Project 13: Semantic similarity

Teemu Rönty
Faculty of Science
University of Oulu
teemu.ronty@gmail.com

Toni Väisänen
Computer Science and Engineering
University of Oulu
vaisanen.toni@gmail.com

Valtteri Vuorio
Computer Science and Engineering
University of Oulu
vvuorio19@edu.oulu.fi

Project source

***Abstract*—Our project demonstrates the effectiveness of different methods for computing semantic similarity between words and sentences. Our approach utilizes Datamuse API, word2vec, Glove, fastText and YAGO to create different representations of words and sentences for similarity comparison. Furthermore, we compare the effectiveness of the methods by calculating the pearson correlation coefficients between calculated similarity scores and human labeled similarity scores in our data sets. Our focus is on the performance of the Datamuse approaches with respect to the other methods. Our results show that the Datamuse approaches perform well in the group and require less from the system running the program.**

I. Introduction

Semantic similarity is a persisting and much studied problem in natural language processing (NLP). Machine based detection of semantic similarity between words, sentences, and documents has a variety of use cases including summarization [1], semantic searching [2], question answering [3], document classification [4], sentiment analysis [5] [6], plagiarism detection [7] and query expansion [8]. The first attempts to measure similarity focused on methods such as character wise matching, bag-of-words and term frequency - inverse document frequency (TF-IDF). These methods however, fail in many cases, as they only consider the relative frequency of syntactic elements. Take for example the sentences

- ”Bob needs to visit the hospital.”
- ”Bob should see a doctor.”
- ”The hospital needs Bob.”

Intuitively, sentences 1 and 2 convey similar meaning, while sentences 1 and 3 may have completely different meanings. However, a frequency-based method usually gives the result that sentences 1 and 3 are more similar than 1 and 2. This should make it clear that more advanced methods are needed in order to arrive at a reliable measure for semantic similarity. In recent times, great effort has been made towards achieving this goal, as can be seen from the current state of the art summarized in figure 1.

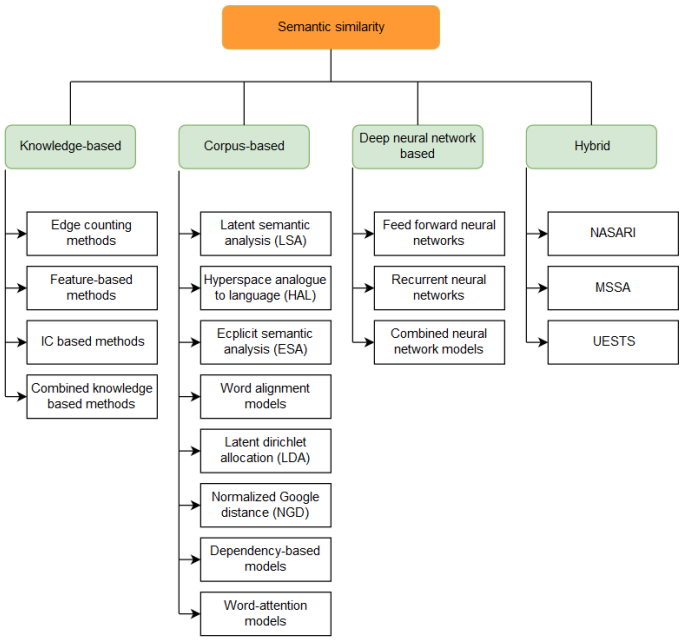

Figure 1. Approaches to semantic similarity. Adapted from Chandrasekaran and Mago [9].

Chandrasekaran and Mago [9] roughly divide different approaches of measuring semantic similarity to the four classes presented in figure 1. We shortly describe the key aspects of knowledge-based approaches and corpus-based approaches since they are most relevant in terms of this report. The descriptions are based on Chandrasekaran and Mago [9] unless mentioned otherwise.

A. Different approaches

Knowledge-based approaches rely on knowledge sources such as ontologies, lexical databases, dictionaries and thesauri that provide information for similarity measurements. There are several methods to obtain similarity measures from these databases. The knowledge-based methods described in this report are edge-counting methods, feature-based methods and information content-based methods.

Egde-counting methods view an ontology as a graph of taxonomically connected elements. Most edge-counting similarity measures rely on the shortest distance between the words in the graph and/or the depth of the least common subsumer (LCS). In this framework, words that have short paths between them are more similar than words with longer paths between them.

Feature-based measures use properties of the words to obtain the semantic similarity. The properties can for example be dictionary descriptions (e.g. glossary) or neighbouring concepts of the word. The key idea for similarity is based on properties shared between the words. The properties that are shared by both words increase the similarity and properties that are associated only with one of the words lower the similarity measure.

Information content-based methods compute the similarity based on the information content (IC) of the LCS and/or the IC of the individual words. The IC is a measure of the specifity of the word. IC is based on the assumption that a more specific word occurs less frequently and is thus based on the IDF. Intuitively, the logic seems to work: if two words and their LCS all have high specificity, they most likely are highly related. On the contrary, if either of the words and/or the LCS have low IC the words could be assumed to be less related.

Corpus-based approaches use corpora to obtain information for similarity measuring. The supporting idea is that similar words tend to co-occur in documents. One flaw of these approaches is that they tend to neglect the actual meaning of the word. The corpus-based approaches use the corpora to learn latent relationships between the words.

Again, there are several published methods for corpus-based approaches to calculate the semantic similarity. We shortly discuss only a method based on word embeddings due to relatedness to the project. Schnabel, Labutov, Mimno, and Joachims [10] explain that word embeddings create a vector representation for each word. A well performed word embedding provides vectors that have relationships similar to the linguistic relationships between the words. There are different methods to obtain the vector representations, and depending on the corpora and the method, the process can be tedious. For easier implementation there are pre-trained word embeddings available. Most commonly used pre-trained embedding tools include ***word2vec***, ***GloVe*** and ***fastText***.

We leave out the discussion and description of deep neural network based models and hybrid models. For a more detailed discussion of the approaches listed in figure 1 and the development of the field, see Chandrasekaran and Mago [9].

B. Our Approach

The main focus of our project is to compare the semantic similarity measures computed from the output of Datamuse API to other methods. Datamuse API is a query engine designed for developers. Datamuse API allows for easy access to word-finding and it has uses in applications such as auto completion and search query suggestions. There is a variety of options for querying that for example allow for matching words on similar meaning, spelling and pronunciation. Datamuse forms its similarity output for a word by using the famous knowledge-base WordNet 3.0 and several online dictionaries that are found with OneLook Dictionary Search. Thus, our approach, that determines the semantic similarity based on Datamuse outputs, is a knowledge-based approach. Furthermore, we employ a feature-based method to calculate the similarity for two words and further extend our methods to compare the semantic similarity between sentences. The methods that are tested along the Datamuse approaches are word embeddings, namely word2vec, GloVe and fastText, and an IC-based method, that is constructed YAGO classes obtained from Sematch.

Machine-based semantic similarity scores are often validated by comparing the correlation of the score with human-evaluated scores. The intuition is, that a better machine-based measure would correlate more highly with the human evaluated scores. Some datasets that contain human evaluated similarity scores for word and sentence pairs have become an industry standard in benchmarking. In this project, we use the somewhat famous datasets presented in Table 1.

Table 1: Datasets used in the project.

Name	Total pairs	Year	Reference
MC	30	1965	[11]
RG	65	1991	[12]
Wordsim	353	2002	[13]
STSS	131	2013	[14]

State-of-the-art performances of ontology based models and word embeddings from several studies is presented by Lastra-Diaz, Goikoetxea, Hadj Taieb et al. [15]. It appears that the state-of-the-art correlations for MC and RG data above 0.9. With word embeddings correlations above 0.7 have been reported for WordSim data [16].

The goal of this project is to see and demonstrate the capability of an easily implementable Datamuse approach to obtain similarity for words and sentences. Our method is evaluated by calculating the Pearson correlation coefficients between our similarity scores and the human-evaluated similarity scores, for the data sets listed in Table 1. The correlations of our approaches are compared to the correlations provided by the benchmarks. The project is implemented with Python 3 and Python 2 using standard NLP, scientific computing and machine learning libraries, in addition to the built-in Python libraries. Finally, we present a graphical user interface that demonstrates our findings.

II. Methodology

We aim to construct a feature-based similarity measure for words and sentences on knowledge-based information retrieved from Datamuse API. We will first describe the data retrieving process and the use of Datamuse API and then proceed to introduce the used similarity measures. In addition, we discuss the methods that we use for hyper parameter optimization. Finally, we present the benchmark methods and explain the procedures and configurations used for obtaining their similarity measures.

A. Data retrieval

Datamuse API can be used to get a set of similar words for a given word based on several criteria, e.g. similarity based on ”sounds like”, ”means like” etc. We are interested in words

with similar meaning with the compared word thus the "ml" query as in "means like" is used for data retrieval.

Datamuse query returns a set of words with similar meaning as an output. Query results from the API are used as the related words or as the representation of a word at later stages. The amount of output words to use to form the representation of the word is a hyper parameter that will be experimentally optimized further on.

To gain direct access to the Datamuse query results, we fetch all the needed results in advance and store them to the disk. In order to sustain the capability to optimize the size of the representation of the word, we do this query with a very large number of outputs. These word sets are later loaded into a Pandas data frame for further analysis.

B. Similarity measures

One semantic similarity method used in the context of this work is Jaccard´s similarity. It describes the similarity of two sets with a value bound to the interval $[0, 1]$. The value is calculated by dividing the count of the elements in the intersection of set A and set B, and dividing it with the count of the elements in the union of the same sets. In this context the sets are collections of words that are related to query words. Jaccard similarity between two words can be expressed mathematically as:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}, \tag{1}$$

where A is the set of related words of word a and B is the set of related words of word b. We extend this concept to sentence level similarity measures in two different ways. We call these different approaches the union method and the overlap method. Mathematically, the union method expression of sentence similarity is:

$$J_u(A_s, B_s) = \frac{|\bigcup_{i=1}^{wc_a} A_i \cap \bigcup_{i=1}^{wc_b} B_i|}{|\bigcup_{i=1}^{wc_a} A_i \cup \bigcup_{i=1}^{wc_b} B_i|}, \tag{2}$$

where the comparison sets A_s, B_s are unions of the sets of related words for all the words of a sentence. The overlap representation for sentence level similarity is:

$$J_o(A_s, B_s) = \frac{|\bigcap_{i=1}^{wc_a} A_i \cap \bigcap_{i=1}^{wc_b} B_i|}{|\bigcap_{i=1}^{wc_a} A_i \cup \bigcap_{i=1}^{wc_b} B_i|}, \tag{3}$$

where A_s, B_s are the intersection of the sets of the related words for all words in the sentence.

We also test another method based on the Datamuse output. In this method, we take into account the fact that the lists of related words obtained from a sentence may contain multiple instances of the same word. The frequency of each related word is placed into a frequency vector, also known as a histogram. The appropriate similarity measure to be used between the frequency vectors is the cosine similarity, which can be described as:

$$sim_{cos}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}|| \cdot ||\mathbf{B}||}, \tag{4}$$

where $\mathbf{A}$, $\mathbf{B}$ are the frequency vectors of sentences A and B, and $\cdot$ denotes the dot product and $||\mathbf{x}||$ is the Euclidean norm of vector $\mathbf{x}$. We call the method of computing cosine similarity between frequency vectors the histogram method (since histograms plot frequencies). This method is a somewhat similar to word embedding methods, since it represents words/sentences as vectors. However, it is not corpus-based like for example word2vec but relys on the knowledge in Datamuse output.

C. Parameter optimization

The M28 data contains 28 pairs of words and the average of human evaluated similarity scores as a measure of true similarity between the words. The data consists of tuples (w_1, w_2, s_h), where w_i are the words and s_h is the human judged similarity score.

Now with our Datamuse approach, the words are represented as sets of related words. The size of these sets is optimized the following way. We compute the Jaccard similarity for each of MC word pairs with each word represented by a set of N related words. The calculation is done for $N = 1, 2, ..., 100$. This results in N different series of Jaccard similarity scores. For each of the Jaccard series the pearson correlation coefficient with the human judgement scores is calculated. The sample pearson correlation coefficient is calculated as:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}, \tag{5}$$

where $\bar{x}$ is the sample mean of x. The optimal value for n is determined by the highest correlation. The found optimum is used in further parts of the project. The optimal n is also computed based on the RG data and WordSim data but are not used later on. These additional computations are done just to show that the optimum for n might be sensitive to the data used as the base.

D. Extending similarity measures to sentences

This project aims to extend the similarity comparison from word level to sentence level. For sentece level similarity we use STSS data that consists of sentence pairs and human evaluated similarity scores. Sentences, however, require some processing before the presented methods are applicable.

In equations (2) and (3), a sentence is represented as the union or the intersection of all the representative sets of its tokens. The tokens are obtained in a twofold process. First, the sentence is tokenized using the word_tokenize function of nltk Python library. Effectively, the sentence is transformed into an array of string tokens. Second, common stop words are removed from the array of tokens by using a stop word

list provided by the nltk library. The process is demonstrated in the following example:

```
1. "The dog is happy."
2. "The", "dog", "is", "happy"
3. "dog", "happy"
```

For each token in the processed sentence (3. part of the example), the set of related words is retrieved. Now, the sentence can be presented either as the union or the intersection of all the sets of related words. Depending on the approach, the similarity between sentences is calculated with equations (2) or (3). Note that we add the original tokens of a sentence to the set of it's related words before calculating the similarity score.

For the histogram method, we use a somewhat similar approach. The retrieved related words are not converted to a set, i.e. one related word can occur several times in the sentence representation. The occurrences are counted to form a frequency vector. The similarity for two sentences is calculated with equation (4) by using the frequency vectors.

E. Word embeddings and YAGO classes

Performance of the methods presented in the previous sections are compared to existing corpus based methods. In addition, we form an IC based method. The corpus-based methods limit to word embeddings, namely word2vec, GloVe and fastText. The IC-based method is one relying on YAGO classes with IC-scores. The semantic similarity is calculated for STSS sentence pairs using these alternative methods and the correlation with human judgement is used to represent their performance. The correlations are then compared to the results of the Datamuse approaches.

With the word embedding methods, pre-trained word vector representations are used to calculate the similarity score for each of the sentences in STSS. The similarity score is obtained with GenSim Python library that provides a function for calculating similarities between sets of word vectors $\mathbf{X}$ and $\mathbf{Y}$. The implementation of the similarity is the cosine similarity between the average of the word vectors:

$$sim_{gensim} = \frac{(\frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_i)\cdot(\frac{1}{M}\sum_{i=1}^{M}\mathbf{Y}_i)}{||\frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_i||\cdot||\frac{1}{M}\sum_{i=1}^{M}\mathbf{Y}_i||} \quad (6)$$

Sentence similarities are calculated with all the word embedding methods. Table 2 lists the methods and the used pre-trained data files that contain the vector representations.

Table 2: Pretrained word embeddings

Method	Used data file
FastText	wiki-news-300d-1M[1]
Glove	glove.6B [2]
Word2Vec	GoogleNews-vectors-negative300 [3]

[1]https://dll.fbaipublicfiles.com/fasttext/vectors-english/wiki-news-300d-1M.vec.zip

[2]http://nlp.stanford.edu/data/glove.6B.zip

[3]https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM

The IC-based method is built on top of Sematch and the YAGO classes that it provides. The YAGO classes come with predefined IC-scores. Sematch also provides a function (yago_similarity) for computing the semantic similarity between two classes.

For sentence level similarity calculation, the sentences are tokenized and the tokens are mapped to YAGO classes. I.e. the sentence "Would you like to go out to drink with me tonight?" would result in the following set of YAGO classes that would be the sentence´s YAGO representation

```
http://dbpedia.org/class/yago/Drink107885223
http://dbpedia.org/class/yago/Go115292069
http://dbpedia.org/class/yago/Like105845888
http://dbpedia.org/class/yago/Tonight115263045
```

The similarity of YAGO sentence representations Sj is calculated by first comparing the class pairs in $[(Sj_1)_n] \times Sj_2$ resulting in a map of similarity score vectors as demostrated in the following exercise.

```
1
2  let classes = c1, c2, c3, c4, c5
3
4  let s1 = [c1,c2]
5  let s2 = [c3,c4,c5]
6
7  s1_score_map = {
8      c1: [ score(c1,c3)
9          , score(c1,c4)
10         , score(c1,c5)
11         ]
12     c2: [ score(c2,c3)
13         , score(c2,c4)
14         , score(c2,c5)
15         ]
16 }
17
18 s1_score_map = {
19     c3: [ score(c3,c1)
20         , score(c3,c2)
21         ]
22     c4: [ score(c4,c1)
23         , score(c4,c2)
24         ]
25     c5: [ score(c5,c1)
26         , score(c5,c2)
27         ]
28 }
29
```

For each class the highest score is selected from the matching array to represent the similarity of that class wrt. to Sj_n. The mean of the maximum values is selected to represent one directional similarity between the two sentences. Both of these one directional similarities are computed and then averaged to give the final sentence similarity score.

```
1
2  s1_to_s2_score = mean([
3      max(s) for s
4      in s1_score_map
5  ])
6  s2_to_s1_score = mean([
7      max(s) for s
```

```
    in s2_score_map
])

final_score = mean([
    s1_to_s2_score,
    s2_to_s1_score
])
```

III. Results and discussion

A. Semantic similarity between words

As explained in the previous section, using equation (1) and pair-wise lists of related words retrieved from Datamuse API, we calculated similarity scores for each pair of words and compared it against human judgment scores. We found that for MC the optimal amount of related words retrieved was 25 (figure 2.). The correlation values are listed in table 2, including the maximum correlation value for the range N = 1, 2, ... , 100.

Table 3: Correlation values for MC, RG and WordSim

Name	Correlation (N=25)	Correlation (N=argmax)
MC	0.72	0.72
RG	0.67	0.68
WordSim	0.37	0.42

Figures 2, 3, and 4 illustrate the variation of correlation as the number of related words increases. The vertical line highlights the number of outputs that results in the highest correlation.

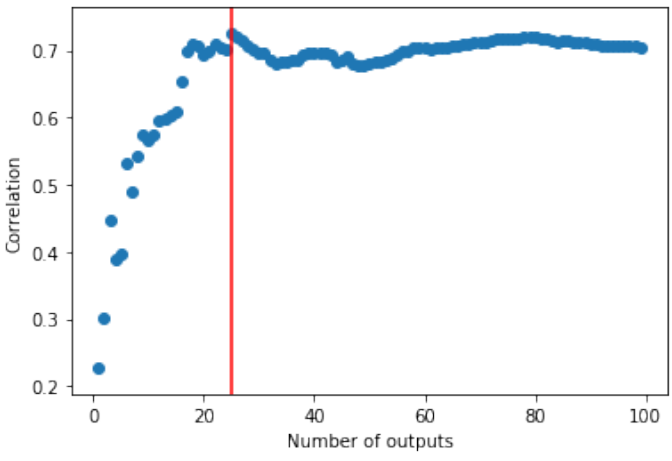

Figure 2. Improvement of correlation by increasing Datamuse output for MC data.

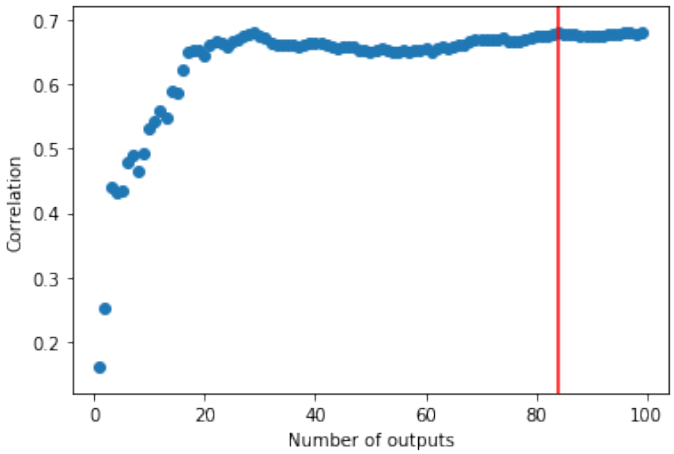

Figure 3. Improvement of correlation by increasing Datamuse output for RG data.

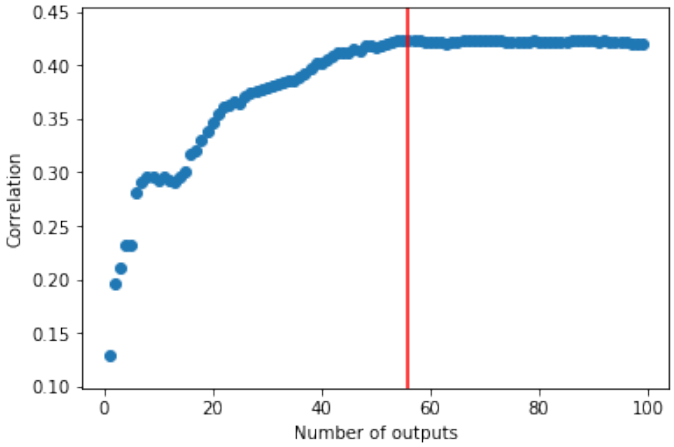

Figure 4. Improvement of correlation by increasing Datamuse output for Wordsim data.

The correlation score quickly rises as N is increased, and then slows down significantly. It may be possible to get marginally better results for $N > 100$, but taking into account the fact that it is much faster to transmit data from Datamuse with a lower value of N, it is reasonable to work with a local maximum.

There also seems to be much variation in the performance of the Datamuse approach with the different data sets. With the more recent WordSim data, the correlation falls short of correlation with MC and RG. This is true with both $N = 25$ and the optimum N from range $[1, 100]$. To form generalizations of the performance, a larger data of human labelled similarity scores and word pairs would be needed.

B. Semantic similarity between sentences

For sentence-wise similarity, using equations (2), (3), (4) and the related sentence representations, we calculated similarity scores for each pair of sentences and compared it against human judgment. The correlation values are listed in table 3 for each respective method, including the maximum correlation value for the range N = 1, 2, ... , 1000.

Table 3: Correlation values for STSS

Method	Argmax(N)	Correlation(Argmax(N))
Overlap	0	0.694
Union	825	0.744
Histogram	825	0.814

Figures 5, 6, and 7 illustrate the variation of correlation as the number of related words increases.

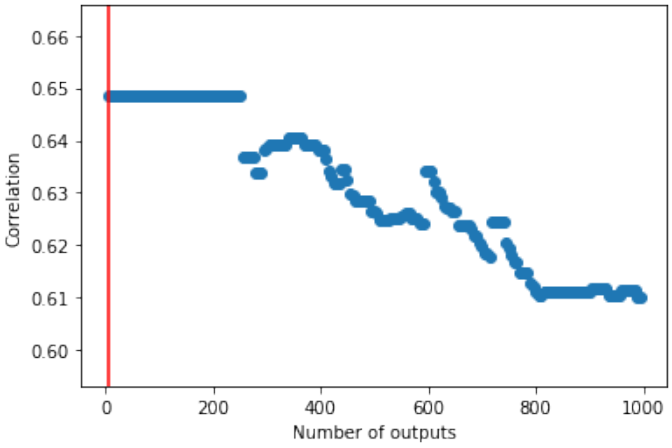

Figure 5: Improvement of correlation by increasing Datamuse output for STSS data, using the overlap method as in equation (3).

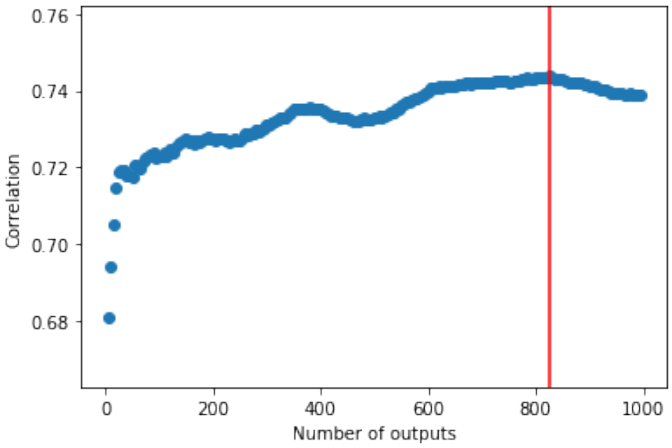

Figure 6: Improvement of correlation by increasing Datamuse output for STSS data, using the union method as in equation (2).

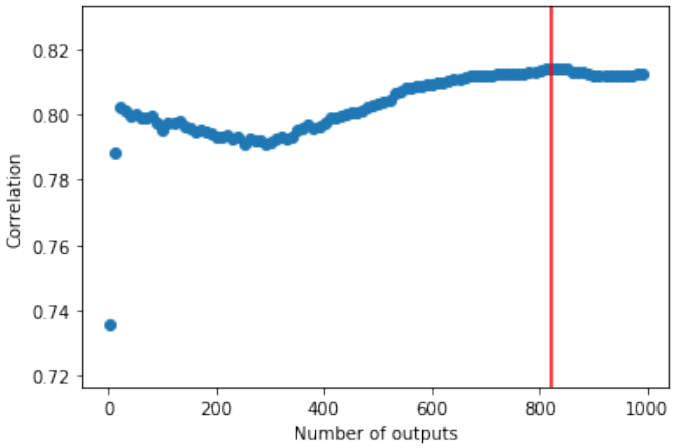

Figure 7: Improvement of correlation by increasing Datamuse output for STSS data, using the histogram method as in equation (4).

As in the results between word pairs, the correlation score quickly rises as N is increased. Using a value of $N > 100$ we gain at most a marginal increase in correlation. Overall, the results seem promising.

C. *Word embeddings and YAGO concept similarities*

Using the pre-trained word vectors and equation (5) we obtained correlation values for the STSS data set, listed in table 4. The last row shows the result for the IC-based YAGO similarity.

Table 4: Correlation values for the approaches based on Word2vec, Glove, Fasttext and YAGO classes.

Method	Correlation
word2vec	0.771
GloVe	0.656
fastText	0.729
YAGO	0.641

D. *Comparison of results for sentence-wise similarity*

Our methods achieve a slightly higher correlation score compared to the word2vec, glove and fastText approach. However, it is important to note that we used pre-trained models, and the vocabulary of the pre-trained models did not include some of the words present in the STSS data set. In this case we simply ignored the word vector that was missing from the vocabulary. The key results are compared in table 5.

The rationale behind each Datamuse-based method performing as shown is not entirely clear. It is clear to place importance on related words that coexist between sentences, but not so clear as how they should be weighted. The union method places equal weight for all related words obtained from a sentence. In contrast, with the overlap method we seem to end up with a relatively small set of specialized words that rarely coexist between sentence pairs. The histogram method is sort of a balance between these two extrema by weighing the words based on frequency.

Table 5: Comparison of the key results.

Method	Correlation
Ours	0.814
word2vec	0.771
fastText	0.729
GloVe	0.656
YAGO	0.641

E. Graphical user interface

We created a graphical user interface that demonstrates the calculation of various similarity scores used in this project.

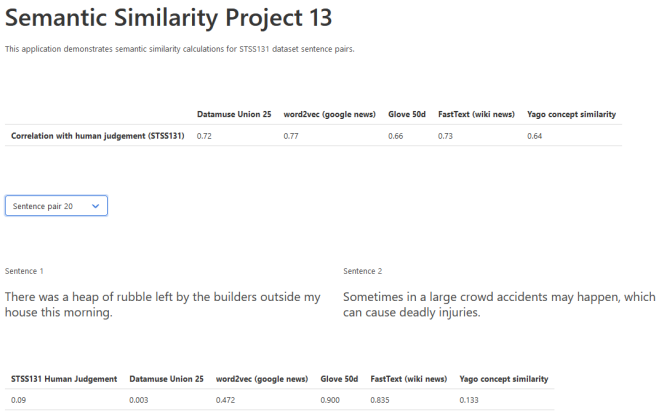

Figure 8: Screenshot from our GUI.

IV. Overall discussion

Overall, the Datamuse approach seems to provide a reasonable way to implement practical applications that require semantic similarity detection. Our approach is simple to implement and does not require heavy processes running the calculations. Using Datamuse requires minimal computational resources but as a downside the data must be transmitted over a network and it has some limitations on the amount of allowed requests.

In the process of implementing this approach we discovered some directions that may be explored to improve the results:

- The list of related words returned from Datamuse API is ordered in terms of relevance, but our methods treat each word as equally relevant. Further work may consider using this fact to derive an improved method.
- As discussed in section 3D, future works may consider finding a better alternative to the overlap, union and histogram methods.
- To query a large number of related words from Datamuse, another approach would be to do the process recursively, i.e. fetch a list of related words, and for each related words we fetch their related words, and so on.

V. Conclusion

In conclusion, we have demonstrated the effectiveness of computing semantic similarity between words and sentences by utilizing the output of Datamuse API and implemented other methods to compare our results (word2vec, Glove, fastText and YAGO). The effectiveness of our approach was evaluated with the pearson correlation coefficient between calculated similarity scores and human labeled similarity scores in our data sets. Our results show the Datamuse approach seems to provide a reasonable way to implement practical applications that require semantic similarity detection. We created a graphical user interface that demonstrates the calculation of similarity scores for the above mentioned methods.

References

[1] Wang, Li, Zhu, and Ding. "Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization", In Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '08), 2008, Association for Computing Machinery, New York, NY, USA, pp. 307–314. doi: 10.1145/1390334.1390387

[2] Zhong, Zhu, Li, Yu. "Conceptual Graph Matching for Semantic Search", Conceptual Structures: Integration and Interfaces, 2002, Springer, Berlin, Heidelberg, pp. 92-106, doi: 10.1007/3-540-45483-7_8

[3] Yih, He and Meek. "Semantic parsing for single-relation question answering", In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Volume 2: Short Papers, 2014, pp. 643-648, doi: 10.3115/v1/p14-2105

[4] Zongda Wu, Hui Zhu, Guiling Li, Zongmin Cui, Hui Huang, Jun Li, Enhong Chen, Guandong Xu. "An efficient Wikipedia semantic matching approach to text document classification", Information Sciences, Volume 393, 2017, pp. 15-28, doi: 10.1016/j.ins.2017.02.009.

[5] Oscar Araque, Ganggao Zhu, Carlos A. Iglesias. "A semantic similarity-based perspective of affect lexicons for sentiment analysis", Knowledge-Based Systems, Volume 165, 2019, pp. 346-359, doi: 10.1016/j.knosys.2018.12.005

[6] S. Poria, I. Chaturvedi, E. Cambria and F. Bisio. "Sentic LDA: Improving on LDA with semantic similarity for aspect-based sentiment analysis", 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, 2016, pp. 4465-4473, doi: 10.1109/IJCNN.2016.7727784.

[7] D. Gupta, V. K and C. K. Singh. "Using Natural Language Processing techniques and fuzzy-semantic similarity for automatic external plagiarism detection". 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI), New Delhi, 2014, pp. 2694-2699, doi: 10.1109/ICACCI.2014.6968314

[8] Gallant, Isah, Zulkernine, and Khan. "Xu: An Automated Query Expansion and Optimization Tool". 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), 2019, pp. 443-452, doi: 10.1109/COMPSAC.2019.00070

[9] Chandrasekaran, Dhivya, and Vijay Mago. "Evolution of Semantic Similarity–A Survey." arXiv preprint arXiv:2004.13820 (2020).

[10] Schnabel, Labutov, Mimno, and Joachims. "Evaluation methods for unsupervised word embeddings", In Proceedings of the 2015 conference on empirical methods in natural language processing, 2015, Association for Computational Linguistics, Lisbon, Portugal, pp. 298–307, doi:10.18653/v1/D15-1036

[11] Miller, George A., and Walter G. Charles. "Contextual correlates of semantic similarity."

[12] Rubenstein, H., and Goodenough, J. B. (1965). "Contextual correlates of synonymy."

[13] Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G. et Ruppin, E. (2001). "Placing search in context : The concept revisited"

[14] O'shea, James, Zuhair Bandar, and Keeley Crockett. "A new benchmark dataset with production methodology for short text semantic similarity algorithms."

[15] Lastra-Diaz, Goikoetxea, Hadj Taieb et al. "A large reproducible benchmark of ontology-based methods and word embeddings for word similarity". Information Systems, 2020, doi:10.1016/j.is.2020.101636

[16] Toshevska, Stojanovska, Kalajdjieski. "Comparative Analysis of Word Embeddings for Capturing Word Similarities", 6th International Conference on Natural Language Processing (NATP 2020), doi:10.5121/csit.2020.100402