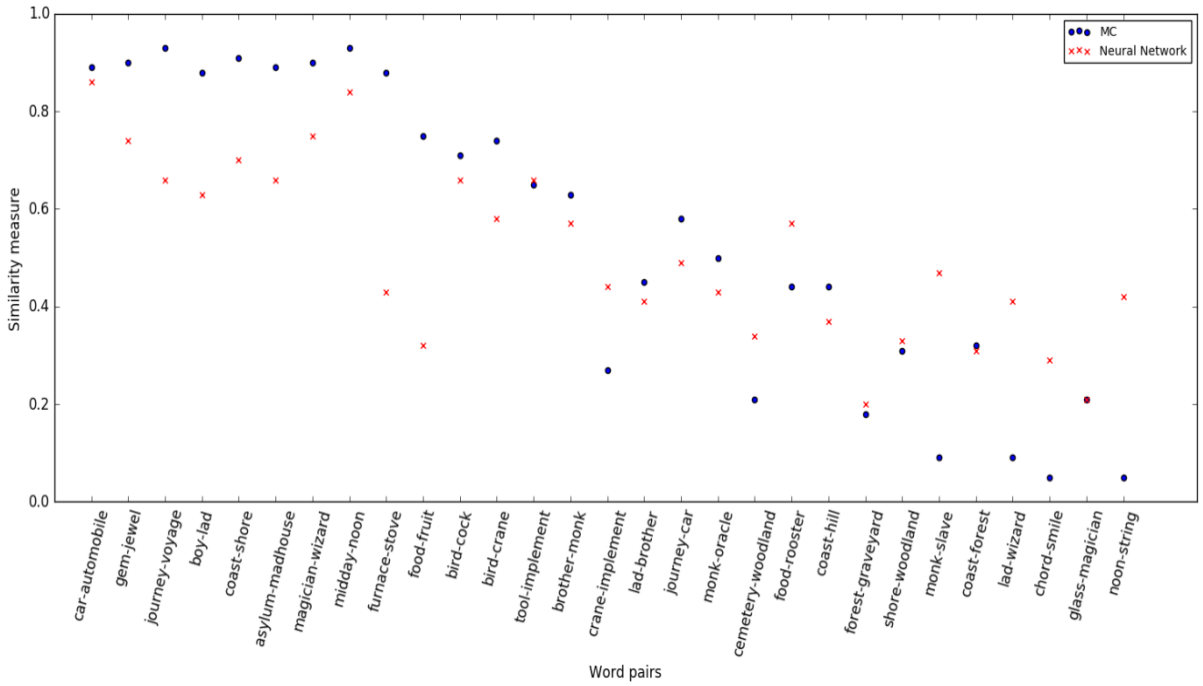


Figure 1: Comparison of MC and Neural Network Results

The neural network learnt features from the given vector representation to fit the dataset to these values, but it is not clear what exactly the neural network has learnt. Table 2 shows the comparison results between M&C and NN for the same data set. Figure 1 is a scatter plot that illustrates the same.

### *C. Linear Regression*

Next we used linear regression to fit a model to predict word similarity between pairs of words. The same method is used to generate the vectors of words as in the neural network.

Let

X = First layer, a Nx8 matrix, where N is the number of samples in the current training batch

Y = Output from dataset, a vector of order N, where each element is the similarity between the two words.

W = A vector of order 8. It is basically a vector where each term is multiplied with the corresponding term in the input vector X

B = A bias term

O = Output of the linear regression model

$O = XW + B$

Then, the loss is calculated. The squared loss error function $L = 1/N \times \Sigma[(Y-O)^2]$ is used.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 09, September - 2017

Although the linear regression model did well for some pairs of words, such as (lad, brother) and (coast, hill), it was objectively worse than the neural network for most pairs of words.

Table 3: Comparison of Miller-Charles and Linear Regression Results

| Word 1 | Word 2 | M&C | LR |
|---|---|---|---|
| lad | brother | 0.45 | 0.42 |
| coast | hill | 0.44 | 0.45 |
| food | fruit | 0.75 | 0.36 |

Linear regression tries to fit the given input data to the required output by fitting a line to the trend. The linear regression model did not fit the word-pair vector to the similarities as well as the neural network. Word-similarity measurement cannot be appropriately modeled used a linear regression model. With the 'sklearn' implementation used for linear regression, the results are unlikely to improve beyond this as there are no tunable parameters for it. Table 3 shows the comparison results between MC and Linear Regression for a subset of the word pairs considered.

### *D. Support Vector Machines*

Support Vector Machines are another popular supervised learning method. They are often used in cases where there is high dimensionality. Support Vector Regression is used to solve regression problems using support vector machines.

In our study, we used a penalty parameter of C=100.0 and the Radial Basis Function kernel. The Support Vector Regression fared worse than linear regression for this problem. Table 4 shows the comparison results between MC and Linear Regression for a subset of the word pairs considered.

Table 4: Comparison of Miller-Charles and Support Vector Machine Results

| Word 1 | Word 2 | M&C | SVR |
|---|---|---|---|
| lad | brother | 0.45 | 0.46 |
| magician | wizard | 0.90 | 0.96 |
| journey | car | 0.58 | 0.21 |

## IV. CONCLUSION

Initially a vector (tuple) was created to represent the words in the above outlined steps. The words were mapped to their properties in the WordNet graph. Then, a matrix was generated with the different word pairs from the training set, using the tuple as the rows. This matrix was fed as training data to the machine learning models. The model was then trained on the matrix. In order to validate the model, the M&C data set was used. The same process of generating the vector was used for the M&C data set as well. The output from the models were then tabulated. For the data set considered, the Neural Network Model gave the best results. The Linear Regression method fared better than the Support Vector Machine with Regression.

## ACKNOWLEDGMENT

I would like to single out and thank my guide Dr. S H Manjula, and my student Karthik Karanth for their immense contribution to executing this work. I thank my family for the cooperation extended by them.

## REFERENCES

[1] Danushka Bollegala, Yutaka Matsuo, Mitsuru, "A Web Search Engine-Based Approach to Measure Semantic Similarity between Words," IEEE Transactions on Knowledge and Data Engineering, vol. 23, no.7, July 2011.

[2] Eneko Agirrea , Carmen Baneab , Daniel Cerd , Mona Diabe, "SemEval-2016 Task 1: Semantic Textual Similarity, Monolingual and Cross-Lingual Evaluation," *University of Michigan, Ann Arbor, MI.*

[3] Asli Eyecioglu, Bill Keller, "ASOBEK at SemEval-2016 Task 1: Sentence Representation with Character N- gram Embeddings for Semantic Textual Similarity," *University of Sussex, Department of Informatics, Brighton, BN19QJ, UK.*

[4] M. Sahami and T. Heilman, "A Web-Based Kernel Function for Measuring the Similarity of Short Text Snippets," *Proceedings of the 15th International World Wide Web Conference*, 2006.

[5] C. Buckley, G. Salton, J. Allan, and A. Singhal, "Automatic Query Expansion Using Smart: Trec 3," *Proceedings of the Third Text REtreival Conference*, pp. 69-80, 1994.

[6] WordSim: alfonseca.org/eng/research/wordsim353.htm

[7] WordNet: https://wordnet.princeton.edu/

[8] Scikit-NeuralNetwork: scikit-learn.org/dev/modules/neural_networks_supervised.htm

## AUTHORS PROFILE

Leena Giri G is currently an Associate Professor in the Department of Computer Science, Dr. Ambedkar Institute of Technology, Bangalore. She obtained her Bachelor of Engineering from SJCE, Mysore. She received her M.Tech Degree in Computer Science and Engineering from IIT Mumbai. Her research interest is in the area of Semantic Web.

Karthik Karanth received his B.E. degree from the Department Computer Science and Engineering, Dr. Ambedkar Institute of Technology, affiliated to Visvesvaraya Technology University, Bangalore. He is currently a Computer Vision Engineer at GreedyGame and designs systems that automatically overlay and generate images, and build models that use deep learning for image classification, among other taska.

S H Manjula is currently an Associate Professor, Department of Computer Science and Engineering, University Visvesvaraya College of Engineering, Bangalore University, Bangalore. She obtained her Bachelor of Engineering and Masters Degree in Computer Science and Engineering from University Visvesvaraya College of Engineering. She was awarded Ph.D in Computer Science from Dr. MGR University, Chennai. Her research interests are in the field of Wireless Sensor Networks and Data mining.

K R Venugopal is currently the Principal, University Visvesvaraya College of Engineering, Bangalore University, Bangalore. He obtained his Bachelor of Engineering from University Visvesvaraya College of Engineering. He received his Masters degree in Computer Science and Automation from Indian Institute of Science Bangalore. He was awarded Ph.D in Economics from Bangalore University and Ph.D in Computer Science from Indian Institute of Technology, Madras. He has a distinguished academic career and has degrees in Electronics, Economics, Law, Business Finance, Public Relations, Communications, Industrial Relations, Computer Science and Journalism. He has authored 31 books on Computer Science and Economics, which include Petrodollar and the World Economy, C

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 09, September - 2017

Aptitude, Mastering C, Microprocessor Programming, Mastering C++ and Digital Circuits and Systems etc.. During his more than three decades of service at UVCE he has over 300 research papers to his credit. His research interests include Computer Networks, Wireless Sensor Networks, Parallel and Distributed Systems, Digital Signal Processing and Data Mining.