Word Recommendation For Crosswords Using Distributed Words Embedding

Charu Agarwal

Indian Insitute of Technology, Dharwad
160010038@iitdh.ac.in

November 22, 2019

Abstract

In this paper, we consider the problem of creating thematic crosswords from seed words. Typically, human constructors select words and clues based on some theme around which the grid can be created. Modern constructors frequently use software to speed the task of filling the rest of the grid. Although several programs are available for this task, they do not create themes and do not create a fill that is dense and thematic. We propose novel neural language-based algorithms specifically tailored to suggest effective word recommendations for crosswords. We prototype and test these algorithms against realistic inputs. We find that our technique is feasible, often suggesting good words. We also shed some light on algorithmic options.

1. Introduction

Crosswords are word puzzles that usually take the form of a square or a rectangular grid of white- and black-shaded squares. The game's goal is to fill the white squares with letters, forming words or phrases, by solving clues, which lead to the answers. Usually, the answer words and phrases are placed in the grid from left to right and from top to bottom. The black squares are used to separate words or phrases.

Ever since their introduction in 1913 by Arthur Wayne, crossword puzzles are used every day by millions of people not only for entertainment but also have applications in educational and rehabilitation contexts. As actress Betty White states: "I don't have any trouble memorizing lines because of the crossword puzzles I do every day to keep my mind a little limber". Crosswords are absorbing and fascinating, giving a mild amount of mental stimulation and recreation at the same time. They form an essential part of newspapers and magazines, and more recently are being used by organizations and schools as a teaching tool [1], for example, it has been used to study sociological concepts [2] and medical concepts [3].

Crossword construction is a difficult and complex art and is mainly done by experts. There are two main parts to the crossword construction problem: (1) selecting a "good" set of words to be fitted in a crossword and, (2) given a set of words, fitting the words in a grid while respecting the constraints of the crossword. While much work has been done for (2), by posing the crossword construction problem as a constraint satisfaction problem [4] or a heuristic-based search problem [5], the problem of (1) is still largely unsolved.

Crossword puzzles often have a theme, that is, the set of words to be filled in the crossword come from a theme. Category theme puzzles have theme elements as members of the same set, eg.

parts of a tree. Anniversary or tribute theme puzzles commemorate a specific person, place, or event eg. on October 7, 2011 The New York Times crossword commemorated the life of Apple CEO Steve Jobs who had died on October 5 which included words such as 'STEVE JOBS', 'IPOD', 'APPLE', 'MACINTOSH', among others. Synonym theme puzzles have the theme entries all contain synonyms e.g., a Los Angeles Times puzzle featuring a set of theme entries that contain the words RAVEN, JET, EBONY, and SABLE, all synonyms for "black". Typical crossword puzzles contain 4-5 theme entries and the rest of the entries are filled using a software, with particularly bad or obscure entries edited out by human experts.

The generation of thematic puzzles on specific subjects typically requires a great deal of human expert work. Software systems such as Crossword Compiler, CrossFire, Dr. Fill [6] and PuzzleMe have been created to speed up the process. However, none of these generate puzzles based on cohesive themes automatically. We exemplify this point with an example in figure 2 where we compare the results of our algorithm with a standard gridfilling software .

1.1. Our key contributions

- We propose new techniques of suggesting themed words for a crossword given seed words and clues. The main idea is that we leverage word semantics as captured by high-dimensional distributed word embeddings.
- We have created a working prototype system for recommmending words for Amuse Labs, a leading provider of crosswords. The effort included developing a product-level word suggestion algorithm based on the given word and clues, capable of giving near real-time suggestions to the users.
- We have evaluated the algorithm and the system by considering realistic examples.

1.2. Organization of the report

The rest of the report is organized as follows. Section 2 gives a brief overview of related work and word vector representations. Section 3 describes our methodology to suggest words for the crossword. Section 4 presents a method to generate distributed word embeddings from domain-specific corpora. Section 5 describes an attempt to derive relations between clues and answers. Section 6 explains the system developed. Finally, section 7 briefly discusses the results and section 8 concludes.

2. Related Work

In WebCrow-generation[7], topic-specific crosswords[1] are automatically generated where a user specifies a set of topics of expertise, and the automatic system generates a crossword that is targeted to his/her skills. While the results are good, this will not work in the case where the topic is not explicitly specified. In the most natural case, the designer would like to extend the words list by adding more related words to the crossword. For example, if a crossword designer suggests the words 'XMAS TREE', 'SANTA' and 'REINDEER', then it can be understood that the expert wants to design a Christmas themed puzzle.

The lexical relations encoded in WordNet, a human-curated lexical database, have been used in [8] to enhance the aesthetics of the resulting crossword, to construct crosswords with a thematic focus. However, this approach has several limitations as WordNet does not include much domain-specific terminology and does not uncover certain types of semantic relations.

[1] A topic-specific crossword is a crossword having most of the definition/answer pairs belonging to a given topic T.

To illustrate the need for themed word suggestions, we ran our algorithm on a sample crossword puzzle with the theme Gandhi. Figure 1 shows the crossword, along with some of the words suggested, highlighted in red. We can observe that a lot of closely related words to the theme have been suggested. For example, 'Sabarmati Ashram' is the famous residence of Gandhi. 'Harijans' was a term popularized by Gandhi to give respect to the communities that were traditionally labeled as untouchables. 'Hind Swaraj' is the title of a book authored by Gandhi. 'Bapu' is another name of Gandhi. Just as a comparison, we used a popular crossword construction tool Sparkling Fill[a] to suggest words for the same set of input words. While from the layout point of view, Sparkling Fill definitely improves the fill of the grid, it is not clear how words such as 'AAA', 'LII', 'WET', 'TRA', 'EEK' and, 'GREET' classify as a good fill. Of course, if we have a clever clue designer, it might be possible to overcome this difficulty, but it is difficult to use such words in general without reducing the quality of the crossword.

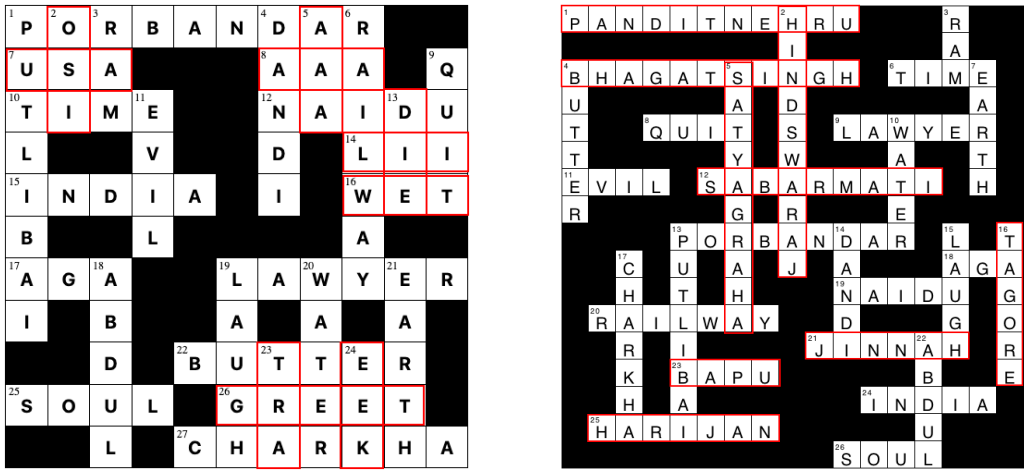

Figure 1: *Crossword based on the theme 'Mahatma Gandhi'. In the left image, the words highlighted in red have been suggested using the Sparkling Fill tool. In the right image, words highlighted in red have been suggested using the distributed word embeddings of the answers and the clues.*

[a]https://sparklingfill.com/

Figure 2: *Example result from our word recommendation algorithm, compared to a standard grid filling software. The puzzle used in this example was published by the Hindu on October 2, 2019 to commemorate Mahatma Gandhi's 150th birthday.*

Our algorithm is based on the recent work in learning word vector representations using neural networks [9] [10] [11] [12] [13]. The idea is that a word is represented by context in use. For example, consider the following two sentences: (1) I eat an *apple* everyday. (2) I eat an *orange* everyday. Since apple and orange are used in similar contexts, it might suggest that there is some similarity relation between these two words. In this formulation, each word is represented by a vector that is concatenated or averaged with other word vectors in a context, and the resulting vector is used to predict other words in the context.

The outcome is that after the model is trained, the word vectors are mapped into a vector space such that semantically-similar words have similar word representations (e.g. "happy" is close to "pleased"). [9] used a feedforward neural network with a linear projection layer and a non-linear hidden layer to jointly learn the word vector representations and a statistical language

model. [10] and [11] leveraged distributed word vectors to show that neural network based models match or outperform feature-engineered systems for standard Natural Language Processing (NLP) tasks that include part-of-speech tagging, chunking, named entity recognition, and semantic role labeling. [14] introduced a technique to learn better word embeddings by incorporating both local and global document context, and account for homonymy and polysemy by learning multiple embeddings per word. [15] [16] [17] introduced Word2Vec and the Skip-gram model, a very simple method for learning word vectors from large amounts of unstructured text data. The model avoids non-linear transformations and therefore makes training extremely efficient. This enables learning of high dimensional word vectors from huge datasets with billions of words, and millions of words in the vocabulary.
High-dimensional word vectors can capture subtle semantic relationships between words. For example, word vectors can be used to answer analogy questions using simple vector algebra: $v_{king} - v_{man} + v_{woman} = v_{queen}$, where v_x denotes the vector for the word x. [18] later introduced GloVe, Global Vectors for Word Representation, which combines word-word global co-occurrence statistics from a corpus, and context based learning similar to Word2Vec to deliver an improved word vector representation. Recently, Doc2Vec [24] was proposed as an extension to Word2Vec to learn document-level embeddings for sentences and paragraphs. A model that learns dense word vectors jointly with Dirichlet-distributed latent document-level mixtures of topic vectors was introduced in [25].
Word vectors are an attractive building block and are being used as input for many neural net based natural language tasks such as sentiment analysis [19] [20], question and answer systems, [21] [22], and others. More recently, the technique has been applied to recommenders and advertising [23]. These powerful, efficient models have shown very promising results in capturing both syntactic and semantic relationships between words in largescale text corpora, obtaining state-of-the-art results on a plethora of NLP tasks.

3. Methodology

In this section, we describe our algorithm for generating word suggestions from a given set of clues and answers. Our approach consists of three steps. First, we extract the nouns from the clues which involves sentence segmentation, tokenization and parts of speech tagging. Next, using the word embedding, we find the closest words for each clue-answer set. The clues help in adding context to the answer. Finally, we find the top-N most similar words and rank them based on importance. Algorithm 1 describes all the steps. We will use the example of the Gandhi puzzle shown in Figure 2 to explain all the steps of the algorithm, while providing empirical results[2]. Table 1 shows the clues and the answers for the puzzle. Figure **??** shows a schematic diagram of the algorithm.

3.1. Word Embeddings

For the word embeddings, we used a model published by Google, which was trained on 100 billion words from a Google News dataset[3]. The model contains 300-dimensional vectors for 3 million words and phrases. The reason for choosing this dataset was that modern crosswords allow many non-dictionary words and phrases too. For example, some words only became popular (in terms of the crossword) in recent years: 'BAE' and 'LGBT' appeared four and five time respectively in

[2]While we suggest different strategies for each step, our goal is to not classify which one is better. The results are given just to get a sense of the different strategies.

[3]https://code.google.com/archive/p/word2vec/

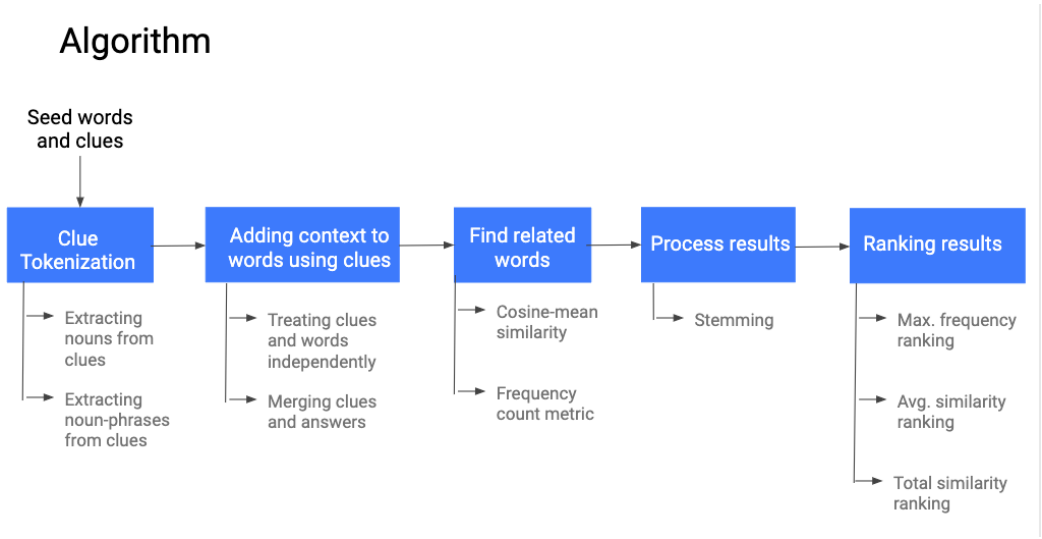

Figure 3: *Schematic Diagram of the algorithm for suggesting words for crossword.*

Algorithm 1 Algorithm to suggest themed-words for a crossword.

procedure SUGGEST_WORDS(model, words_list, clues_list) ▷ Returns word suggestions for a crossword grid.
    $clue\_nouns \leftarrow extract\_clue\_nouns(clues\_list)$ ▷ Tokenize clues. Ref. section 3.3.

    ▷ Combine clues and answers. Ref. section 3.4.
    $clue\_words\_combined \leftarrow combine(words\_list, clues\_nouns)$

    ▷ Filter words not in model vocabulary.
$filter\_words\_not\_in\_vocab(model, clue\_words\_combined)$

    ▷ Find related words. Ref. section 3.2.
    $word\_suggestions \leftarrow get\_related\_words(clue\_words\_combined)$

    ▷ Process the results. Ref. section 3.5.
    $processed\_results \leftarrow process\_results(word\_suggestions)$

    ▷ Rank the results. Ref. section 3.6.
    $ranked\_suggestions \leftarrow rank\_words(processed\_results, words\_list, clues\_list, model)$

    return *ranked_suggestions*
end procedure

Answer	Clue
Porbandar	Gandhi's birthplace
Putli Bai	Gandhi's mother
Ram	Hey _____ : Gandhi's last words
Time	Gandhi was _____ Magazine's Man of the Year in 1930
India	Young _____ : A journal published by Gandhi
Aga	Gandhi and Kasturba were jailed at _____ Khan's palace
Abdul	_____ Khan Gaffar Khan was also known as Frontier Gandhi
Soul	Mahatma means Great _____
Charkha	The spinning wheel made iconic by Gandhi
butter	The villagers want bread not _____ : quote by Gandhi
lawyer	Gandhi's profession in South Africa
Naidu	Sarojni _____ became president of the Indian National Congress after Gandhi
Railway	Gandhi was thrown out of the train at Pietermaritzburg _____ Station
Quit	Gandhi started the _____ India Movement in 1942
laugh	First they ignore you, then they _____ at you, then they fight you, then you win: quote by Gandhi
water	We may not be God, but we are of God, even as a little _____ drop is of the ocean: quote by Gandhi
earth	_____ provides enough to satify every man's needs, but not every man's greed: quote by Gandhi
evil	Good and _____ are found together: quote by Gandhi
Dandi	Gandhi led the Salt March to this beach

Table 1: *Running example seed input for puzzle published by the Hindu in commemoration of Mahatma Gandhi's 150th birthday.*

the year 2017 in the New York Times crosswords [26]. News is the best source of upcoming trends and modern vocabulary.

3.2. Finding related words

We first used only the answers to find word recommendations. Two metrics were used:

1. Cosine-mean similarity: This method computes cosine similarity between a simple mean of the projection weight vectors of the given words and returns the top-N most similar vecs from the training set.
$$similarity = cos\theta = \frac{X.Y}{\|X\|.\|Y\|} \tag{1}$$
2. Frequency count: This method finds the words which occur most number of times within top-N cosine distance of each input word.

Table 2 shows the results of applying both the metrics on the Gandhi puzzle. 'Karmabhoomi' was a novel inspired by Gandhi's satyagraha movement. The rest of the words from metric 1 are not very related to Gandhi, but more to politicians and the last word 'By Riyanki Das' is the name of a

reporter, taken from an article. In contrast, 'Gujarat' was the birthplace of Gandhi, 'Bhavnagar' is the place where Gandhi received their education, and there is a Mahatma Gandhi (M.G.) road in 'Junagadh'. Frequency count metric performs better for this puzzle[4]. We ran the two metrics on diverse themed puzzles and found that, in most cases, the frequency count metric gives much better results.

Cosine-mean similarity metric	Frequency count metric
karma bhoomi	Gujarat
Birwa	Porbander
Lalooji	Bhavnagar
Shastriji	Valsad
By Riyanki Das	Junagadh
Photo C. Ratheesh	Navsari
Dutt saab	Bharuch
AP Photo Bikas	Ramanathapuram
Subhendu	Amreli
sanyasi	Bhatkal

Table 2: *Top 10 words suggested using the cosine-mean similarity metric and the frequency count metric for the Gandhi puzzle answers. The themed words correctly suggested are highlighted in red.*

3.3. Using clues to disambiguate words

We exploit the fact that in our problem domain, we not only have words but also clues, which can help in disambiguating the words. For example, the word 'Donald' alone may be ambiguous, but '_____ Trump' or '_____ Duck' will certainly add context. Going by this intuition, we decided to use the clues as well. We used two methods to tokenize the clues using standard natural language processing libraries:

1. Extracting nouns from clues
2. Extracting noun-phrases from clues

Table 3 shows the results of tokenizing the Gandhi puzzle clues. We observe that extracting nouns works better, since some words are lost while extracting noun phrases.

Nouns extracted	Noun-phrases extracted
['Gandhi','birthplace']	['Gandhi']
['Gandhi','mother']	['Gandhi']
['Hey','Gandhi','words']	['Gandhi']
['Gandhi','Magazine','Man','Year']	['Gandhi','Magazine']
['Young','journal','Gandhi']	['Young']

Table 3: *First 5 clues tokenized for the Gandhi puzzle.*

[4] All evaluations have been done qualitatively, may vary from person to person.

3.3.1 Generating noun-phrases from clues and answers

Since the standard libraries were not able to find good noun-phrases, we decided to find phrases of variable length by combining adjacent words in the clues and querying them in the Google News model. For the Gandhi puzzle, we found four noun-phrases from the clues and the answers: ['Aga_Khan', 'Abdul_Gaffar', 'Dandi_March', 'South_Africa']. While this technique helped in finding phrases, it did not show significant improvement in the results for our example. Nevertheless, this idea of combining words to form phrases will definitely make the input more meaningful.

3.4. Method of combining clue and answers

To use the clues and the answers together, we considered two heuristic methods:

1. Treating clue nouns and answers independently: In this approach, we simply added the clue nouns to the input list and used this words list to find similar words using the frequency count metric.

2. Merging the clues and the answers: In this approach, we first merge the clue nouns and the answer to generate k most-similar words. That is, for each clue c_i and the corresponding word w_i, we compute the cosine similarity between a simple mean of the projection weight vectors of c_i and w_i to compute the top-k most similar words $mean_i$. Using this list of $mean_i's$, we find similar words using the frequency count metric. This seemed intuitive as we want to capture the relation between the clues and the corresponding answers.

Table 4 shows the results of applying both methods on the Gandhi puzzle. If we treat each word independently, the model is not able to differentiate Mahatama Gandhi from Rahul Gandhi (an Indian politician) as both have Gandhi surname, due to which names of other politicians such as Karunanidhi and Gadkari are suggested. However, if we do not break the link between the clue and the answers, the model has a better understanding that Gandhi refers to the great freedom fighter Mahatma Gandhi, and hence very related words such as Sabarmati Ashram (Gandhi's Residence), Satyagraha (movement started by Gandhi), Vinobha Bhave (spiritual successor of Mohandas Gandhi), are suggested. Merging clue and answers performs better for this puzzle.

3.5. Processing results

The model contains a lot of words which are almost same, for example, 'Gandhi', 'Gandhian', 'Gandhi ji', 'Mahatma Gandhiji', 'Mahatma', 'Mahatama Gandhi', 'Mohandas Gandhi', 'Mohandas Karamchand Gandhi'. Also, there are many misspellings of the same word in the model. For example, one of the closest words to 'Porbandar' is 'Porbander'. Since we do not want morphological variants of the same word suggested, we use a stemming algorithm which filters out words which have the same root/base as the input words, or the result words. The algorithm reduces the words "chocolates", "chocolatey", "choco" to the root word, "chocolate" and "retrieval", "retrieved", "retrieves" reduce to the stem "retrieve". Also, we remove duplicate words which only differ in the case. This is needed as the Google News model is case-sensitive. This helps us to promote diversity in the recommendations. Table 5 shows the results of applying the stemming algorithm on the results for the Gandhi puzzle. Clearly, more good words come to top by filtering words with the same root.

Without merging	With merging
salt satyagraha	Sabarmati Ashram
Karunanidhi	Shivaji Maharaj
Gadkari	satyagraha
Joshi	Acharya Vinoba Bhave
Vinoba	Swami Vivekananda
Bapu	Bapu
Ambedkar	Ambedkar
Pandit Nehru	Hind Swaraj
Swami Vivekananda	Pandit Nehru
Tagore	Sree Narayana Guru

Table 4: *Top 10 words suggested using the clues and the answers for the Gandhi puzzle. In the first column, the clue words and the answer words are treated independently, while in the second column, the clue words and the answer words are averaged by finding the closest word to each clue-answer pair, and then used to generate the result words. The themed words correctly suggested are highlighted in red.*

Before filtering	After filtering
Sabarmati Ashram	Sabarmati Ashram
Shivaji Maharaj	Shivaji Maharaj
Gandhi	satyagraha
Gandhian	Acharya Vinoba Bhave
Gandhi ji	Swami Vivekananda

Table 5: *Top 5 words suggested using the clues and the answers for the Gandhi puzzle before and after using a stemming algorithm to filter out words with the same root.*

3.6. Ranking of result words.

The final most important step is to rank the result words based on their importance. For each result word, we consider the input words which have generated the word. We considered three metrics for ranking:

1. Max frequency ranking: This metric ranks the words based on the number of input words which lie within δ distance of the result word.

2. Average similarity ranking: This metric ranks the words based on the average similarity from the input words.

3. Total similarity ranking: This metric ranks the words based on the sum of similarity from each word. This metric takes account of both frequency and the similarity of the word.

Table 6 shows the results of applying the different rankings to the result words. It is not clear which ranking is better than the others.

We wanted to observe the effect of input size on the accuracy of the results. Figure 5 shows the number of themed words correctly suggested out of top 10 words suggested by both methods described in 3.4 based on the input size. For small input sizes, the method of using the clues

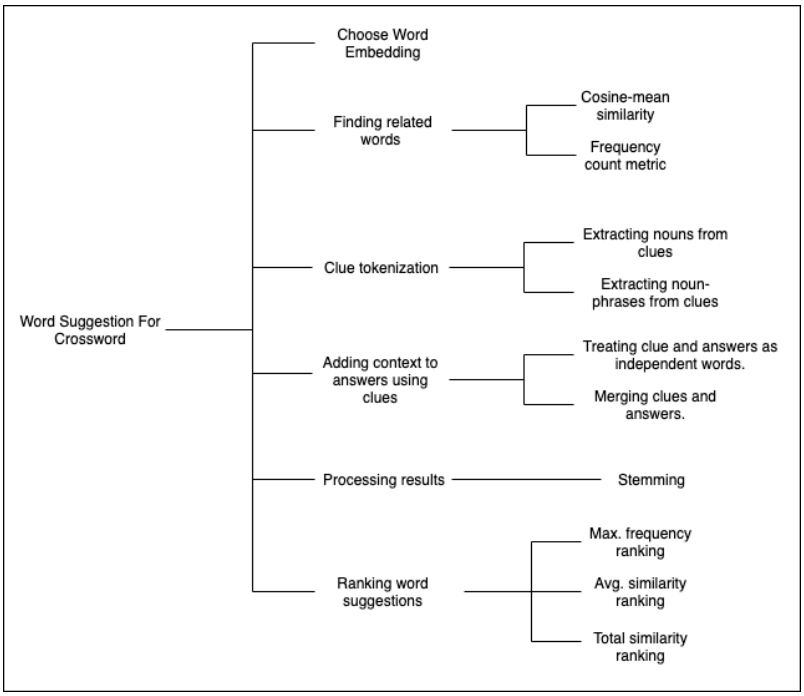

Figure 4: *Summary of techniques discussed for word suggestions for crosswords.*

and answers independently works better. For larger inputs, it is better to merge the clue and the answers first. This may be due to increase in the number of noise words as the size of the input words increases. Merging the clue and the answers may help in reducing the noise. Figure 4 summarizes the techniques discussed in this section.

Max frequency ranking	Avg similarity ranking	Total similarity ranking
Sabarmati Ashram	Bapu	Sabarmati Ashram
Acharya Vinoba Bhave	Swami Vivekananda	Shivaji Maharaj
Shivaji Maharaj	Pandit Nehru	satyagraha
satyagraha	Shri Guruji	Acharya Vinoba Bhave
Hind Swaraj	Hind Swaraj	Swami Vivekananda
Sree Narayana Guru	Tagore	Bapu
Ambedkar	Bhagat Singh	Ambedkar
Pandit Nehru	Ambedkar	Hind Swaraj
Bapu	Vinobha Bhave	Pandit Nehru
Swami Vivekananda	Pandit Jawaharlal Nehru	Sree Narayana Guru

Table 6: *Top 10 word suggestions ranked for the Gandhi puzzle using different metrics.*

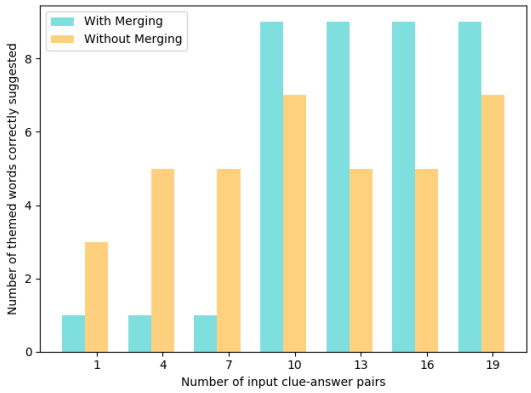

Figure 5: *Number of themed words correctly suggested out of top 10 words suggested based on input size.*

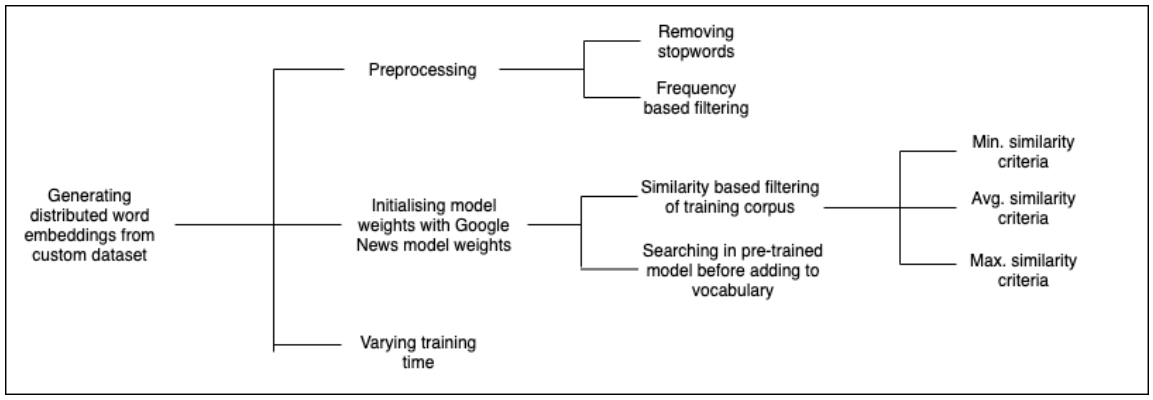

Figure 6: *Summary of techniques discussed for generating word embeddings from a domain-specific corpus.*

4. Training on domain-specific corpus

For some problems, the topic may not be present (eg. a small city or village) or the topic not well represented (eg. Indian history may not be as well documented in news as by authors and biographers.) in the pretrained Google News model. Another major drawback of the Google News model is that the model is case-sensitive, as discussed in section 3.5. Hence 'Gandhi' and 'gandhi' will have different vector representations. If we want to design puzzles which feel truly personal and special to the users, we cannot be limited by a pretrained model. Consider Dharwad, the home to the author's institute which is renowned for education, music and art. A simple query to the GoogleNews pretrained model to find the closest words to Dharwad include other well known cities. However, the model does not capture other aspects of Dharwad that are well known to the locals as Dharwad is not well-represented in the news. Figure 6 summarizes the strategies discussed for generating word embeddings from a domain-specific corpus.

4.1. Preprocessing

Considering we have a set of documents on which we would like to train our model, the first step is to tokenize the document. Tokenization involves breaking a document into a list of sentences. Next, we clean up the document by removing all punctuations. Finally, we split each sentence into

a list of words. Since we want our model to be case-insensitive, each word is converted to lower case. Table 7 shows a small document before and after preprocessing.

Before	**After**
I would like to say to the diligent reader of my writings and to others who are interested in them that I am not at all concerned with appearing to be consistent. In my search after Truth I have discarded many ideas and learnt many new things.	[['i', 'would', 'like', 'to', 'say', 'to', 'the', 'diligent', 'reader', 'of', 'my', 'writings', 'and', 'to', 'others', 'who', 'are', 'interested', 'in', 'them', 'that', 'i', 'am', 'not', 'at', 'all', 'concerned', 'with', 'appearing', 'to', 'be', 'consistent'], ['in', 'my', 'search', 'after', 'truth', 'i', 'have', 'discarded', 'many', 'ideas', 'and', 'learnt', 'many', 'new', 'things']]

Table 7: *Some lines of 'Hind Swaraj' by Gandhi, before and after preprocessing.*

4.1.1 Stopwords

Typically, documents contain many commonly used words such as "the", "a", "an", "in" which are not useful. These words are known as stopwords and need to be removed before processing. Stop words take up valuable processing time, and dilute the context. Hence, we took a standard list of stop words (~1073 words) and filtered them out from the document. Table 8 shows the document after preprocessing from 7 before and after removing stop words. As we can observe, a lot of noise words have been removed.

Before filtering stop words	**After filtering stop words**
[['i', 'would', 'like', 'to', 'say', 'to', 'the', 'diligent', 'reader', 'of', 'my', 'writings', 'and', 'to', 'others', 'who', 'are', 'interested', 'in', 'them', 'that', 'i', 'am', 'not', 'at', 'all', 'concerned', 'with', 'appearing', 'to', 'be', 'consistent'], ['in', 'my', 'search', 'after', 'truth', 'i', 'have', 'discarded', 'many', 'ideas', 'and', 'learnt', 'many', 'new', 'things']]	[['reader', 'diligent', 'reader', 'writings', 'others', 'interested', 'concerned', 'appearing', 'consistent'], ['search', 'truth', 'discarded', 'ideas', 'learnt', 'things']]

Table 8: *Effect of removing stop words from the corpus. The lines are taken from 'Hind Swaraj' by Gandhi.*

4.1.2 Frequency based filtering

In addition to the standard stop words, we also considered removing the top K frequently occuring words from the corpus as they might not be useful. This could not be implemented as after removing the stopwords, the ten most frequently occuring words included 'Gandhi', 'India' and 'non-violence', among other important words.

4.2. Result

We trained the model using 14 books on Mahatma Gandhi, including his autobiographies and novels on his life. The total corpus contained 1,11,813 lines consisting of 8,04,432 words. The size of the vocabulary was 35,672. The training was done using both the Skip-gram model[5] and the CBOW model[6]. Table 9 shows the words suggested by the model after training for the Gandhi puzzle. From the results, we observe that CBOW model performs better for this puzzle.

Suggested Words (Skipgram Model)	Suggested Words (CBOW Model)
lala	vithalbhai
fazlul	rajagopalachari
hasan	besant
esq	monday
vora	patna
dutt	midnight
carl	pyarelal
flew	devadas
deshpande	journey
objecting	birthday

Table 9: *Top 10 word suggestions ranked for the Gandhi puzzle. The results in the first column are generated from a model trained on the Skip-Gram model. The results in the second column are generated from a model trained on the CBOW model. Both the models are trained on 14 books on Mahatma Gandhi. The themed words correctly suggested are highlighted in red.*

4.3. Initializing weights before training

The good performance of the Google News dataset is due to the large text corpus it has been trained on, about 100 billion words. In reality, it is almost impossible to find this much amount of data for a specific topic to train the model. Hence, we proposed to exploit the Google News model information by initializing the weights of our model with the weights of the Google News model. The vocabulary was taken as a union of the vocabulary of the Google News model and the vocabulary of the domain-specific text corpus. In addition to this, we proposed two ideas to better preprocess the document:

4.3.1 Similarity based filtering

One intuitive idea is to filter the words which are semantically far from the input words. This makes sense as such words are not useful for our topic. Now, there are three possible ways to decide the criteria for filtering:

1. Average Similarity Criteria: Filter if the average similarity of word from the input words is less than ϵ.

2. Minimum Similarity Criteria: Filter if the minimum similarity of word from the input words is less than ϵ.

[5]Skip-gram model is designed to predict the context.
[6]In CBOW, the model learns to predict the word by the context

3. Maximum Similarity Criteria: Filter if the maximum similarity of word from the input words is less than ϵ.

Table 10 shows the words suggested by the model after training for the Gandhi puzzle, with the training words filtered out according to different similarity criteria. From the results, we can observe that filtering significantly improves results. (Max. filtering may be same as no-filtering in this case as each word in the corpus may be having at least one word in the input list which is at least ϵ-similar.)

Avg. Similarity	**Min. Similarity**	**Max. Similarity**
honour	Sabarmati Ashram	spirit
bhangi	bhagavadgita	believe
travancore	tulsidas	means
gurudev	learnt	gokhale
never	comorin	better
transvaal	neighbours	true
labour	himalayas	possible
country	parsi	world
sanatanists	coates	nothing
life	ishopanishad	use

Table 10: *Top 10 word suggestions ranked for the Gandhi puzzle. The model has been initialized with the weights of the Google News model and training is done on 14 books on Mahatma Gandhi. Different filtering criteria have been used for preprocessing the document based on similarity (computed from Google News model) with input words. The themed words correctly suggested are highlighted in red.*

Effect of varying ϵ In order to observe the dependence of the results on the similarity threshold (ϵ), we varied ϵ and recorded the results for different values of ϵ. Table 11 shows the words suggested by the model after training for the Gandhi puzzle, with the training words filtered out according to different ϵ values. From the results, we can observe that the number of correct words do not vary significantly with different ϵ values (for the minimum-similarity criteria).

4.3.2 Searching in model before adding to vocabulary

Since not all words in the training corpus would necessarily be in the Google News model vocabulary, we simply retain such words. Also, in order to not have two vector representations for the same word, we first query the Google News model to find if any other case (uppercase, titlecase) of the word is already present in the model vocabulary. If present, we replace the word in our corpus with the case followed by the Google News word.

4.4. Effect of training time

We decided to increase the training time as increasing the number of epochs usually benefits the quality of the word representations. Table 12 shows the words suggested by the model after training for the Gandhi puzzle, with increasing training time. From the results, it cannot be said if training longer definitely improves results.

$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.2$
Sabarmati Ashram	Sabarmati Ashram	Sabarmati Ashram
neighbour	bhagavadgita	zoroastrianism
zoroastrianism	tulsidas	selfcontrol
bhagavadgita	learnt	varnashram
moonje	comorin	satyagrahi
tulsidas	neighbours	neighbour
tolstoy	himalayas	tulsidas
devanagari	parsi	gandhijinnah
karamchand	coates	godfearing
huq	ishopanishad	bhagavadgita

Table 11: *Top 10 word suggestions ranked for the Gandhi puzzle. The model has been initialized with the weights of the Google News model and training is done on 14 books on Mahatma Gandhi. Different values of ϵ have been used for preprocessing the document based on minimum similarity criteria (computed from Google News model) with input words. The themed words correctly suggested are highlighted in red.*

epochs $= 5$	*epochs* $= 15$	*epochs* $= 25$	*epochs* $= 35$
Sabarmati Ashram	know	true	yeravda
zoroastrianism	use	centre	mahatmaji
selfcontrol	labour	sevagram	tolstoy
varnashram	truth	santiniketan	gita
satyagrahi	tagore	know	kshitimohanbabu
neighbour	never	antiuntouchability	lessmore
tulsidas	country	today	oversight...
gandhijinnah	yeravda	rajkot	orlov
godfearing	gandhiji	hindumuslim	iare
bhagavadgita	long	rolland	segaon

Table 12: *Top 10 word suggestions ranked for the Gandhi puzzle. The model has been initialized with the weights of the Google News model and training is done on 14 books on Mahatma Gandhi, with increasing training time. The themed words correctly suggested are highlighted in red.*

5. Deriving relations between clues and answers

We wanted to derive relations between clues and answers. For example, if the user suggests 'Paris' as answer and '_____ : capital of France' as the clue, we wanted to see if we could suggest 'New Delhi' as answer and '_____ : capital of India' as the clue. An analogy question is a question that asks one to find the relationship between words. For instance, the answer to the question: man is to woman, what king is to _____ is "queen" since we have to use the relationship in the first part of the question to fill in the blank. Now, it has been shown that given words a, b and c, using vector representations produced by Word2Vec we can find d such that it satisfies $a : b :: c : d$ [16]. d is such that vec(d) is closest to vec(c) + vec(b) - vec(a) according to cosine distance (we discard the input words from the search).

One possible way of attempting this is as follows: we take the answer word a and for each word

```
Clue: captial of China  Answer: China
Clue: captial of China  Answer: Bejing
Clue: captial of China  Answer: Shanghai
Clue: captial of China  Answer: Beijng
Clue: captial of China  Answer: Guangzhou
Clue: captial of China  Answer: Chinese
Clue: captial of China  Answer: Taipei
Clue: captial of China  Answer: inBeijing
Clue: captial of China  Answer: Hu
Clue: captial of China  Answer: Beijing_Olympics
Clue: capital of Chinas  Answer: China
Clue: capital of Chinas  Answer: Bejing
Clue: capital of Shenzhen  Answer: Shanghai
Clue: capital of Chinas  Answer: Beijng
Clue: capital of Guangdong  Answer: Guangzhou
Clue: capital of Taiwan  Answer: Taipei
```

Figure 7: *Attempt to derive similar clues and answer pairs from the clue "_____ : capital of China" and answer "Beijing". Out of the 16 suggestions, only two are correct (highlighted in red).*

c_i in the clue, we do the following. Let $N(a)$ be the m nearest neighbours of a and $N(c_i)$ be the n nearest neighbours of c_i. Now for every pair (x, y) in $N(a) \times N(c_i)$, check if y satisfies equation 2. If y satisfies the equation, output a new clue-answer pair where $a_{new} = x$ and $c_{new} = c.replace(c_i, y)$.

$$vec(y) = \underset{vec(z)}{\arg\min}\, D(vec(z), vec(c_i) - vec(a) + vec(x)) \tag{2}$$

where $D(a, b)$ is the cosine distance between a and b.

Figure 7 shows the result of applying our naive algorithm to the clue "_____ : capital of China" and answer "Beijing". As we can observe, only two correct capital:country/province relation was captured. It is interesting to note that Word2Vec captures many false relations too. If the given word c is not of the same category as a, it will find a d it thinks is related to c in the same manner as a is related to b. Thus, to obtain the correct relation, one must ensure that c is of the same type as a.

6. System Development

In this section, we cover the details of our system implementation that led to the final product prototype. There were mainly two requirements for the system: (1) API-based recommendations which can be integrated with Amuse Lab's products such as Crossword, Wordsearch etc. (2) Web-based GUI for experiments and debugging. In Figure 8 we show an example of our word recommender, where 'sandesh', 'modak' and 'halwa' are suggested after some Indian sweets themed clues and words are given as input.

6.1. API

A python-based server listens for GET requests at `/similar-words`. The request accepts two query parameters: words and clues, which are list of strings. The response consists of four fields: (1) `suggested_words`: the words suggested for the crossword, (2) `words_not_in_vocab`:

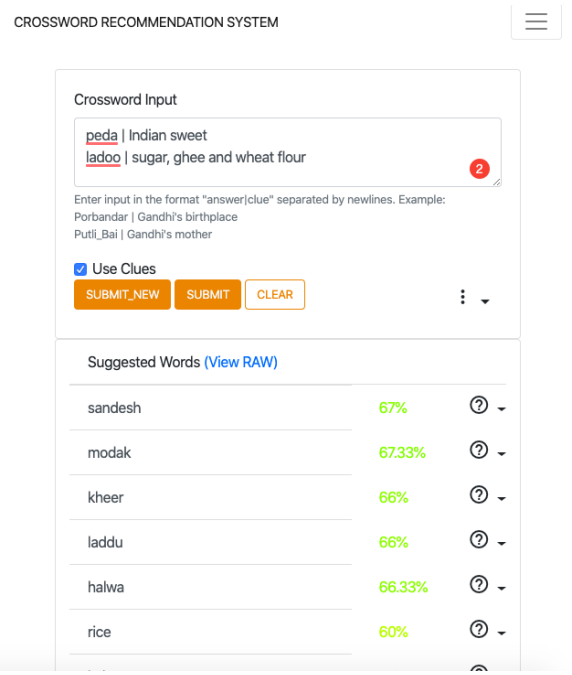

Figure 8: *Word recommendation product for Amuse Labs crosswords.*


```
{"suggested_words": ["vadas", "kaju", "barfi"], "words_not_in_vocab": [],
"words_map": {"vadas": [["peda", 0.7464032173156738], ["kheer",
0.7508113980293274]], "kaju": [["peda", 0.7262005805969238], ["kheer",
0.7184025049209595]], "barfi": [["peda", 0.7213644981384277], ["kheer",
0.7227498292922974]]}, "clue_nouns": [["sweet"], ["sweet", "milk"]]}
```


Figure 9: *Sample API response for two clues and answers. The results have been truncated for ease of understanding.*

to prompt user that an input word is not present in the model's vocabulary, (3) `words_map`: to capture the input words which have generated a result word, along with their cosine similarity (for debugging and understanding the results), and (4) `clue_nouns`: to capture the noun words extracted from the clues (for debugging). Figure 9 shows a sample API response for the query `similar-words/?words=peda&clues=Indian%20sweet&words=kheer&clues=sweet%20and%20milk`.

6.2. GUI

For the Graphical User Interface, we created an interactive web application which renders the API response in Javascript and HTML. The requests are made asynchronously using AJAX for seamless use, and are made either at the press of the submit button or pressing of ENTER key (after a clue-answer pair has been input). It supports adding words from the suggestions to the input (positive feedback) with a click, displays the input words responsible for a particular result word, displays color-coded score of each word (computed from input word cosine-similarities), supports viewing the nouns extracted from the clues and also allows to disable clues.

6.3. Timing issues

Using the system realtime, we found that the response time grows proportionally with the size of the input. For example, for an input of 19 clue-answer pairs, it takes ~17s for the response to return. Since, this is meant to be an interactive system, we cannot accept this much delay. On a careful analysis, we found that the most expensive operation of the algorithm is `model.most_similar()` which finds the k closest words for the given word as it searches in the entire model vector space. We proposed the idea to cache the top k similar words for each input word in a per-user session. This is because one word may be common across different clues, and more importantly, typically a user will enter the input line-by-line. We do not want to reprocess the previous input. Hence, after implementing caching, the average response time became $< 1s$.

7. Discussion

We would like to highlight few limitations of our work. First, the quality of suggestions degrades as the scope of the theme becomes more broad. For example, in the example we considered, the theme was highly specific due to which we were able to get very good results. If the theme is more general such as sports, the results may or may not be as good depending on the input. Secondly, the embeddings generated from a custom dataset cannot replace the embeddings from Google News as they have been trained on a significantly larger dataset and more sophisticated processors. The results can, in most cases, only supplement the results from a pretrained model. Our work to derive relations between clues and answers was preliminary and needs to be extended for practical use. All evaluations have been done qualitatively and the techniques have been developed using several heuristic based approaches.
In spite of these, we have considered many real-life themed puzzles and observed that our algorithm works well in most cases, and can help in a richer fill. We are able to fine-tune and apply the idea of word-vector representation to our particular use case of crosswords. This, in itself, is quite significant as we are able to introduce state-of-the-art machine learning techniques to crosswords.

8. Conclusions and Future Work

In this report we proposed a novel technique for suggesting words for crosswords that leverages understanding of word semantics using high-dimensional word vectors. We showed that our algorithm works well to extract a theme across a given set of clues and answers, and is able to give themed suggestions. Further, we provided serveral strategies for using clues, generating suggestions, ranking the words and generating embeddings from a domain-specific corpus. Since the word embeddings can be generated over any text corpus in an unsupervised fashion, our technique will work well for highly specialized domain crosswords too. We also presented the details of a system developed which is able to give near real-time word suggestions to the users. The next step could be to extend our algorithm to use the topic modeling techniques (LDA2Vec)[25] and document embeddings (Doc2Vec)[24]. We can utilize user feedback to further improve the performance of our word recommendation system. We can also have a rule based system which captures the most common categories of clue-answer pairs for automatic clue suggestions.

9. Acknowledgements

The author would like to thank Prof. Prabuchandran KJ and Prof. Sudheendra Hangal for guiding this project and for giving their invaluable time and input. The author would also like to thank Dept. of Computer Science and Engineering, IIT Dharwad, for providing the opportunity to work on this project.

References

[1] Edward K. Crossman & Sharyn M. Crossman (1983) The Crossword Puzzle as a Teaching Tool, Teaching of Psychology, 10:2, 98-99, DOI: *10.1207/s15328023top1002_10*.

[2] Childers, Cheryl D. "Using Crossword Puzzles as an Aid to Studying Sociological Concepts." Teaching Sociology, vol. 24, no. 2, 1996, pp. 231–235. JSTOR, *www.jstor.org/stable/1318816*.

[3] Anurag Saxena and Raenelle Nesbitt and Punam Pahwa and Sheryl Mills. Crossword Puzzles: Active Learning in Undergraduate Pathology and Medical Education, Archives of Pathology & Laboratory Medicine, vol. 133, no. 9, 2009, pp. 1457-1462, DOI: *10.1043/1543-2165-133.9.1457*.

[4] M. L. Ginsberg et al. Search Lessons Learned From Crossword Puzzles. AAAI'90 Proceedings of the eighth National conference on Artificial intelligence - Volume 1 Pages 210-215. Available *here*

[5] L. J. Mazlack. Computer Construction of Crossword Puzzles Using Precedence Relationships Articial Intelligence 7, pp 1{19, 1976. DOI: *10.1016/0004-3702(76)90019-9*.

[6] M. L. Ginsberg (2011). Dr.Fill: Crosswords and an Implemented Solver for Singly Weighted CSPs Jair.org. Retrieved 2012-03-12. *http://www.jair.org/papers/paper3437.html*.

[7] L. Rigutini, M. Diligenti, M. Maggini, and M. Gori, "Automatic generation of crossword puzzles," International Journal on Artificial Intelligence Tools, vol. 21, no. 03, 2012. [Online]. Available: *https://doi.org/10.1142/S0218213012500145*.

[8] Aherne, A. & C. Vogel (2006). Wordnet enhanced automatic crossword generation. In P. Sojka, K.-S. Choi, C. Fellbaum, & P. Vossen (Eds.), Proceedings of the Third International Wordnet Conference, pp. 139 – 145. Available *here*.

[9] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. (2003). A neural probabilistic language model. J. Mach. Learn. Res., 3:1137–1155

[10] R. Collobert and J. Weston. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th International Conference on Machine Learning, ICML '08, pages 160–167, New York, NY, USA, 2008. ACM.

[11] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. (2011). Natural language processing (almost) from scratch. J. Mach. Learn. Res., 12:2493–2537, Nov. 2011.

[12] A. Mnih and G. E. Hinton. (2009). A scalable hierarchical distributed language model. In Advances in neural information processing systems, pages 1081–1088, 2009.

[13] J. Turian, L. Ratinov, and Y. Bengio. (2010). Word representations: a simple and general method for semi-supervised learning. In Proceedings of the 48th annual meeting of the association for computational linguistics, pages 384–394. Association for Computational Linguistics, 2010.

[14] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng. (2012). Improving word representations via global context and multiple word prototypes. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1, ACL '12, pages 873–882, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean (2013). Efficient estimation of word representations in vector space. CoRR, abs/1301.3781, 2013.

[16] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013). Distributed representations of words and phrases and their compositionality. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, Advances in Neural Information Processing Systems 26, pages 3111–3119. Curran Associates, Inc., 2013.

[17] T. Mikolov, W.-T. Yih, and G. Zweig (2013). Linguistic regularities in continuous space word representations. In HLT-NAACL, pages 746–751, 2013.

[18] J. Pennington, R. Socher, and C. D. Manning (2014). Glove: Global vectors for word representation. In EMNLP, volume 14, pages 1532–1543, 2014.

[19] R. Socher, E. H. Huang, J. Pennin, C. D. Manning, and A. Y. Ng (2011). Dynamic pooling and unfolding recursive auto-encoders for paraphrase detection. In Advances in Neural Information Processing Systems, pages 801–809, 2011.

[20] R. Socher, C. C. Lin, C. Manning, and A. Y. Ng (2011). Parsing natural scenes and natural language with recursive neural networks. In Proceedings of the 28th international conference on machine learning (ICML-11), pages 129–136, 2011.

[21] A. Kumar, O. Irsoy, J. Su, J. Bradbury, R. English, B. Pierce, P. Ondruska, I. Gulrajani, and R. Socher (2015). Ask me anything: Dynamic memory networks for natural language processing. arXiv preprint arXiv:1506.07285, 2015.

[22] J. Weston, A. Bordes, S. Chopra, and T. Mikolov (2015). Towards ai-complete question answering: A set of prerequisite toy tasks. arXiv preprint arXiv:1502.05698, 2015.

[23] M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, and D. Sharp (2015). E-commerce in your inbox: Product recommendations at scale. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15, pages 1809–1818, New York, NY, USA, 2015. ACM.

[24] Le, T. Mikolov. (2014). Distributed Represenations of Sentences and Documents. In Proceedings of ICML 2014.

[25] Moody, Christopher E. (2016). Mixing dirichlet topic models and word embeddings to make lda2vec. arXiv preprint arXiv:1605.02019, 2016.

[26] Tan J. (2017). 24 Years of NYT Crossword Answers. *https://jtanwk.github.io/nytcrossword/*.