Vocab Learn: A Text Mining System to Assist Vocabulary Learning

Jingwen Wang, Changfeng Yu, Wenjing Yang and Jie Wang
Department of Computer Science, University of Massachusetts, Lowell, MA, U.S.A.

Keywords: TFIDF, TextRank, RAKE, Word2Vec, Minimum Edit Distance.

Abstract: We present a text mining system called Vocab Learn to assist users to learn new words with respect to a knowledge base, where a knowledge base is a collection of written materials. Vocab Learn extracts words, excluding stop words, from a knowledge base and recommends new words to a user according to their importance and frequency. To enforce learning and assess how well a word is learned, Vocab Learn generates, for each word recommended, a number of semantically close words using word embeddings (Mikolov et al., 2013a), and a number of words with look-alike spellings/strokes but with different meanings using Minimum Edit Distance (Levenshtein, 1966). Moreover, to help learn how to use a new word, Vocab Learn links each word to its dictionary definitions and provides sample sentences extracted from the knowledge base that includes the word. We carry out experiments to compare word-ranking algorithms of TFIDF (Salton and McGill, 1986), TextRank (Mihalcea and Tarau, 2004), and RAKE (Rose et al., 2010) over the dataset of Inspec abstracts in Computer Science and Information Technology Journals with a set of keywords labeled by human editors. We show that TextRank would be the best choice for ranking words for this dataset. We also show that Vocab Learn generates reasonable words with similar meanings and words with similar spellings but with different meanings.

1 INTRODUCTION

Learning a natural language begins with learning words. To learn a new word involves mastering the following three vocab skills with respect to an underlying knowledge base: (1) recognize its correct form; (2) understand its meanings; (3) use it appropriately in a sentence (Nurmukhamedov and Plonsky, 2018). A knowledge base is a collection of written materials with respect to the same topics. For example, a high-school student taking an English literature course will have the list of reading materials assigned to the student as a knowledge base for the course. A medical-school student taking a human anatomy course will have the lecture notes, textbooks, and lab materials as the knowledge base for the course. Each knowledge base contains a new vocabulary to be learned. Indeed, learning the underlying vocabularies is fundamental in any field of study and in any standardized test such as SAT Reading (available at https://collegereadiness.collegeboard.org/sat/inside-the-test/reading) and GRE Vocabulary (available at https://www.ets.org/gre). The importance and methods of learning vocabularies have been studied widely (see, e.g., (Dickinson et al., 2019; Godfroid et al., 2018; Harmon, 2002; Wilkerson et al., 2005; Chen and Chung, 2008; Al-Rahmi et al., 2018; Xie et al., 2019)).

Using computing technology, including text mining, natural language processing, and mobile apps, it is possible to develop a vocabulary-learning tool to help people of different academic backgrounds and abilities to learn new words more efficiently and more effectively.

We devise such a text mining system called Vocab Learn, which extracts and recommends to the user a list of new words according to their importance and frequency in the underlying knowledge base, excluding words already learned by the user. In other words, a more important and more frequent unlearned word will be recommended to the user with a higher probability.

To enhance vocab skills 1 and 2, for each word recommended, Vocab Learn provides a number of semantically close words using Word2Vec embeddings (Mikolov et al., 2013a), and a number of look-alike words that have almost the same spellings/strokes but with different meanings using Minimum Edit Distance (Levenshtein, 1966). Clearly, semantically close words include standard synonyms. Vocab Learn

Wang, J., Yu, C., Yang, W. and Wang, J.
Vocab Learn: A Text Mining System to Assist Vocabulary Learning.
DOI: 10.5220/0008319301550162
In *Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2019)*, pages 155-162
ISBN: 978-989-758-382-7

also links each word to its definitions with an online dictionary and its parts of speech. To enhance vocab skill 3, Vocab Learn provides sample sentences with high salience scores that include the word, all extracted from the underlying knowledge base.

To help assess the user's understanding of words and determined which words have been learned by the user, Vocab Learn selects words according to their importance and frequency in the underlying knowledge base, and displays to the user the word itself and a definition of the word, together with a few semantically close words and look-alike words. The user is to select the correct word based on the given definition. Likewise, Vocab Learn also displays a word together with the definitions of a few semantically close words and look-alike words. The user is to select the correct definition of the given word. Words that are tested well will be marked learned.

This paper is focused on the essential text-mining features of Vocab Learn, excluding straightforward features such as linking a word to its dictionary definitions. In particular, for each word in the knowledge base, we are interested in getting its correct classifications, and generating its semantically close words and look-alike words. The latter two types of words may or may not appear in the knowledge base. For word classifications, we investigate the following common word ranking algorithms: TFIDF (Salton and McGill, 1986), TextRank (Mihalcea and Tarau, 2004), and RAKE (Rose et al., 2010), and use them to classify words according to their importance.

We would like to compare the accuracy of these word-ranking algorithms. To do so we need a collection of documents of a reasonable size with extracted words that are numerically ranked by experts on importance. Lacking such a dataset has hindered such comparisons. Instead, we follow the standard practice on studying keyword extractions (Hulth, 2003; Mihalcea and Tarau, 2004; Rose et al., 2010) using Inspec abstracts for comparing accuracy of the binary classifications of important words. We carry out substantial experiments and show that, among linear combinations of TFIDF, TextRank, and RAKE, TextRank alone provides the best $F_{0.5}$-score on this dataset. The reason that we use the $F_{0.5}$ score rather than the F_1 score to measure accuracy is that the labeled dataset contains words that are not extracted from the underlying knowledge base, and so it is impossible to obtain a full recall no matter what algorithms are used. In this situation precision is more important and so should carry a larger weight.

We further show that Vocab Learn generates reasonable words with similar meanings and words with similar spellings but with different meanings.

The rest of the paper is organized as follows: We describe in Section 2 some notations for the paper and in Section 3 related work on ranking words, discovering semantically close words, and finding look-alike words. We present Vocab Learn in Section 4 and carry out experiments In Section 5. We conclude the paper in Section 6.

2 NOTATIONS

Throughout this paper, we will convert each verb in a document to its stem form and each noun to its singular form. Let n and m denote, respectively, the number of documents in a given knowledge base and the number of words (excluding stop words) from it. Denote the knowledge base by $\mathcal{D} = \{D_1, D_2, \ldots, D_n\}$, where D_i is a text document, and the vocabulary extracted from it by $V = \{w_1, w_2, \ldots, w_m\}$, which is to be learned by the user.

3 RELATED WORK

Prior arts on how to devise an effective vocabulary learning system have been studied intensively and extensively for several decades (Teodorescu, 2015; Chen et al., 2019a; Chen et al., 2019b; Murphy et al., 2013; Park, 2011; Peters, 2007). An effective vocabulary learning system should be capable of 1) ranking words based on their features such as alphabetical order, frequency, and salience; and 2) finding both semantically close words and look-alike words to help assess the user's understanding of words.

3.1 Ranking Words

TFIDF (Salton and McGill, 1986) is a widely used measure to identify significant words in a document over other documents. A word is considered significant in a document if it appears frequently in the document, but appears rarely in the other documents. For each word w in a document D, its TFIDF value is the product of its term frequency (TF) in D, denoted by let $\mathrm{TF}(w,D)$ denote the term frequency (TF) of w in D and $\mathrm{IDF}(w,\mathcal{D})$ the inverse document frequency (IDF) with respect to $\mathcal{D}$, where $\mathrm{IDF}(w,\mathcal{D}) = \log n - \log|\{D \in \mathcal{D} : w \in D\}|$. Then the TFIDF value of w in D is computed by $\mathrm{TFIDF}_D(w) = \mathrm{TF}(w,D) \cdot \mathrm{IDF}(w,\mathcal{D})$.

TextRank (Mihalcea and Tarau, 2004), on a given document, first removes stop words from a given document, and then constructs a weighted word co-occurrence graph such that nodes represent words and

two nodes are connected exactly when the two words they represent co-occur in the document. The weight of an edge is the number of times of co-occurrences of its adjacent nodes. It uses the PageRank algorithm (Page et al., 1999) to compute a ranking of each word. It was noted in (Mihalcea and Tarau, 2004) that TextRank achieves the best performance when only nouns and adjectives are used to construct a word graph. For our purpose, we need to rank all words, except predefined stop words.

RAKE (Rose et al., 2010), on a given document D, first removes stop words and then generates word sequences using a set of word delimiters. It constructs a weighted word co-occurrence graph, where each node represents a word, the weight of the self loop of each word is the frequency of the word, and two different words are connected if they appear in the same word sequence with its weight equal to the number of different sequences they both appear in. For each word w in the graph, compute its score by $deg(w)/freq(w)$, where $deg(w)$ is the degree of w and $freq(w) = \text{TF}(w,D)$. We note that under RAKE, the score of each word is in $[1,\infty)$, while under TFIDF and TextRank, the corresponding score of each word is in $[0,\infty)$.

3.2 Finding Semantically Close Words

Given a word $w \in V$, a word embedding method such as Word2Vec (Mikolov et al., 2013a; Mikolov et al., 2013b) and GloVe (Pennington et al., 2014) can be used to find semantically close words of w, which may or may not be included in V. Word embedding, trained on a large corpus of documents, represents words with vectors of high dimensions, where semantically close words have similar vectors.

3.3 Finding Look-alike Words

Given a word $w \in V$, the minimum word edit distance can be used to find look-alike words from a dictionary, which may or may not be included in W. Word edit distance is first studied by Levenshtein (Levenshtein, 1966) in his effort to study Minimum Levenshtein distance, also known as minimum edit distance. Edit distance counts the number of editing operations to transform a word to another word, including insertion, deletion, and substitution.

4 VOCAB LEARN

Vocab Learn consists of an online pipeline and an offline pipeline (see Figure 1). Given a knowledge base $\mathcal{D}$, the offline pipeline filters out stop words and extracts words from each document in $\mathcal{D}$ to form V. It then classifies words based on their rankings into three categories of "Very Important", "Important", and "Not so Important", and based on their frequencies into three categories of "Very Frequent", "Frequent", and "Not so Frequent". For each word $w \in V$, Vocab Learn computes its semantically close words and look-alike words using $\mathcal{D}$ and out-of-band data, such as some other knowledge bases and online dictionaries. The data generated by the offline pipeline is stored in a database for later use. The online pipeline is a mobile application including a learning module and a testing module, interacting with the user and the database.

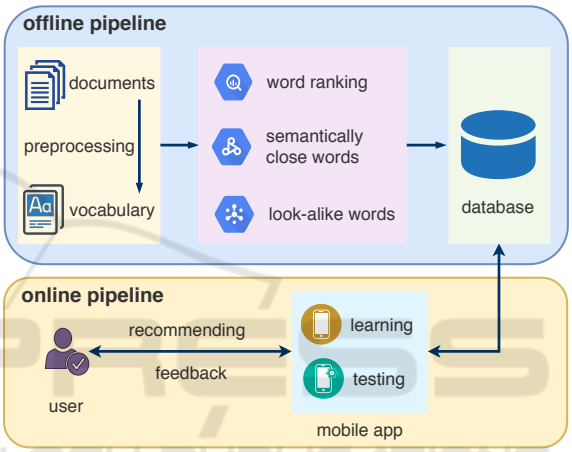

Figure 1: Vocab Learn architecture.

4.1 Stop Words and Preprocessing

Common prepositions, transitional words, conjunction and connecting words, and articles (such as *a*, *above*, *and*, *at*, *the*, *for*, etc.), words without much lexical meaning (such as *be*, *he*, *him*, etc.), and proper nouns (such as *Intel*, *Charles*, *Berlin*, etc.) are straightforward and so should be excluded from the vocabulary of the underlying knowledge base. These form the list of stop words for our purpose (see, e.g., Table 1). Vocab Learn uses a word filter to eliminate stop words.

Table 2 shows an example of text and expert-assigned important words extracted from it. We can see that they contain no stop words.

4.2 Word Classifications

Vocab Learn provides two types of word classifications, one is based on word frequency and the other on word ranking.

Table 1: A subset of stop words for the word filter.

Company	Adobe, Amazon, Boeing, Cisco, Dell, ExxonMobil, facebook, FedEx, Gap, Google, Honda, Intel, iRobot, KFC, Microsoft, Nike, Nvidia, SanDisk, VMware, Walmart
Person	Abraham, Alexander, Braden, Cale, Charles, Edward, Emmy, Fabiano, Francesco, Gale, Giff,, Hilton, James, Jackson, Madison, Niko, Peter, Simeon, Tad, Yuma, Zed
Place	Acra, Berlin, Boston, Chicago, Duffield, Dortmund, Edgemont, Engelhard, Galesburg, Gladbrook, Graz, Jeffersonton, London, Lowell, Paris, Vienna
Common stopwords	am, an, and, as, at, be, both, but, by, can, do, each, etc, ever, for, has, he, here, hi, if, in, into, is, it, let, may, no, of, on, or, per, self, she, so, than, the, this, to, us, was, you

Table 2: An example of text and important words marked by exports.

Title: Discrete output feedback sliding mode control of second order systems - a moving switching line approach

Text: The sliding mode control systems (SMCS) for which the switching variable is designed independent of the initial conditions are known to be sensitive to parameter variations and extraneous disturbances during the reaching phase. For second order systems this drawback is eliminated by using the moving switching line technique where the switching line is initially designed to pass the initial conditions and is subsequently moved towards a predetermined switching line. In this paper, we make use of the above idea of moving switching line together with the reaching law approach to design a discrete output feedback sliding mode control. The main contributions of this work are such that we do not require to use system states as it makes use of only the output samples for designing the controller. and by using the moving switching line a low sensitivity system is obtained through shortening the reaching phase. Simulation results show that the fast output sampling feedback guarantees sliding motion similar to that obtained using state feedback

Expert-assigned important vocabularies: sliding, mode, control, switching, variable, parameter, variations, moving, line, discrete, output, feedback, fast, sampling, state

Frequency. A word w is more common if it appears in the corpus more frequently. Let $W_{\mathcal{D}}$ denote the bag of words contained in corpus $\mathcal{D}$ after removing stop words. Then the frequency of w with respect to $\mathcal{D}$ is given by $F_{\mathcal{D}}(w) = |\{w \in W_{\mathcal{D}}\}|/|W_{\mathcal{D}}|$. Sort the words in descending order of frequencies. Then words in the top 25% are classified as Very Frequent, words in the bottom 25% are Less Frequent, and words in the middle are Frequent.

Importance. Vocab Learn classifies vocabularies into three categories of importance. We may use a word ranking algorithm to do so. For example, we may select one of TFIDF, TextRank, and RAKE; or a linear combination of them that is best for the underlying dataset. When there is no way to determine which would be the best, we will use RAKE, for it provides the best F-measure for the binary classifications of important words over the Inspec abstracts in Computer Science and Information Technology (see Section 5). Sort words in descending order according to their rankings. Then words in the top 25% are classified as Very Important, words in the bottom 25% are Less Important, and words in the middle are Important.

4.3 Semantically Close Words

For each word $w \in V$, Vocab Learn finds three most semantically close words of w. It uses Word2Vec (Mikolov et al., 2013a) to train word embedding vectors over an out-of-band data such as a very large English Wikipedia corpus available at https://dumps.wikimedia.org/. Note that semantically close words may not be synonyms. In particular, Vocab Learn computes the similarity between the embedding vector of w and the embedding vectors for each word in the trained Word2Vec model. The semantic similarity of two embedding vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ is computed using cosine similarity as following:

$$Sim_{semantic}(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{u}\| \, \|\boldsymbol{v}\|},$$

where $\boldsymbol{u} \cdot \boldsymbol{v}$ is the inner product of two vectors and $\|\boldsymbol{u}\|$ is the length of the vector. Let $\mathbf{v}_w$ denote the embedding vector of w. Vocab Learn selects three different words $w_i \neq w$ $(i = 1, 2, 3)$ with the highest semantic similarity scores between $\boldsymbol{v}_w$ and $\boldsymbol{v}_{w_i}$. Table 3 is the top three semantic close words of the major words included in the text shown in Table 2.

Table 3: Examples of top three semantic close words.

slide	pivot, lock, swivel
mode	gameplay, configuration, functionality
control	power, advantage, monitor
switch	transfer, shift, transitioning
variable	parameter, binary, vector
parameter	variable, constant, vector
variation	variant, combination, difference
move	relocate, transfer, head
line	route, loop, mainline
discrete	finite, linear, stochastic
output	input, voltage, amplitude
feedback	input, response, feedforward
fast	slow, quick, agile
sample	measurement, calibration, estimation
state	federal, national, government

4.4 Look-alike Words

Vocab Learn uses a dynamic programming of finding minimal edit distance between two words to find three different look-alike words for w. A pseudocode of finding the minimal edit distance of given two words is given in Algorithm 1.

Algorithm 1: Minimal Edit Distance.

$X, Y \leftarrow$ two words
$x, y \leftarrow$ length of X and Y
$D(i,0) = i$
$D(0,j) = j$
for $i \in [1,x]$ **do**
  for $j \in [1,y]$ **do**
    if $X(i) == Y(j)$ **then**
      $D(i,j) = D(i-1,j-1)$
    if $X(i) \neq Y(j)$ **then**
      $D(i,j) = D(i-1,j-1)+2$
    $D(i,j) = min(D(i,j), D(i-1,j)+1)$
    $D(i,j) = min(D(i,j), D(i,j-1)+1)$
$D(x,y)$ is minimal edit distance of X and Y

Vocab Lean selects three different words for w with the smallest minimal edit distance scores $D(x,y)$ as look-alike words such that their semantic similarities with w are greater than a threshold (to ensure that they represent different meanings). To reduce redundancy, only singular nouns and the original forms of verbs are considered. Table 4 depicts the top three look-alike words of the major words in the text of Table 2.

Table 4: Examples of top three look-alike words.

slide	alive, bride, aside
mode	more, modem, mod
control	central, onto, monroe
switch	with, smith, width
variable	valuable, variance, durable
parameter	diameter, adapter, paradise
variation	aviation, radiation, valuation
move	more, movie, mode
line	nine, life, lane
discrete	disney, diameter, distant
output	outlet, input, outer
feedback	paperback, frederick, feeding
fast	fact, east, vast
sample	simple, maple, apple
state	estate, safe, static

4.5 Generation of Multiple-choice Single-answer Questions

To assist the user to learn new words efficiently and effectively, Vocab Learn selects words from the list of words that the user has not mastered according to the distribution of importance and frequency. For each new word w selected to be learned, Vocab Learn displays its definition and generates a multiple-choice single-answer question that includes w and three semantically close words. It may also generate a multiple-choice single-answer question that includes w and three look-alike words. Figure 2 is a conceptual example of the generated questions of word *discrete* with three semantically close words on the left and with three look-alike words on the right.

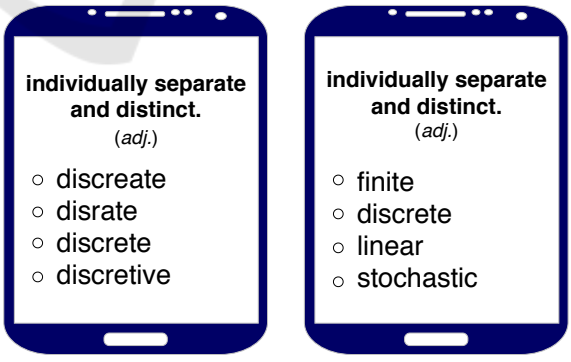

Figure 2: Examples of generated questions for learning word *discrete*.

5 EVALUATIONS

We describe the evaluation dataset, parameter settings, and experiments.

5.1 Dataset

We carry out evaluations on a collection of 1,500 Inspec abstracts in Computer Science and Information Technology journals (available at https://www.theiet.org/publishing/inspec/) as the training and validation dataset. Inspec abstracts were first used in (Hulth, 2003), and latter in TextRank (Mihalcea and Tarau, 2004) and in RAKE (Rose et al., 2010). For each Inspec abstract, there is a list of controlled keywords included in the abstract, and a list of uncontrolled keywords written by professional editors that may or may not appear in the abstract. The list of controlled keywords is typically very small, and the list of uncontrolled keywords typically includes controlled keywords, and is much larger. Following the previous practice of evaluating TextRank and RAKE, we will label uncontrolled keywords as important for the underlying knowledge base, and the rest of the extracted words as not-so-important. Note that it is impossible to obtain a full recall in this case, for some of the uncontrolled keywords may not appear in the abstracts. In our case, precision is more important and so it should be given a larger weight. We therefore use the $F_{0.5}$-score instead of the F_1-score to measure the accuracy of the ranking algorithms.

We partition the dataset into a training dataset and a test dataset. The training dataset consists of 1,000 abstracts and the collective uncontrolled keywords of these abstracts. The test dataset consists of the remaining 500 abstracts and the collective uncontrolled keywords of these abstracts.

Recall that Vocab Learn classifies words into three categories of importance. It is therefore more desirable to have a dataset that provides word rankings marked by exports over a corpus of documents to evaluate word-ranking algorithms. Unfortunately, we are not aware of any such dataset, and settled for a sub-optimal dataset of Inspec abstracts.

5.2 Comparison of Word Classifications

We will use the training dataset to train coefficients of linear combinations of TFIDF, TextRank, and RAKE to maximize the $F_{0.5}$-scores. Recall that we convert each verb appearing in the document to its stem form and each noun to its singular form. To carry out TextRank and RAKE, we treat all abstracts as one document. Namely, for TextRank, we construct a word graph for all words in the underlying abstracts (i.e., we treat all abstracts as one document) with a window size of 2 for co-occurrences of words to connect words. We similarly define $deg(w)$ and $freq(w)$ for each word $w \in V$ for RAKE.

We also consider linear combinations of TFIDF, TextRank, and RAKE to investigate if these algorithms may complement each other.

There are three different combinations of two algorithms and one combination of three algorithms. We name a linear combination of them by listing the names of the algorithms with non-zero coefficients in the chronological order of publications of the algorithms. For convenience, let A_1 denote TFIDF, A_2 TextRank, and A_3 RAKE. Then a linear combination of TFIDF and TextRank, denoted by TFIDF-TextRank, is $\lambda_1 A_1 + \lambda_2 A_2$, where $\lambda_i > 0$ and $\lambda_1 + \lambda_2 = 1$. In other words, the ranking score of each word w is equal to

$$\lambda_1 \text{score}_{A_1}(w) + \lambda_2 \text{score}_{A_2}(w).$$

The linear combination of other two algorithms is similarly defined, so is the linear combination of all three algorithms.

We select uniformly and independently at random n abstracts from the training dataset and extract a vocabulary from them. Let K_n be the total number of uncontrolled keywords for these abstracts. We label the top K_n words according to their rankings produced by the underlying algorithm as important, and the rest as not-so-important. We train coefficients for a combination of algorithms to maximize the $F_{0.5}$-score. We repeat the same experiments for the same value of n for 10 times in one round, and compute the average precision, recall, and $F_{0.5}$-score. We begin with $n = 50$, increase it by 50 in each round, until $n = 1,000$. For each underlying set of abstracts,

We then average the coefficients obtained in each round and set them up as the final coefficients. The results are shown in Table 5 (excluding single, where we use $\lambda_1 \parallel \lambda_2$ to denote the average coefficients of the corresponding two algorithms.

Table 5: Average coefficients of linear combinations.

TFIDF-TextRank	0.73 ‖ 0.27
TFIDF-RAKE	0.75 ‖ 0.25
TextRank-RAKE	0.75 ‖ 0.25
TFIDF-TextRank-RAKE	0.75 ‖ 0.11 ‖ 0.14

To test the accuracy of the linear combinations of the algorithms with coefficients trained on the training dataset (including the algorithms just by themselves), we select 250 abstracts uniformly and independently

Table 6: Examples of top six semantically close words and top six look-alike words.

	Top six semantically close words
slide	pivot, lock, swivel, tilt, hinge, fold
mode	gameplay, configuration, functionality, setup, multiplayer, module
control	power, advantage, monitor, protection, stability, responsibility
switch	transfer, shift, transitioning, move, swap, change
variable	parameter, binary, vector, fix, discrete, function
parameter	variable, constant, vector, scalar, gaussian, function
variation	variant, combination, difference, variety, version, type
move	relocate, transfer, head, return, progress, migrate
line	route, loop, mainline, stretch, tramline, branchline
discrete	finite, linear, stochastic, nonlinear, quantize, homogeneou
output	input, voltage, amplitude, waveform, impedance, torque
feedback	input, response, feedforward, interaction, cue, amplification
fast	slow, quick, agile, nimble, rapid, pace
sample	measurement, calibration, estimation, sequencing, resampling, analysi
state	federal, national, government, northcentral, legislative, regional
	Top six look-alike words
slide	alive, bride, aside, spider, oxide, life
mode	more, modem, mod, node, moore, mom
control	central, onto, monroe, intro, congo, context
switch	with, smith, width, swift, rich, speech
variable	valuable, variance, durable, portable, reliable, charitable
parameter	diameter, *adapter*, *paradise*, *prayer*, *carter*, *porter*
variation	aviation, radiation, valuation, validation, navigation, duration
move	more, movie, mode, moore, eve, mom
line	nine, life, lane, lone, lite, lake
discrete	*disney*, *dicke*, *diameter*, *distant*, *desperate*, *cigarette*
output	outlet, input, outer, *onto*, *struct*, *deputy*
feedback	*paperback*, *frederick*, *feeding*, *february*, *necklace*, *fireplace*
fast	fact, east, vast, fatty, nasa, font
sample	simple, maple, apple, sale, example, temple
state	estate, safe, static, statute, suite, sake

at random from the test dataset, and compute the precision, recall, and $F_{0.5}$-score for each algorithm. We repeat this for 10 times and average the precision scores, recall scores, and $F_{0.5}$-scores obtained from each round. Table 7 shows these results.

Table 7: Comparisons of precision, recall, and $F_{0.5}$-score of important words, with bold representing the highest score under each category.

Algorithms	**Precision**	**Recall**	**$F_{0.5}$-score**
TFIDF	0.59986	0.67460	0.61339
TextRank	0.64596	**0.76994**	**0.66744**
RAKE	**0.67766**	0.62861	0.66720
TFIDF-TextRank	0.62569	0.61035	0.62253
TFIDF-RAKE	0.64160	0.56343	0.62423
TextRank-RAKE	0.66701	0.58430	0.64863
All	0.63677	0.54691	0.61647

We can see that under this dataset, TextRank provides the best recall and best $F_{0.5}$-score, while RAKE provides the best precision.

5.3 Semantically Close Words and Look-alike Words

Table 6 shows examples of top six semantically close words for each important word extracted from Table 2, as well as top six look-alike words. Under look-alike words for each word (shown on the left column), the minimal edit distance d between it and each of the top six look-alike words is at most 2 ($d \leq 6$). Words with edit distance $d > 3$ are displayed in *italic*. Short words such as "mode", "line", and "fast" tend to have more look-alike words with edit distance $d \leq 3$, while longer words such as "parameter", "discrete", and "feedback" have more look-alike words with edit distance $d > 3$. Using these look-alike words to generate multiple-choice questions can help users to distinguish and remember the words to be learned.

6 CONCLUSIONS

We presented a text mining system called Vocab Learn to assist users to learn new words for an underlying knowledge base, and assess how well the user has learned these words. We described the technical details of classifying words, finding semantically close words, and look-alike words, and demonstrated the effects through experiments.

In a future project, we will conduct human subject research and quantify the effect of Vocab Learn in helping people to learn new vocabulary with respect to the underlying knowledge base.

REFERENCES

Al-Rahmi, W. M., Alias, N., Othman, M. S., Alzahrani, A. I., Alfarraj, O., Saged, A. A., and Rahman, N. S. A. (2018). Use of e-learning by university students in malaysian higher educational institutions: A case in universiti teknologi malaysia. *IEEE Access*, 6:14268–14276.

Chen, C.-M., Chen, L.-C., and Yang, S.-M. (2019a). An english vocabulary learning app with self-regulated learning mechanism to improve learning performance and motivation. *Computer Assisted Language Learning*, 32(3):237–260.

Chen, C.-M. and Chung, C.-J. (2008). Personalized mobile english vocabulary learning system based on item response theory and learning memory cycle. *Computers & Education*, 51(2):624–645.

Chen, C.-M., Liu, H., and Huang, H.-B. (2019b). Effects of a mobile game-based english vocabulary learning app on learners' perceptions and learning performance: A case study of taiwanese efl learners. *ReCALL*, 31(2):170–188.

Dickinson, D. K., Nesbitt, K. T., Collins, M. F., Hadley, E. B., Newman, K., Rivera, B. L., Ilgez, H., Nicolopoulou, A., Golinkoff, R. M., and Hirsh-Pasek, K. (2019). Teaching for breadth and depth of vocabulary knowledge: Learning from explicit and implicit instruction and the storybook texts. *Early Childhood Research Quarterly*, 47:341–356.

Godfroid, A., AHN, J., CHOI, I., BALLARD, L., CUI, Y., JOHNSTON, S., LEE, S., SARKAR, A., and YOON, H.-J. (2018). Incidental vocabulary learning in a natural reading context: an eye-tracking study. *Bilingualism: Language and Cognition*, 21(3):563–584.

Harmon, J. M. (2002). Teaching independent word learning strategies to struggling readers. *Journal of Adolescent & Adult Literacy*, 45(7):606–615.

Hulth, A. (2003). Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223. Association for Computational Linguistics.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Mihalcea, R. and Tarau, P. (2004). Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Murphy, A., Farley, H., and Rees, S. (2013). Revisiting the definition of mobile learning.

Nurmukhamedov, U. and Plonsky, L. (2018). Reflective and effective teaching of vocabulary. In *Issues in Applying SLA Theories toward Reflective and Effective Teaching*, pages 115–126. Brill Sense.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Park, Y. (2011). A pedagogical framework for mobile learning: Categorizing educational applications of mobile technologies into four types. *The international review of research in open and distributed learning*, 12(2):78–102.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Peters, K. (2007). m-learning: Positioning educators for a mobile, connected future. *The International Review of Research in Open and Distributed Learning*, 8(2).

Rose, S., Engel, D., Cramer, N., and Cowley, W. (2010). Automatic keyword extraction from individual documents. *Text mining: applications and theory*, pages 1–20.

Salton, G. and McGill, M. J. (1986). Introduction to modern information retrieval.

Teodorescu, A. (2015). Mobile learning and its impact on business english learning. *Procedia - Social and Behavioral Sciences*, 180:1535 – 1540. The 6th International Conference Edu World 2014 "Education Facing Contemporary World Issues", 7th - 9th November 2014.

Wilkerson, M., Griswold, W. G., and Simon, B. (2005). Ubiquitous presenter: increasing student access and control in a digital lecturing environment. In *ACM SIGCSE Bulletin*, volume 37, pages 116–120. ACM.

Xie, H., Zou, D., Zhang, R., Wang, M., and Kwan, R. (2019). Personalized word learning for university students: a profile-based method for e-learning systems. *Journal of Computing in Higher Education*, pages 1–17.