SEMI-SUPERVISED LABEL PROPAGATION FOR IDENTIFICATION OF DERIVATIONAL NOUNS IN ENGLISH

Submitted by:

Satanik Mitra (17IM91R04)

Samridha Kumar (16CS71P03)

Project Mentor:

Amrith Krishna

Faculty:

Prof. Niloy Ganguly

ABSTRACT

Derivational nouns are widely used in the English language and is a prevalent means of productivity in the language. Derivational nouns are derived by affixing suffixes to the base form of verbs. For example, affixing *-er* to the base form of the verb *learn* results in the noun *learner*, and affixing *-or* to the base form of the verb *interrogate* results in the noun *interrogator.*

However, affixes that modify the mere morphological or syntactical role of a word in its usage are not considered derivational, rather inflectional.

Identifying derived words from a corpus is challenging. Usage of pattern matching approaches in strings are often inept for the task. Tasks that rely on string matching approaches alone, often result in a large number of false positives. For example, while the word 'postal' is generated from 'post', the word 'canal' is not generated from 'can'. String matching approaches often result in low recall as well, due to the variations in patterns in the derived and source word pairs, even for the same affix. Both 'postal' and 'minimal' are derived using the affix 'al', but the source word for postal is 'post', while the source word for minimal is 'minimum'.

In our work, we propose an approach for the identification and analysis of derivational nouns in the English Corpora. We obtain the rules for generation of derivational nouns and combine the diverse features ranging from them with the word embeddings for the candidate words. The lack of labelled data and other resources have led us to propose a graph based semi-supervised label propagation approach, using Modified Adsorption (MAD), to identify the usage of derivational nouns in a corpus and map them to their corresponding source words. The label propagation can be summazsed as:

1. The nodes of the graph represent the labelled/unlabelled data points, and the weight of the edges is the measure of closeness or proximity with the other node.

2. Lables of a node propagate to all other nodes through edges.

3. Larger edge weights allow labels to travel through easier.

By leveraging on the rules obtained, we not only find the different pattern differences between the source and the derived word pairs, but also group pattern that are likely to emerge from the same affixes.

CONTENTS

		Page No
1.	**OBJECTIVES**	**3**
2.	**DESCRIPTION**	**4**
3.	**METHODOLOGY**	**5-8**
4.	**RESULTS**	**9**
5.	**CONCLUSION**	**10**

OBJECTIVE

The entire work done can be broadly categorized into two objective goals. Our work has focused on achieving these objective goals.

1. Find the different pattern differences between source and derived word pairs, and group patterns from same affixes.
2. Assessment of MODIFIED ADSORPTION (MAD) algorithm with respect to prediction of derivational affixes.

The basic challenges faced in the process of achieving the objectives were:

1. Detection of False Positives (FP):

   There are some word pairs which though bear semblance to the derivational affix pattern, but are actually not *'derivational'*.

   e.g. *'Canal'* is not derived from the root *'Can'* affixing the morpheme *'-al'*.

   Our model was so trained such that these word pairs were identified as False Positives.

2. Correcting Low Recall:

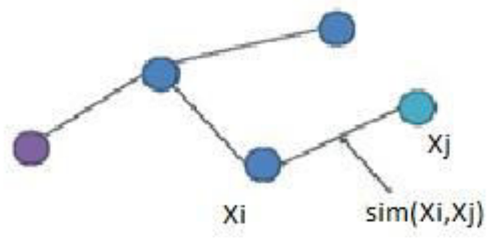
   There are some derivational words which may bear the same affix but the source word is not similar.

   e.g. *'Post-al'* is derived from *'Post'*, but *'Minim-al'* is derived from *'Minimum'*.

   Our work has focused on detecting these word pairs which give a Low Recall to the trained model, and matching them to their corresponding correct source word.

DESCRIPTION

For the purpose of our work, we have adopted a Graph based Semi Supervised Learning (SSL) Approach for transductive learning.
Semi-supervised Learning was chosen because of the lack of labeled data. Semi-supervised algorithms learn from limited amounts of labeled data and vast amounts of freely available unannotated data. SSL also helps in generation of a more accurate decision boundary.
In a Graph based SSL, each example is associated with a vertex in an undirected graph and a weighted edge between two vertices represents similarity between the two corresponding examples. Graph-based semi-supervised algorithms often propagate the label information from the few labeled vertices to the entire graph. Most of the algorithms trade-off between *accuracy* (initially labeled nodes should retain those labels, relaxations allowed by some methods) with *smoothness* (adjacent vertices in the graph should be assigned similar labels). In a transductive setting, the algorithms output label information to the unlabeled data.
Among the various Graph based SSL algorithms present, we have chosen **Modified Adsorption** for the purpose of our work. It performs the multiclass classification required for our task, and can be parallelized and scaled to handle large data.
For our work, we have used the *Junto Label Propagation Toolkit* which contains an implementation of Modified Adsorption.



There are two stages in the complete Modified Adsorption algorithm:

1. Graph Construction
2. Label Inference

The next section describes the two stages and the entire methodology in detail.

METHODOLOGY & EXPERIMENTS

This section describes in detail the entire methodology adopted for the purpose of our work. The work-flow can be divided into two categories:

1. Graph Construction
2. Label Inference

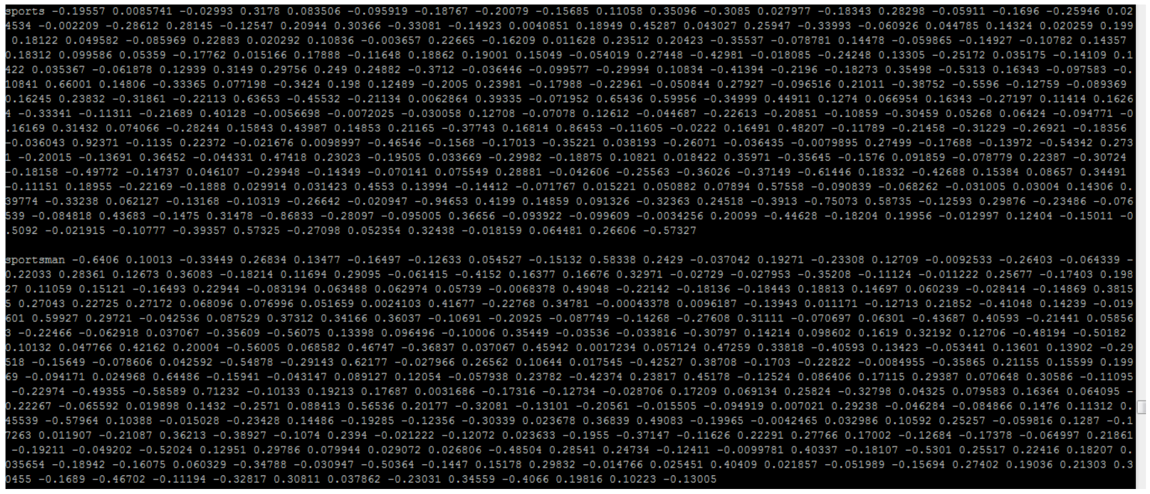
Graph Construction

STEP 1: Extraction of source word – derived word pairs from the English Corpus.

A total of 982 source word – derived word pairs have been extracted from the English corpora. These word pairs consist of a mixture of derivational affixes, namely: *-ally, -ism, -ist, -ic, -y, -man, -like, -hood, -ation.* Each affix group consists of 98 word pairs, apart from the group corresponding to affixes *-ist* and *-y* which consist of 99 word pairs.

STEP 2: Representation of the word pairs as word vectors.

Each word of the word pair has been represented by a word vector of length 300. The word vectors for each word were derived using the *fastText* library. Facebook has released pre-trained fastText 300-dimensional word vectors for over 90 languages and our work makes use of that.



```
sports -0.19557 0.0085741 -0.02993 0.3178 0.083506 -0.095919 -0.18767 -0.20079 -0.15685 0.11058 0.35096 -0.3085 0.027977 -0.18343 0.28298 -0.05911 -0.1696 -0.25946 0.02
4534 -0.002209 -0.28612 0.28145 -0.12547 0.20944 0.30366 -0.33081 -0.14923 0.0040851 0.18949 0.45287 0.043027 0.25947 -0.33993 -0.060926 0.044785 0.14324 0.020259 0.199
 0.18122 0.049582 -0.085969 0.22883 0.020292 0.10836 -0.003657 0.22665 -0.16209 0.011628 0.23512 0.20423 -0.35537 -0.078781 0.14478 -0.059865 -0.14927 -0.10782 0.14357
0.18312 0.099586 0.05359 -0.17762 0.015166 0.17888 -0.11648 0.18862 0.19001 0.15049 -0.054019 0.27448 -0.42981 -0.018085 -0.24248 0.13305 -0.25172 0.035175 -0.14109 0.1
422 0.035367 -0.061878 0.12939 0.3149 0.29756 0.249 0.24882 -0.3712 -0.036446 -0.099577 -0.29994 0.10834 -0.41394 -0.2196 -0.18273 0.35498 -0.5313 0.16343 -0.097583 -0.
10841 0.66001 0.14806 -0.33365 0.077198 -0.3424 0.198 0.12489 -0.2005 0.23981 -0.17988 -0.22961 -0.050844 0.27927 -0.096516 0.21011 -0.38752 -0.5596 -0.12759 -0.089369
0.16245 0.23832 -0.31861 -0.22113 0.63653 -0.45532 -0.21134 0.0062864 0.39335 -0.071952 0.65436 0.59956 -0.34999 0.44911 0.1274 0.066954 0.16343 -0.27197 0.11414 0.1626
4 -0.33341 -0.11311 -0.21689 0.40128 -0.0056698 -0.0072025 -0.030058 0.12708 -0.07078 0.12612 -0.044687 -0.22613 -0.20851 -0.10859 -0.30459 0.05268 0.06424 -0.094771 -0
.16169 0.31432 0.074066 -0.28244 0.15843 0.43987 0.14853 0.21165 -0.37743 0.16814 0.86453 -0.11605 -0.0222 0.16491 0.48207 -0.11789 -0.21458 -0.31229 -0.26921 -0.18356
-0.036043 0.92371 -0.1135 0.22372 -0.021676 0.0098997 -0.46546 -0.1568 -0.17013 -0.35221 0.038193 -0.26071 -0.036435 -0.0079895 0.27499 -0.17688 -0.13972 -0.54342 0.273
1 -0.20015 -0.13691 0.36452 -0.044331 0.47418 0.23023 -0.19505 0.033669 -0.29982 -0.18875 0.10821 0.018422 0.35971 -0.35645 -0.1576 0.091859 -0.078779 0.22387 -0.30724
-0.18158 -0.49772 -0.14737 0.046107 -0.29948 -0.14349 -0.070141 0.075549 0.28881 -0.042606 -0.25563 -0.36026 -0.37149 -0.61446 0.18332 -0.42688 0.15384 0.08657 0.34491
-0.11151 0.18955 -0.22169 -0.1888 0.029914 0.031423 0.4553 0.13994 -0.14412 -0.071767 0.015221 0.050882 0.07894 0.57558 -0.090839 -0.068262 -0.031005 0.03004 0.14306 0.
39774 -0.33238 0.062127 -0.13168 -0.10319 -0.26642 -0.020947 -0.94653 0.4199 0.14859 0.091326 -0.32363 0.24518 -0.3913 -0.75073 0.58735 -0.12593 0.29876 -0.23486 -0.076
539 -0.084818 0.43683 -0.1475 0.31478 -0.86833 -0.28097 -0.095005 0.36656 -0.093922 -0.099609 -0.0034256 0.20099 -0.44628 -0.18204 0.19956 -0.012997 0.12404 -0.15011 -0
.5092 -0.021915 -0.10777 -0.39357 0.57325 -0.27098 0.052354 0.32438 -0.018159 0.064481 0.26606 -0.57327

sportsman -0.6406 0.10013 -0.33449 0.26834 0.13477 -0.16497 -0.12633 0.054527 -0.15132 0.58338 0.2429 -0.037042 0.19271 -0.23308 0.12709 -0.0092533 -0.26403 -0.064339 -
0.22033 0.28361 0.12673 0.36083 -0.18214 0.11694 0.29095 -0.061415 -0.4152 0.16377 0.16676 0.32971 -0.02729 -0.027953 -0.35208 -0.11124 -0.011222 0.25677 -0.17403 0.198
27 0.11059 0.15121 -0.16493 0.22944 -0.083194 0.063488 0.062974 0.05739 -0.0068378 0.49048 -0.22142 -0.18136 -0.18443 0.18813 0.14697 0.060239 -0.028414 -0.14869 0.3815
5 0.27043 0.22725 0.27172 0.068096 0.076996 0.051659 0.0024103 0.41677 -0.22768 0.34781 -0.00043378 0.0096187 -0.13943 0.011171 -0.12713 0.21852 -0.41048 0.14239 -0.019
601 0.59927 0.29721 -0.042536 0.087529 0.37312 0.34166 0.36037 -0.10691 -0.20925 -0.087749 -0.14268 -0.27608 0.31111 -0.070697 0.06301 -0.43687 0.40593 -0.21441 0.05856
3 -0.22466 -0.062918 0.037067 -0.35609 -0.56075 0.13398 0.096496 -0.10006 0.35449 -0.03536 -0.033816 -0.30797 0.14214 0.098602 0.1619 0.32192 0.12706 -0.48194 -0.50182
0.10132 0.047766 0.42162 0.20004 -0.56005 0.068582 0.46747 -0.36837 0.037067 0.45942 0.0017234 0.057124 0.47259 0.33818 -0.40593 0.13423 -0.053441 0.13601 0.13902 -0.29
518 -0.15649 -0.078606 0.042592 -0.54878 -0.29143 0.62177 -0.027966 0.26562 0.10644 0.017545 -0.42527 0.38708 -0.1703 -0.22822 -0.0084955 -0.35865 0.21155 0.15599 0.199
69 -0.094171 0.024968 0.64486 -0.15941 -0.043147 0.089127 0.12054 -0.057938 0.23782 -0.42374 0.23817 0.45178 -0.12524 0.086406 0.17115 0.29387 0.070648 0.30586 -0.11095
 -0.22974 -0.49355 -0.58589 0.71232 -0.10133 0.19213 0.17687 0.0031686 -0.17316 -0.12734 -0.028706 0.17209 0.069134 0.25824 -0.32798 0.04325 0.079583 0.16364 0.064095 -
0.22267 -0.065592 0.019898 0.1432 -0.2571 0.088413 0.56536 0.20177 -0.32081 -0.13101 -0.20561 -0.015505 -0.094919 0.007021 0.29238 -0.046284 -0.084866 0.1476 0.11312 0.
45539 -0.57964 0.10388 -0.015028 -0.23428 0.14486 -0.19285 -0.12356 -0.30339 0.023678 0.36839 0.49083 -0.19965 -0.0042465 0.032986 0.10592 0.25257 -0.059816 0.1287 -0.1
7263 0.011907 -0.21087 0.36213 -0.38927 -0.1074 0.2394 -0.021222 -0.12072 0.023633 -0.1955 -0.37147 -0.11626 0.22291 0.27766 0.17002 -0.12684 -0.17378 -0.064997 0.21861
 -0.19211 -0.049202 -0.52024 0.12951 0.29786 0.079944 0.029072 0.026806 -0.48504 0.28541 0.24734 -0.12411 -0.0099781 0.40337 -0.18107 -0.5301 0.25517 0.22416 0.18207 0.
035654 -0.18942 -0.16075 0.060329 -0.34788 -0.030947 -0.50364 -0.1447 0.15178 0.29832 -0.014766 0.025451 0.40409 0.021857 -0.051989 -0.15694 0.27402 0.19036 0.21303 0.3
0455 -0.1689 -0.46702 -0.11194 -0.32817 0.30811 0.037862 -0.23031 0.34559 -0.4066 0.19816 0.10223 -0.13005
```

Fig. 2: Word Vectors for one source word – derived word pair.

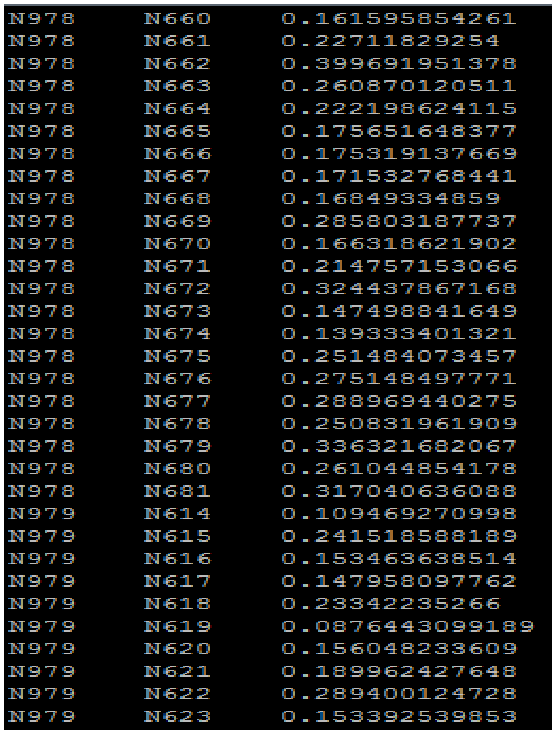
STEP 3: Representation of a single word pair as a single node.

This step requires representing the entire word pair as a single word vector. For this purpose, we have calculated the average of the word vector for source word and derived word.
We can also take the word vector for the source word or the derived word as the word vector for the entire word pair.

STEP 4: Calculation of edge weights between the different nodes.

For the calculation of edge weights between the different nodes, we calculate the similarity, specifically *Cosine Similarity* between each vector i.e. each word pair (node).
For our work, we have considered edge weights between all unlabelled nodes and edge weights between each labelled node to the set of nodes belonging to same affix class. i.e. there is a edge between labeled nodes of -*ally* and the corresponding nodes of class -*ally,* and so on.

With the calculation of edge weights between the nodes, the construction of input graph is complete. The input graph constructed consists of 982 nodes and 20460 edges.

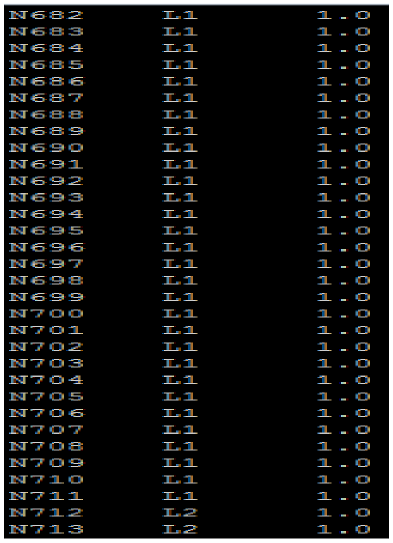


```
N978    N660    0.161595854261
N978    N661    0.22711829254
N978    N662    0.399691951378
N978    N663    0.260870120511
N978    N664    0.222198624115
N978    N665    0.175651648377
N978    N666    0.175319137669
N978    N667    0.171532768441
N978    N668    0.16849334859
N978    N669    0.285803187737
N978    N670    0.166318621902
N978    N671    0.214757153066
N978    N672    0.324437867168
N978    N673    0.147498841649
N978    N674    0.139333401321
N978    N675    0.251484073457
N978    N676    0.275148497771
N978    N677    0.288969440275
N978    N678    0.250831961909
N978    N679    0.336321682067
N978    N680    0.261044854178
N978    N681    0.317040636088
N979    N614    0.109469270998
N979    N615    0.241518588189
N979    N616    0.153463638514
N979    N617    0.147958097762
N979    N618    0.23342235266
N979    N619    0.0876443099189
N979    N620    0.156048233609
N979    N621    0.189962427648
N979    N622    0.289400124728
N979    N623    0.153392539853
```

Fig. 3: Nodes pairs with corresponding edge weights.

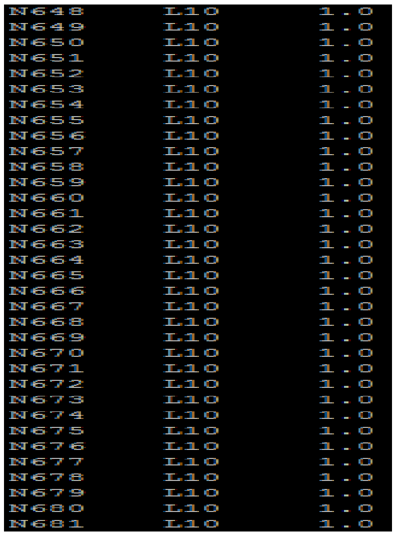
Label Inference:

For the label inference task, the MAD algorithm was implemented on our constructed input graph. This was done using *Junto* toolkit, which provides implementation of the MAD algorithm.
The input graph constructed was given input to the *Junto* as *input_graph.*
A seed file was created, *seeds,* which consists of the labeled nodes. A *gold_labels* file was also created which was used to compare the predicted labels of the unlabeled nodes with the actual labels.



```
N682      L1      1.0
N683      L1      1.0
N684      L1      1.0
N685      L1      1.0
N686      L1      1.0
N687      L1      1.0
N688      L1      1.0
N689      L1      1.0
N690      L1      1.0
N691      L1      1.0
N692      L1      1.0
N693      L1      1.0
N694      L1      1.0
N695      L1      1.0
N696      L1      1.0
N697      L1      1.0
N698      L1      1.0
N699      L1      1.0
N700      L1      1.0
N701      L1      1.0
N702      L1      1.0
N703      L1      1.0
N704      L1      1.0
N705      L1      1.0
N706      L1      1.0
N707      L1      1.0
N708      L1      1.0
N709      L1      1.0
N710      L1      1.0
N711      L1      1.0
N712      L2      1.0
N713      L2      1.0
```

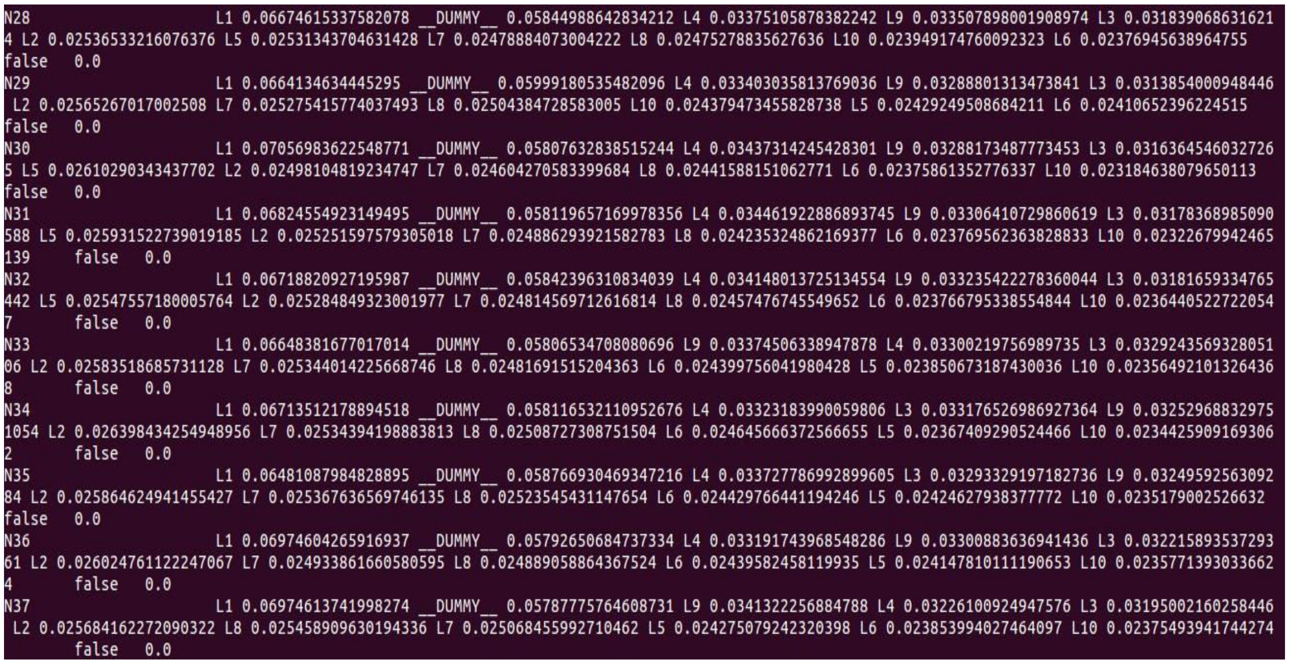
Fig. 4: Seeds for the input graph



```
N648      L10     1.0
N649      L10     1.0
N650      L10     1.0
N651      L10     1.0
N652      L10     1.0
N653      L10     1.0
N654      L10     1.0
N655      L10     1.0
N656      L10     1.0
N657      L10     1.0
N658      L10     1.0
N659      L10     1.0
N660      L10     1.0
N661      L10     1.0
N662      L10     1.0
N663      L10     1.0
N664      L10     1.0
N665      L10     1.0
N666      L10     1.0
N667      L10     1.0
N668      L10     1.0
N669      L10     1.0
N670      L10     1.0
N671      L10     1.0
N672      L10     1.0
N673      L10     1.0
N674      L10     1.0
N675      L10     1.0
N676      L10     1.0
N677      L10     1.0
N678      L10     1.0
N679      L10     1.0
N680      L10     1.0
N681      L10     1.0
```

Fig. 5: Gold labels for the unlabeled nodes.

Upon giving *input_graph, seeds* and *gold_labels* as input to *Junto,* MAD algorithm was implemented on the constructed graph and an output file, *label_prop_output* was generated. It shows the predicted label for each node with its corresponding score.



```
N28                     L1 0.06674615337582078 __DUMMY__ 0.05844988642834212 L4 0.03375105878382242 L9 0.033507898001908974 L3 0.031839068631621
4 L2 0.02536533216076376 L5 0.02531343704631428 L7 0.02478884073004222 L8 0.02475278835627636 L10 0.023949174760092323 L6 0.02376945638964755
false   0.0
N29                     L1 0.0664134634445295 __DUMMY__ 0.05999180535482096 L4 0.033403035813769036 L9 0.03288801313473841 L3 0.0313854000948446
 L2 0.02565267017002508 L7 0.025275415774037493 L8 0.02504384728583005 L10 0.024379473455828738 L5 0.02429249508684211 L6 0.02410652396224515
false   0.0
N30                     L1 0.07056983622548771 __DUMMY__ 0.05807632838515244 L4 0.03437314245428301 L9 0.03288173487773453 L3 0.0316364546032726
5 L5 0.02610290343437702 L2 0.02498104819234747 L7 0.024604270583399684 L8 0.02441588151062771 L6 0.02375861352776337 L10 0.023184638079650113
false   0.0
N31                     L1 0.06824554923149495 __DUMMY__ 0.05811965716997835 6 L4 0.034461922886893745 L9 0.03306410729860619 L3 0.03178368985090
588 L5 0.025931522739019185 L2 0.025251597579305018 L7 0.024886293921582783 L8 0.024235324862169377 L6 0.023769562363828833 L10 0.02322679942465
139     false   0.0
N32                     L1 0.06718820927195987 __DUMMY__ 0.05842396310834039 L4 0.034148013725134554 L9 0.033235422278360044 L3 0.03181659334765
442 L5 0.02547557180005764 L2 0.025284849323001977 L7 0.024814569712616814 L8 0.02457476745549652 L6 0.023766795338554844 L10 0.0236440522722054
7       false   0.0
N33                     L1 0.06648381677017014 __DUMMY__ 0.05806534708080696 L9 0.03374506338947878 L4 0.03300219756989735 L3 0.0329243569328051
06 L2 0.02583518685731128 L7 0.025344014225668746 L8 0.02481691515204363 L6 0.024399756041980428 L5 0.023850673187430036 L10 0.02356492101326436
8       false   0.0
N34                     L1 0.06713512178894518 __DUMMY__ 0.058116532110952676 L4 0.03323183990059806 L3 0.033176526986927364 L9 0.03252968832975
1054 L2 0.026398434254948956 L7 0.02534394198883813 L8 0.02508727308751504 L6 0.024645666372566655 L5 0.02367409290524466 L10 0.0234425909169306
2       false   0.0
N35                     L1 0.06481087984828895 __DUMMY__ 0.058766930469347216 L4 0.033727786992899605 L3 0.03293329197182736 L9 0.03249592563092
84 L2 0.025864624941455427 L7 0.025367636569746135 L8 0.02523545431147654 L6 0.024429766441194246 L5 0.02424627938377772 L10 0.0235179002526632
false   0.0
N36                     L1 0.06974604265916937 __DUMMY__ 0.05792650684737334 L4 0.033191743968548286 L9 0.03300883636941436 L3 0.032215893537293
61 L2 0.026024761122247067 L7 0.024933861660580595 L8 0.024889058864367524 L6 0.02439582458119935 L5 0.024147810111190653 L10 0.0235771393033662
4       false   0.0
N37                     L1 0.06974613741998274 __DUMMY__ 0.05787775764608731 L9 0.0341322256884788 L4 0.03226100924947576 L3 0.03195002160258446
 L2 0.025684162272090322 L8 0.025458909630194336 L7 0.025068455992710462 L5 0.024275079242320398 L6 0.023853994027464097 L10 0.02375493941744274
        false   0.0
```

Fig 6: Output file, *label_prop_output*

RESULTS

Among the 682 predictions for the unlabeled nodes, we calculate the False Negatives (wrong predictions) and True Positives (correct predictions). Based on that the accuracy was calculated as:

True Positives = 592
False Negatives = 90

Hence, Prediction Accuracy = (True Positives) / (True Positives + False Negatives)
= 86.8035190616%

CONCLUSION

In our work, Modified Adsorption was implemented on a set of 982 source word – derived word pairs, and the accuracy was checked based on the no. of correct predictions. The labeled nodes (seeds) in the input graph were 300 (30% of the total no. of nodes). The trained model was 86.8% accurate, which is satisfactory.

A lot of future improvements can be done. The percentage of seed labels being input to the graph can be varied and the accuracy can be compared. The variation of accuracy can also be seen by varying th hyperparameters of the MAD implementation in *Junto.* A detailed comparison with respect to the existing label propagation algorithms can be made, and statistical measures can be derived to assess the better among the two algorithms.