Volume 4, Issue 3, March 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

A Teacher-Guided Automated Essay Grading Tool using Gradient Descent Algorithm and Reflective Random Indexing

Reinald Kim T. Amplayo
Computer Studies Cluster
Ateneo de Davao University

Sean Carlo C. Bermejo
Computer Studies Cluster
Ateneo de Davao University

Michael John N. Pedros
Computer Studies Cluster
Ateneo de Davao University

Abstract— The emergence of automated essay scoring (AES) tools is an effect of the problems of teachers in tediously checking essays of students. The AES tools proved to be useful for this task. However, current tools disables the ability of the teacher to select the criteria to be used in checking these essays. Thus, these tools are discouraged by teachers. This study made use of the gradient descent algorithm for multiple variables to automate the checking of the essay responses with teacher-guided scoring preferences. The extracted features are the word count, the keyword count, the spelling error value, the grammar error value and the content similarity value. Reflective random indexing is used to extract the content similarity value of a student essay response to the essay response provided by the teacher. The model shows an 89.78% accuracy based on the quadratic weighted kappa error (QWKE) metric. This shows that the proposed model is able to correctly grade student essays. The model was then implemented in a batch-processed essay quiz creation and scoring web application for full usage on any educational institution.

Keywords— gradient descent, reflective random indexing, automated essay scoring, teacher-guided scoring

I. INTRODUCTION

Teachers especially in the social sciences, humanities and letters department usually include essay questions in their exams to further measure the current viewpoint of the student regarding a problem, a scenario, or a specific topic. Although essays are one of the most effective ways in measuring students' capabilities, they usually become the main reason for teachers returning the test papers to the students. There are usually a lot of students in these subjects as universities require these subjects to be taken up by all the students regardless of their majors. Checking these essay questions is a very tedious task that consumes a lot of time of the teachers. This would result to students who lack information regarding their current class standing.

The use of specialized programs to assign grades to a school-based and student-written essay or the Automated Essay Scoring (AES), is a widely known problem in the field of educational assessment and natural language processing. Various researches on AES are already available with a lot of success and with high accuracy. However, most teachers still believe that these tools are inefficient as the current approaches on AES are insensitive to the uniqueness of every teacher. They are unsure whether these tools use the same criteria as they are using for grading essays. For example, the AES tool might use the grammar and the spelling as criteria, but the teacher doesn't mind if the words are incorrectly spelled.

A teacher-guided scoring preferences is a feature that enables teachers to control the AES tool by providing their own data and choosing the features to be used in the model. In this study, an automated essay grading tool with teacher-guided scoring preference is implemented using the gradient descent algorithm with multiple variables and reflective random indexing. Other techniques are used for pre-processing the data and extracting other features.

II. AUTOMATED ESSAY SCORING

Automated Essay Scoring is the computer technology that evaluates and scores the written prose [1]. These systems are developed to assist teachers in low-stakes classroom assessment as well as testing companies and states in large-scale high-stakes assessment. These are mainly used to help save time, cost, reliability, and generalizability issues in assessing a written prose. The accuracy and reliability of AES with respect to writing assessment has been studied by a number of research papers. The results concluded that these systems and human graders have high agreement rates.

Although AES systems are producing good results on their accuracy and effectiveness, they have been criticized for their lack in human interaction and their vulnerability to cheating [2]. These shortcomings are very crucial in the educational setting since these would defeat the purpose of AES systems. The lack in human interaction limits the students' understanding on the subject area. Being vulnerable to cheating makes automation of grading essays useless. Teachers also want to reflect their grading styles on the criteria being used by their own AES system. This is a main problem that needs to be answered since current systems assume that all teachers have the same criteria in checking essays.

To answer these problems, future systems should allow student-teacher interactions by implementing a web-based application that is used in a classroom setting. This will also lessen the vulnerability to cheating of students since they are

enforced to take the quiz inside a classroom with computers not connected to the Internet. Lastly, future systems should also include a teacher-guided scoring preferences feature where the teacher has a full control on what criteria to use.

A. Current Algorithms for AES

Most automated essay scoring programs and applications use a linear regression model from several essay features like word count and keyword count [3]. The extracted features will be used to train the linear regression model. The trained model is then used to predict the essay score of the essay responses. A research of Murray and Orii [4] made use of linear regression models to predict essay scores. They used generalized linear models that use two types of features: dense and sparse features. Dense features include character count, word count, misspelled word count, stop word count, symbol count, etc. Sparse features include part-of-speech unigram, bigram, and trigram. The research shows that some features have low impact to the score prediction. Some high impact features are the word count, the keyword count, the misspelled word count and some n-grams.

The k-Nearest Neighbor (kNN) algorithm for text categorization is applied in the automation of the scoring of the essays in the College English Test CET-4, a national English level test in the People's Republic of China [5]. After removing the stop words, all the possible words and phrases were used as features. Thus, each essay is represented by a word space model. The term frequency and inverse document frequency (TF-IDF) weight. The cosine similarity algorithm is used to check whether two word space models are similar. The k-NN showed an accuracy of 76%.

A Backpropagation Neural Network and Latent Semantic Analysis (LSA) were used to assess the quality of Thai essays written by high school students [6]. Forty essays were evaluated by high school teachers and assigned a score. They made two experiments, one with LSA and one without LSA. The experiment without LSA used raw term frequency vectors of the essays and their corresponding annotated scores to train the neural network. The experiment with LSA used Latent Semantic Analysis before feeding the vectors to the neural network. The results show that the addition of a semantic analysis technique such as LSA improves the scoring performance.

III. GRADIENT DESCENT ALGORITHM

Gradient descent algorithm is a first-order optimization algorithm that finds a local minimum of a function by taking steps proportional to the negative of the gradient of the function at the current point. It starts at some value, use the derivative of the function at that point. The derivative tells the algorithm which way to move. This process is repeated until the derivative of the function at the current point is approximately zero. When this happens, an approximate value is outputted.

In this study, gradient descent algorithm is used for linear regression for multiple variables. The linear regression problem states that given a list of feature vector, output an estimated value by equating a linear function. For example, if *x* represents your feature vector and *y* represents your estimated value, then this represents a linear regression problem:

$$h(x) = y = \sum_{i=0}^{n} \theta_i x_i \quad (1)$$

where θ is the parameter vector that controls the intensity or the impact of each feature in the feature vector and *n* is the number of features in the feature vector.

To get the right values for the parameter vector, a cost function is introduced. The cost function is the function that is minimized. The lower the cost function, the better values are gotten for the parameter vector. This further implies that the function (1) would fit properly to the training data. The cost function is as follows:

$$J(\theta) = \frac{1}{2}\sum_{i=0}^{m}(\sum_{j=0}^{n} \theta_j x_{i,j} - y_i)^2 \quad (2)$$

where *m* is the number of training data. (2) is the cost function to be minimized and this minimization problem is the main problem of the linear regression problem.

To answer this problem, gradient descent algorithm can be used. The algorithm goes as follows:

- Start with initial guesses of the values of the parameter vector.
- Change the parameters by looking at the gradient of the function at the current point. This will concurrently reduce the cost function.
- Repeat the first two steps until there is a convergence to a local minimum.

After acquiring the local best parameter vector, the function (1) can now be used to predict future data. The prediction would need the mean and the standard deviation of the training data, and the feature vector of the test input. The learned output is represented as:

$$y' = \left[1, \frac{x' - \mu}{\sigma}\right] * \theta \quad (3)$$

where *y'* is the learned output, *x'* is the feature vector of the test input, μ is the mean of the feature vectors in the training data, and σ is the standard deviation of the feature vectors in the training data. Notice that 1 is inserted in the first vector to indicate that the first value in the parameter vector is a bias value.

IV. REFLECTIVE RANDOM INDEXING

In information retrieval, Latent Semantic Analysis (LSA) is the widely used technique for indirect inference on semantic relatedness between terms that don't co-occur directly in one document. It proceeds in projecting the terms into a high-dimensional semantic space with the use of term-document matrix. The dimension of this space is reduced using Singular Value Decomposition (SVD), a widely known dimension reduction technique. The distance between terms are then calculated using functions like cosine similarity or normalized scalar product. The distance serves as the similarity between two terms.

However, LSA relies heavily on SVD, a computationally demanding technique in reducing the dimensions of the semantic space. This renders LSA not usable in a very large corpus within a personal computer with limited RAM. Random Indexing (RI), an alternative to LSA, answers this problem by not using the SVD at all [7]. The technique

proceeds in randomly allocating values on zero document vectors. This will then be used for term document corpus training by calculating the weight of the product of the number of times the term occurs in a document and the document vector of that specific document. Weights are calculated using the log-entropy function which is defined as:

$$logentropy = \sum \frac{P_{i,j} \log_2 P_{i,j}}{\log_2 n} \quad (4)$$

$$P_{i,j} = \frac{tf_{i,j}}{gf_{i,j}} \quad (5)$$

where $tf_{i,j}$ is the local term frequency of term i and $gf_{i,j}$ is the global term frequency of term i and n is the number of documents in the corpus. Note that even if the document vectors were given randomly allocated values, these document vectors are known to be orthogonal to each other.

The computational complexity of LSA and RI are cubic and quadratic, respectively [8]. But the results of the RI are not optimal for the derivations of meaningful indirect connections. It can also be observed that term vectors can be cyclically retrained in RI, and the related models increase the inference ability of a variant of RI. Thus, an iterative variant Reflective Random Indexing (RRI) is introduced. This method generates new inferences by also considering what it has learned from a dataset in a previous iteration. For example, a Term-based RRI proceeds as follows:

1. Randomly allocate values on zero term vectors.
2. Train a new document vector by adding together the term vectors for each term it contains.
3. Train a new term vector by adding the document vector of each document it occurs in.
4. Repeat steps 2 and 3 if necessary.

V. DESIGN AND METHODOLOGY

A. Data Gathering

In 2012, The Hewlett Foundation uploaded essay datasets. The essays range from an average length of 550 words per essay response. There are four essay dataset that are dependent to a source information. In this study, these source dependent essay responses are used as datasets. All responses were written by students from Grade 7 and Grade 10. Each of the essay sets has its own unique characteristics to test the built AES tool's capabilities.

The Hewlett Foundation provided the essay datasets, the essay descriptions, and the training materials. The essay descriptions provide the source information and the essay question where the students will base their essay responses. The training materials provide the criteria used by the teachers on different essay datasets. The table below shows some statistics and sample data that were provided:

TABLE I
DATASET STATISTICS AND SAMPLE DATA

Dataset	#Test	#Training	Sample Essay	Score
1	431	1295	The story is basically talking about more people with cars should ride bicycles to work or to school's so it want be alot of polution in the air.	0/3
2	442	1329	She said that she was going to take the test agen because to show her mom that she can do it and Past it this time. instad of worring over her old home.	1/3
3	451	1354	"The mood set by the author is thankful. The author says that he is happy to have his parents. He thanks them for giving him a life in Cuba, but still carrying out their Cuban ways. He was happy to have others come into his apartment and sit down and act like family. He likes that his parents are selfless and …"	3/4
4	450	1350	"The builders faced many obstacles while trying to add a mooring mast on top of the Empire State Building. As stated in paragraph fourteen, nature was the biggest obstacle the builders faced. The ""violent air currents"" would cause dirigibles to move around the mooring mast, even if it was tethered. …"	4/4

The Wikipedia Extended Abstracts (WEA) corpus must also be gathered as this will be the basis of the language model. The language model is used by the reflective random indexing technique in finding similarities between two documents. The WEA corpus is uploaded in the DBpedia website[1]. The corpus is a single text file that consists one extended abstracts per line. The first five lines of the corpus is shown below:

TABLE III
FIRST FEW LINES OF WEA CORPUS

<http://dbpedia.org/resource/Autism> <http://dbpedia.org/ontology/abstract> "Autism is a disorder of neural development characterized by impaired social interaction and communication, and by restricted and repetitive behavior. The diagnostic criteria require that symptoms become apparent before a child is three years old. …"
<http://dbpedia.org/resource/Achilles> <http://dbpedia.org/ontology/abstract> "In Greek mythology, Achilles was a Greek hero of the Trojan War and the central character and greatest warrior of Homer's Iliad. Achilles was a demigod; his mother was the nymph Thetis, and his father, Peleus, was the king of the Myrmidons. …"
<http://dbpedia.org/resource/An_American_in_Paris> <http://dbpedia.org/ontology/abstract> "An American in Paris is a symphonic tone poem by the American composer George Gershwin, written in 1928. Inspired by the time Gershwin had spent in Paris, it evokes the sights and energy of the French capital in the 1920s. …"

[1] http://downloads.dbpedia.org/3.9/en/long_abstracts_en.nq.bz2

<http://dbpedia.org/resource/Aristotle> <http://dbpedia.org/ontology/abstract> "Aristotle (384 BC\u00A0\u2013 322 BC) was a Greek philosopher and polymath, a student of Plato and teacher of Alexander the Great. His writings cover many subjects, including physics, metaphysics, poetry, theater, music, logic, rhetoric, linguistics, politics, government, ethics, biology, and zoology. …"
<http://dbpedia.org/resource/Academy_Award_for_Best_Production_Design> <http://dbpedia.org/ontology/abstract> "The Academy Awards are the oldest awards ceremony for achievements in motion pictures. The Academy Award for Best Production Design recognizes achievement in art direction on a film. …"

B. Language Model Building

Before it can be used to build the language model, the WEA corpus needs to be pre-processed. First, strings that are not informative such as HTML and XML tags, Unicode symbols, etc. are removed. Next, the one-file corpus is divided into several text files, one extended abstract per text file. Since there are about 3.6 million extended abstracts gathered, the output is a directory with 3.6 million text files. Lastly, file indexing is used to provide faster and optimized searching for words in the corpus. This method will produce a directory of the indices of the corpus. The language model is then built using the index folder.

The model is built using Reflective Random Indexing (RRI). Terms with frequency of less than 10 are removed. Stop words are also removed. Numbers are also filtered. The model is built with 800 dimensions and two training cycles. Since the RAM of the workstation is relatively small, the model is trained incrementally, sacrificing time efficiency to accommodate the small RAM. It will produce an object file that represents the language model.

C. Linear Regression Model Training

To train the linear regression model using the gradient descent algorithm with multiple variables, the features of the dataset need to be extracted. Five features are extracted: spelling error value, grammar error value, word count, keyword count, and the content similarity value.

The first two features, the spelling error and the grammar error values, are calculated using spelling and grammar collection tools. The spelling error value is the number of spelling corrections of the essay response. The grammar error value is the number of grammar corrections of the essay response. The word count is simply the number of words in the essay response. The keyword count is the number of keywords that are found in the essay response. Keywords are found in the training materials provided along with the dataset. The content similarity value is the cosine similarity between two document vectors: the document vector of the student's essay response and the document vector of the teacher's essay response. The cosine similarity is calculated as follows:

$$cossim = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i x B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} x \sqrt{\sum_{i=1}^{n} (B_i)^2}} \quad (6)$$

where A and B are the document vectors of the student's essay response and the teacher's essay response, respectively, and n is the number of terms in the document vector. The results of the cosine similarity will be in the range [-1, 1] where -1 indicates that the responses are dissimilar and 1 indicates that the responses are similar.

The training materials provided the rubrics and criteria on how the teacher graded the essay responses. The table below show the concise training guideline for each essay dataset:

TABLE III
TRAINING GUIDELINES

Dataset	Guideline	Feature Used
1 2	• Understanding of the complexities of the text and question • Insightful observations are interwoven with textual quotes and meticulous details • Main answer is supported with expressed information • Ideas are extended through evaluation of the situation and appropriate comparison	• word count • keyword count • content similarity value
3 4	• Idea/Topic Development • Organization • Details • Language/Style • Structure • Grammar and Usage • Mechanics	• spelling error value • grammar error value • word count • keyword count • content similarity value

After extracting the features, the mean and the standard deviation of all the extracted features are calculated. Then, the gradient descent algorithm is ran and the parameter vector is acquired. The linear regression model is now trained and ready for scoring new essay responses.

D. Tool Implementation

The proposed tool is a batch-processed web application that enables a teacher to create quizzes with one or more essays and enables students to answer those quizzes. When adding questions, the teacher should provide the source information, the question, the best answer, the keywords, the number of items (high score), and the training data. The teacher should also provide the criteria by ticking on the checkboxes with the corresponding feature names: random

projection (content similarity value), word count, keyword count, spelling error value, and/or grammar error value. The linear regression model is trained using the information given by the teacher. These questions are then delivered to the students. When the students are done answering the essay questions, the teacher would click a button to check the essays by batch. The teacher waits for the procedure to stop. He/she then re-evaluates the essay if necessary. After re-evaluation, the teacher sends the essay scores to the students. The tool is implemented using Jython, a combination of Java and Python, with Django web framework. The figures below presents the screenshots of the tool.

Fig. 1 Teacher Adding a Question

Fig. 2 Student Taking a Quiz

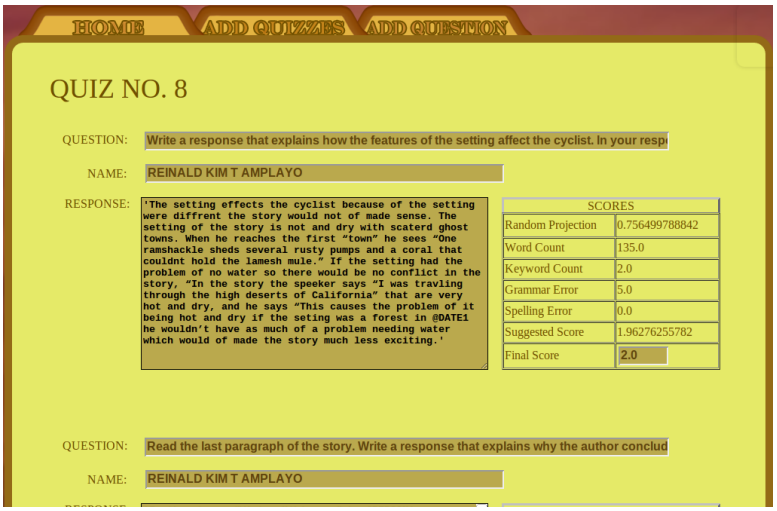

Fig. 3 Teacher Checking the Essays

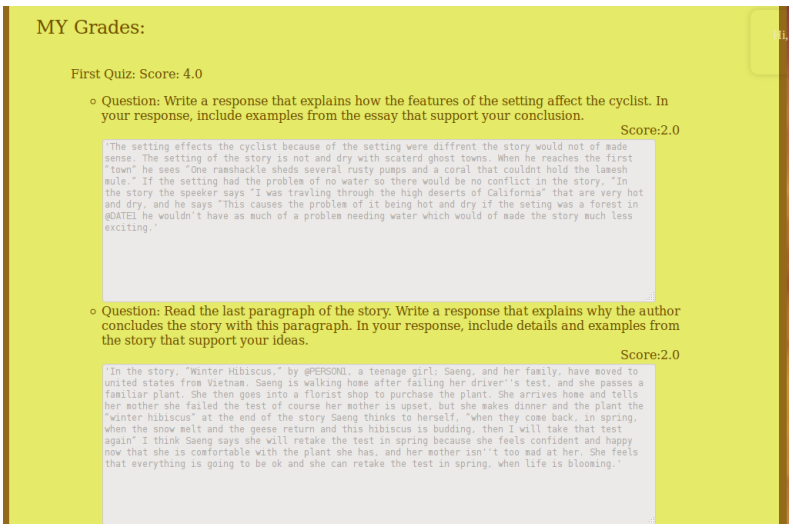

Fig. 4 Student Viewing the Grades

VI. RESULTS

The performance of the automated essay grading tool is evaluated with the quadratic weight kappa error (QWKE) metric, which measures the agreement between two raters, the machine (learned) and the teacher (trained). The metric typically varies from 0 to 1 and my go below 0 for events that there are lesser agreements between the raters. The QWKE metric goes as follows:

1. An N-by-N histogram matrix is constructed over the essay ratings, such that a cell on column *c* and row *r* corresponds to the number of essays that received a rating *c* by the machine rater, and a rating *r* by the teacher.
2. A matrix of weights is calculated based on the difference between raters' scores. The weight for a particular cell is calculated as follows:

$$weight = 1 - \frac{distance^{\,2}}{maxdistance^{\,2}} \tag{7}$$

3. For each cell, the sum of the product of the weight matrix and the score matrix ($P_{observed}$) is calculated.
4. The QWKE metric is calculated as follows:

$$QWKE = \frac{P_{observed} - P_{expected}}{1 - P_{expected}} \tag{8}$$

Table V shows the N-by-N histogram matrix of the predicted vs. the actual scores. The predicted scores are plotted horizontally and the actual scores are plotted vertically.

TABLE V
N-BY-N HISTOGRAM MATRIX (PREDICTED VS. ACTUAL)

	0	**1**	**2**	**3**	**4**
0	26	61	6	1	0
1	15	350	132	12	0
2	0	97	433	71	1
3	0	7	182	256	10
4	0	0	5	79	29

The frequencies of agreement shows the maximum possible agreement between two scores, the expected agreement between two scores, and the observed agreement between two scores. If for every rating, the observed agreement is greater than the expected agreement, then the QWKE is sure to be positive. The tables below show the frequencies of agreement and the QWKE evaluation.

TABLE VI
FREQUENCIES OF AGREEMENT

Rate	Maximum Possible	Chance Expected	Observed
0	41	2.17	26
1	509	147.85	350
2	602	257.37	433
3	419	107.53	256
4	40	2.55	29
Total	**1611**	**517.47**	**1094**

TABLE VII
KAPPA WITH QUADRATIC WEIGHTING

Observed Kappa	Standard Error	.95 Confidence Interval	
		Lower Limit	Upper Limit
0.7441	0.0302	0.685	0.8032
0.8288	**Maximum possible quadratic-weighted kappa**		
0.8978	**Observed as proportion of maximum possible**		

The QWKE metric gave the tool an observed kappa of 74.41% with 3.02% standard error. This can therefore went up to 80.32%. The maximum possible QWKE is 82.88%. Thus the ratio of the QWKE metric over the maximum possible QWKE gives us the accuracy of the implemented tool, which is 89.78%.

VII. CONCLUSION

The study presented a gradient descent algorithm approach to the automated essay scoring problem. With the use of reflective random indexing technique in semantic analysis, a content similarity value was extracted to the essay responses. Together with the spelling error value, grammar error value, word count, and keyword count, the content similarity value was used as input vectors for the gradient descent algorithm for multiple variables. The tool is rated using the quadratic weighted kappa error and has an accuracy of 89.78%. Lastly, the weaknesses of previous AES systems are answered. Cheating can no longer happen since there are no Internet resources available in a classroom setting. Student-teacher interaction can already be done when the teacher gives a comment about the response given by the student. The concept of a teacher-guided scoring preferences is implemented.

ACKNOWLEDGEMENT

This study would not have been possible without the support of many people. The authors wishes to express their gratitude to their adviser, Mr. Dominic Gerald Cimafranca, to the creators of the SemanticVectors package, especially Dr. Dominic Widdows, Ph.D, to their panelists, Ms. Pelobello, Ms. Banawan, and Mr. Freires, to their coordinator Ms. Quindoy, and to their loving friends and families.

REFERENCES

[1] S. Dikli, "Automated essay scoring," *Turkish Online Journal of Distance Education,* vol. 7, pp. 49-62, Jan. 2006
[2] B. Code, C. Vojak, S. Kline, S. McCarthey, and M. Kalantzis, "New spaces and old places: An analysis of writing assessment software," *Computers and Composotion*, vol. 28, pp. 97-111, Jun. 2011.
[3] S. J. Haberman and S. Sinharay, "The application of the cumulative logistic regression model to automated essay scoring," *Journal of Educational and Behavioral Statistics,* vol. 35, pp. 586-602, Oct. 2010.
[4] K. W. Murray and N. Orii (2012) Automatic Essay Scoring. [Online]. Available: http://www.cs.cmu.edu/afs/cs.cmu.edu/Web/People/norii/pub/aes.pdf
[5] B. Li, J. Lu, J. M. Yao, and Q. M. Zhu, "Automated essay scoring using the KNN algorithm," in *Proc. International Conference on Computer Science and Software Engineering*, 2008, pp. 735-738.
[6] C. Loraksa and R. Peachavanish, "Automatic Thai-language essay scoring using neural network and latent semantic analsysis," in *Proc. First Asia International Conference on Modelling & Simulation,* 2007, pp. 400-402.
[7] T. Cohen, R. Schvaneveldt, and D. Widdows, "Reflective random indexing and indirect inference: A scalable method for discovery of implicit connections," *Journal of Biomedical Informatics,* vol. 43, pp. 240-256, Sep. 2009.
[8] D. Widdows and T. Cohen, "The semantic vectors package: New algorithms and public tools for distributional semantics," in *Proc. Fourth IEEE International Conference on Semantic Computing,* 2010, pp. 9-15

