Modelling Sentence Generation from Sum of Word Embedding Vectors as a Mixed Integer Programming Problem

Lyndon White, Roberto Togneri, Wei Liu Mohammed Bennamoun
The University of Western Australia
35 Stirling Highway, Crawley, Western Australia
lyndon.white@research.uwa.edu.au
{roberto.togneri, wei.liu, mohammed.bennamoun}@uwa.edu.au

Abstract—Converting a sentence to a meaningful vector representation has uses in many NLP tasks, however very few methods allow that representation to be restored to a human readable sentence. Being able to generate sentences from the vector representations demonstrates the level of information maintained by the embedding representation – in this case a simple sum of word embeddings. We introduce such a method for moving from this vector representation back to the original sentences. This is done using a two stage process; first a greedy algorithm is utilised to convert the vector to a bag of words, and second a simple probabilistic language model is used to order the words to get back the sentence. To the best of our knowledge this is the first work to demonstrate quantitatively the ability to reproduce text from a large corpus based directly on its sentence embeddings.

1 Introduction

Generally sentence generation is the main task of the more broad natural language generation field; here we use the term only in the context of sentence generation from sentence vector representation. For our purposes, a sentence generation method has as its input a sentence embedding, and outputs the sentence which it corresponds to. The input is a vector, for example $\tilde{s} = [0.11, 0.57, -0.21, ..., 1.29]$, and the output is a sentence, for example "The boy was happy.".

Dinu and Baroni [1] motivates this work from a theoretical perspective given that a sentence encodes its meaning, and the vector encodes the same meaning, then it must be possible to translate in both directions between the natural language and the vector representation. In this paper, we present an implementation that indicates to some extent the equivalence between the natural language space and the sum of word embeddings (SOWE) vector representation space. This equivalence is shown by demonstrating a lower bound on the capacity of the vector representation to be used for sentence generation.

The current state of the art methods for sentence generation produce human readable sentences which are rough approximations of the intended sentence. These existing works are those of Iyyer, Boyd-Graber, and Daumé III [2] and Bowman, Vilnis, Vinyals, *et al.* [3]. Both these have been demonstrated to produce full sentences. These sentences are qualitatively shown to be loosely similar in meaning to the original sentences. Neither work has produced quantitative evaluations, making it hard to compare their performance. Both are detailed further in Section 2. Both these methods use encoder/decoder models trained through machine learning; we present here a more deterministic algorithmic approach, but restrict the input sentence vector to be the non-compositional sum of word embeddings representation.

Ritter, Long, Paperno, *et al.* [4] and White, Togneri, Liu, *et al.* [5] found that when classifying sentences into categories according to meaning, simple SOWE outperformed more complex sentence vector models. Both works used sentence embeddings as the input to classifiers. Ritter, Long, Paperno, *et al.* [4] classified challenging artificial sentences into categories based on the positional relationship described using Naïve Bayes. White, Togneri, Liu, *et al.* [5] classified real-world sentences into groups of semantically equivalent paraphrases. In the case of Ritter, Long, Paperno, *et al.* [4] this outperformed the next best representation by over 5%. In the case of White, Togneri, Liu, *et al.* [5] it was within a margin of 1% from the very best performing method. These results suggest that there is high consistency in the relationship between a point in the SOWE space, and the meaning of the sentence.

Wieting, Bansal, Gimpel, *et al.* [6] presented a sentence embedding based on the related average of word-embedding, showing excellent performance across several competitive tasks. They compared their method's performance against several models, including recurrent neural networks, and long short term memory (LSTM) architectures. It was found that their averaging method outperformed the more complex LSTM system, on most

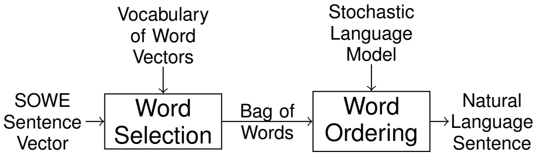
Vocabulary of Word Vectors
Stochastic Language Model
SOWE Sentence Vector
Word Selection
Bag of Words
Word Ordering
Natural Language Sentence

Figure 1. The Sel. BOW+Ord. process for the regenerating sentences from SOWE-type sentence vectors.

sentence similarity and entailment task. Thus these simple methods are worth further consideration. SOWE is the basis of the work presented in this paper.

Our method performs the sentence generation in two steps, as shown in Figure 1. It combines the work of White, Togneri, Liu, *et al.* [7] on generating bags of words (BOW) from sums of word embeddings (SOWE); with the work of Horvat and Byrne [8] on ordering BOW into sentences. The overall approach, of word selection followed by word ordering, can be used to generate proper sentences from SOWE vectors.

The rest of the paper is organized into the following sections. Section 2 discusses the prior work on sentence generation. Section 3 explains the problem in detail and how our method is used to solve it. Section 4 describes the settings used for evaluation. Section 5 presents the results of this evaluation. The paper concludes with Section 6 and a discussion of future work on this problem.

2 Related Works

To the best of our knowledge only three prior works exist in the area of sentence generation from embeddings. The first two (Dinu and Baroni [1], Iyyer, Boyd-Graber, and Daumé III [2]) are based on the recursive structures in language, while Bowman, Vilnis, Vinyals, *et al.* [3], uses the sequential structure.

Dinu and Baroni [1] extends the models described by Zanzotto, Korkontzelos, Fallucchi, *et al.* [9] and Guevara [10] for generation. The composition is described as a linear transformation of the input word embeddings to get an output vector, and another linear transformation to reverse the composition reconstructing the input. The linear transformation matrices are solved using least squares regression. This method of composing, can be applied recursively from words to phrases to clauses and so forth. It theoretically generalises to whole sentences, by recursively applying the composition or decomposition functions. However, Dinu and Baroni's work is quantitatively assessed only on direct reconstruction for decomposing Preposition-Noun and Adjective-Noun word phrases. In these cases where the decomposition function was trained directly on vectors generated using the dual composition function they were able to get perfect reconstruction on the word embedding based inputs.

Iyyer, Boyd-Graber, and Daumé III [2] extends the work of Socher, Huang, Pennington, *et al.* [11] defining an unfolding recursive dependency-tree recursive autoencoder (DT-RAE). Recursive neural networks are jointly trained for both composing the sentence's words into a vector, and for decomposing that vector into words. This composition and decomposition is done by reusing a composition neural network at each vertex of the dependency tree structure, with different weight matrices for each dependency relation. The total network is trained based on the accuracy of reproducing its input word embeddings. It can be used to generate sentences, if a dependency tree structure for the output is provided. This method was demonstrated quantitatively on five examples; the generated sentences were shown to be loosely semantically similar to the originals.

Bowman, Vilnis, Vinyals, *et al.* [3] uses a a modification of the variational autoencoder (VAE) [12] with natural language inputs and outputs, to learn the sentence representations. These input and output stages are performed using long short-term memory recurrent neural networks [13]. They demonstrate a number of uses of this technique, one of which is sentence generation, in the sense of this paper. While Bowman et al. do define a generative model, they do not seek to recreate a sentence purely from its vector input, but rather to produce a series of probability distributions on the words in the sentence. These distributions can be evaluated greedily, which the authors used to give three short examples of resynthesis. They found the sentence embeddings created captured largely syntactic and loose topical information.

We note that none of the aforementioned works present any quantitative evaluations on a corpus of full sentences. We suggest that that is due to difficulties in evaluation. As noted in Iyyer, Boyd-Graber, and Daumé III [2] and Bowman, Vilnis, Vinyals, *et al.* [3], they tend to output lose paraphrases, or roughly similar sentences. This itself is a separately useful achievement to pure exact sentence generation; but it is not one that allows ready interpretation of how much information is maintained by the embeddings. Demonstration of our method at generating the example sentences used in those work is available as supplementary material[1]. As our method often can exactly recreate the original sentence from its vector representation evaluation is simpler.

Unlike current sentence generation methods, the non-compositional BOW generation method of White, Togneri, Liu, *et al.* [7] generally outputs a BOW very close to the reference for that sentence – albeit at the cost of losing all word order information. It is because of this accuracy that we base our proposed sentence generation method on it (as detailed in Section 3.1). The word selection step we used is directly based on their greedy BOW generation method. We improve it for sentence generation by composing with a word ordering step to create the sentence generation process.

[1] http://white.ucc.asn.au/publications/White2016SOWE2Sent/

3 General Framework

As discussed in Section 1, and shown in Figure 1, the approach taken to generate the sentences from the vectors comes in two steps. First selecting the words used – this is done deterministically, based on a search of the embedding space. Second is to order them, which we solve by finding the most likely sequence according to a stochastic language model. Unlike the existing methods, this is a deterministic approach, rather than a machine learn method. The two subproblems which result from this split resemble more classical NP-Hard computer science problems; thus variations on known techniques can be used to solve them.

3.1 Word Selection

White, Togneri, Liu, *et al.* [7] approaches the BOW generation problem, as task of selecting the vectors that sum to be closest to a given vector. This is related to the knapsack and subset sum problems. They formally define the vector selection problem as:

$$(\tilde{s}, \mathcal{V}, d) \mapsto \underset{\left\{\forall \tilde{c} \in \mathbb{N}_0^{|\mathcal{V}|}\right\}}{\operatorname{argmin}} \; d(\tilde{s}, \sum_{\tilde{x}_j \in \mathcal{V}} \tilde{x}_j c_j)$$

to find the bag of vectors selected from the vocabulary set $\mathcal{V}$ which when summed is closest to the target vector $\tilde{s}$. Closeness is assessed with distance metric d. $\tilde{c}$ is the indicator function for that multi-set of vectors. As there is a one to one correspondence between word embeddings and their words, finding the vectors results in finding the words. White, Togneri, Liu, *et al.* [7] propose a greedy solution to the problem[2].

The key algorithm proposed by White, Togneri, Liu, *et al.* [7] is greedy addition. The idea is to greedily add vectors to a partial solution building towards a complete bag. This starts with an empty bag of word embeddings, and at each step the embedding space is searched for the vector which when added to the current partial solution results in the minimal distance to the target – when compared to other vectors from the vocabulary. This step is repeated until there are no vectors in the vocabulary that can be added without moving away from the solution. Then a fine-tuning step, n-substitution, is used to remove some simpler greedy mistakes.

The n-substitution step examines partial solutions (bags of vectors) and evaluates if it is possible to find a better solution by removing n elements and replacing them with up-to n different elements. The replacement search is exhaustive over the n-ary Cartesian product of the vocabulary. Only for $n = 1$ is it currently feasible for practical implementation outside of highly restricted vocabularies. Never-the-less even 1-substitution can be seen as lessening the greed of the algorithm, through allowing early decisions to be reconsidered in the full context of the partial solution. The algorithm does remain greedy, but many simple mistakes are avoided by n-substitution. The greedy addition and n-substitution processes are repeated until the solution converges.

3.2 The Ordering Problem

After the bag of words has been generated by the previous step, it must be ordered (sometimes called linearized). For example "are how , today hello ? you", is to be ordered into the sentence: "hello , how are you today ?". This problem cannot always be solved to a single correct solution. Mitchell and Lapata [14] gives the example of "It was not the sales manager who hit the bottle that day, but the office worker with the serious drinking problem." which has the same word content (though not punctuation) as "That day the office manager, who was drinking, hit the problem sales worker with a bottle, but it was not serious.". However, while a unique ordering cannot be guaranteed, finding the most likely word ordering is possible. There are several current methods for word ordering

To order the words we use a method based on the work of Horvat and Byrne [8], which uses simple trigrams. More recent works, such as beam-search and LSTM language model and proposed by Schmaltz, Rush, and Shieber [15]; or a syntactic rules based method such as presented in Zhang and Clark [16], could be used. These more powerful ordering methods internalise significant information about the language. The classical trigram language model we present is a clearer baseline for the capacity to regenerate the sentences; which then be improved by using such systems.

Horvat and Byrne [8] formulated the word ordering problem as a generalised asymmetrical travelling salesman problem (GA-TSP). Figure 2 shows an example of the connected graph for ordering five words. We extend beyond the approach of Horvat and Byrne [8] by reformulating the problem as a linear mixed integer programming problem (MIP). This allows us to take advantage of existing efficient solvers for this problem. Beyond the GA-TSP approach, a direct MIP formulation allows for increased descriptive flexibility and opens the way for further enhancement. Some of the constraints of a GA-TSP can be removed, or simplified in the direct MIP formulation for word ordering. For example, word ordering does have distinct and known start and end nodes (as shall be detailed in the next section). To formulate it as a GA-TSP it must be a tour without beginning or end. Horvat and Byrne [8] solve this by simply connecting the start to the end with a zero cost link. This is not needed if formulating this as a MIP problem, the start and end nodes can be treated as special cases. Being able to special case them as nodes known always to occur allows some simplification in the subtour elimination step. The formulation to mixed integer programming is otherwise reasonably standard.

[2] We also investigated beam search as a possible improvement over the greedy addition and n-substitution used by White, Togneri, Liu, *et al.* [7], but did not find significant improvement. The additional points considered by the beam tended to be words that would be chosen by the greedy addition in the later steps – thus few alternatives where found.

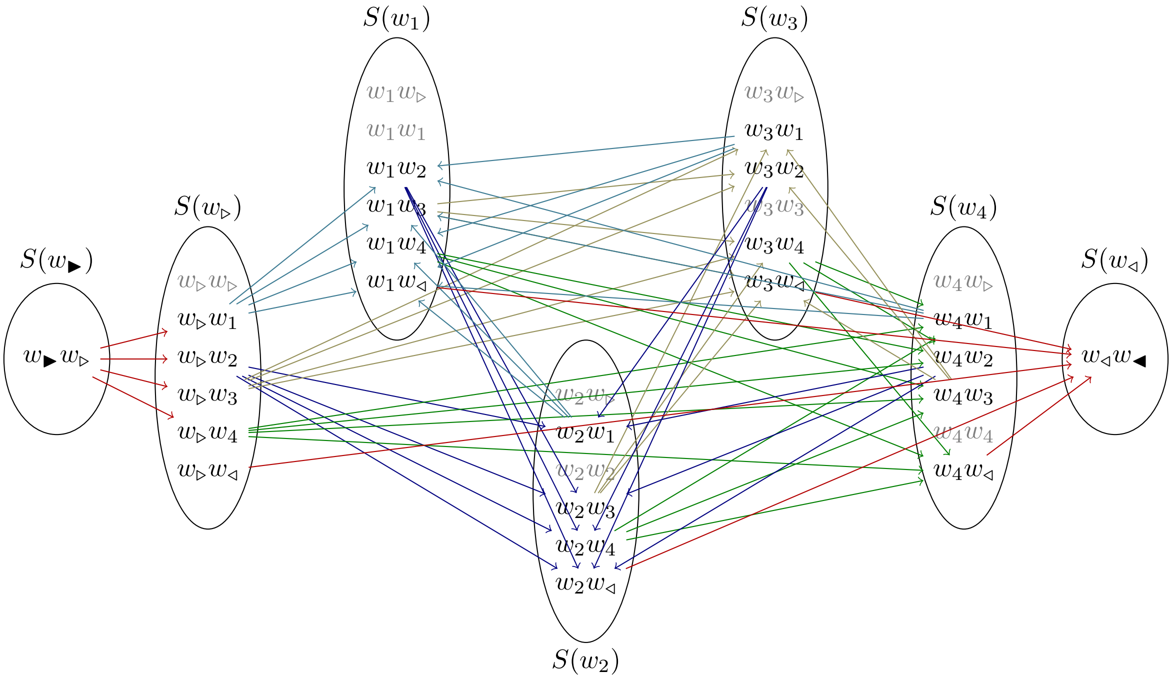

Figure 2. A graph showing the legal transitions between states, when the word-ordering problem is expressed similar to a GA-TSP. Each edge $\langle w_a, w_b\rangle \to \langle w_c, w_d\rangle$ has cost $-\log(P(w_c|w_a w_b)$. The nodes are grouped into districts (words). Nodes for invalid states are greyed out.

3.2.1 Notation

We will write w_i to represent a word from the bag $\mathcal{W}$ ($w_i \in \mathcal{W}$), with arbitrarily assigned unique subscripts. Where a word occurs with multiplicity greater than 1, it is assigned multiple subscripts, and is henceforth treated as a distinct word.

Each vertex is a sequence of two words, $\langle w_i, w_j\rangle \in \mathcal{W}^2$. This is a Markov state, consisting of a word w_j and its predecessor word w_i – a bigram.

Each edge between two vertices represents a transition from one state to another which forms a trigram. The start vertex is given by $\langle w_\blacktriangleright, w_\triangleright\rangle$, and the end by $\langle w_\triangleleft, w_\blacktriangleleft\rangle$. The pseudowords $w_\blacktriangleright, w_\triangleright, w_\triangleleft, w_\blacktriangleleft$ are added during the trigram models' training allowing knowledge about the beginning and ending of sentences to be incorporated.

The GA-TSP districts are given by the sets of all states that have a given word in the first position. The district for word w_i is given by $S(w_i) \subseteq \mathcal{W}^2$, defined as $S(w_i) = \{\langle w_i, w_j\rangle \mid \forall w_j \in \mathcal{W}\}$. It is required to visit every district, thus it is required to use every word. With this description, the problem can be formulated as a MIP optimisation problem.

3.2.2 Optimization Model

Every MIP problem has a set of variables to optimise, and a cost function that assesses how optimal a given choice of values for that variable is. The cost function for the word ordering problem must represent how unlikely a particular order is. The variables must represent the order taken. The variables are considered as a table (τ) which indicates if a particular transition between states is taken. Note that for any pair of Markov states $\langle w_a, w_b\rangle, \langle w_c, w_d\rangle$ is legal if and only if $b = c$, so we denote legal transitions as $\langle w_i, w_j\rangle \to \langle w_j, w_k\rangle$. Such a transition has cost:

$$C[\langle w_i, w_j\rangle, \langle w_j, w_k\rangle] = -\log\left(P(w_k|w_i, w_j)\right)$$

The table of transitions to be optimized is:

$$\tau[\langle w_i, w_j\rangle,\, \langle w_j, w_k\rangle] = \begin{cases} 1 & \begin{array}{l}\text{if transition from}\\ \langle w_i, w_j\rangle \to \langle w_j, w_k\rangle \text{ occurs}\end{array} \\ 0 & \text{otherwise} \end{cases}$$

The total cost to be minimized, is given by

$$C_{total}(\tau) = \sum_{\forall w_i, w_j, w_k \in \mathcal{W}^3} \tau[\langle w_i, w_j\rangle,\, \langle w_j, w_k\rangle] \cdot C[\langle w_i, w_j\rangle,\, \langle w_j, w_k\rangle]$$

The probability of a particular path (i.e. of a particular ordering) is thus given by $P(\tau) = e^{-C_{total}(\tau)}$

The word order can be found by following the links. The function $f_\tau(n)$ gives the word that, according to τ occurs in the nth position.

$$f_\tau(1) = \{w_a \mid w_a \in \mathcal{W} \wedge \tau[\langle w_\blacktriangleright, w_\triangleright\rangle, \langle w_\triangleright, w_a\rangle] = 1\}_1$$
$$f_\tau(2) = \{w_b \mid w_b \in \mathcal{W} \wedge \tau[\langle w_\triangleright, f_\tau(1)\rangle, \langle f_\tau(1), w_b\rangle] = 1\}_1$$
$$f_\tau(n) \underset{\text{when } n\geq 3}{=} \{w_c \mid w_c \in \mathcal{W} \wedge \tau[\langle f_\tau(n-2), f_\tau(n-1)\rangle, \langle f_\tau(n-1), w_c\rangle] = 1\}_1$$

The notation $\{\cdot\}_1$ indicates taking a singleton set's only element. The constraints on τ ensure that each set is a singleton.

3.2.3 Constraints

The requirements of the problem, place various constraints on to τ: The Markov state must be maintained: $\forall\langle w_a, w_b\rangle, \langle w_c, w_d\rangle \in \mathcal{W}^2$:

$$w_b \neq w_c \implies \tau[\langle w_a, w_b\rangle,\ \langle w_c, w_d\rangle] = 0$$

Every node entered must also be exited – except those at the beginning and end.
$\forall\langle w_i, w_j\rangle \in \mathcal{W}^2 \backslash \{\langle w_{\blacktriangleright}, w_{\triangleright}\rangle, \langle w_{\triangleleft}, w_{\blacktriangleleft}\rangle\}$:

$$\sum_{\forall\langle w_a, w_b\rangle \in \mathcal{W}^2} \tau[\langle w_a, w_b\rangle, \langle w_i, w_j\rangle] = \sum_{\forall\langle w_c, w_d\rangle \in \mathcal{W}^2} \tau[\langle w_i, w_j\rangle, \langle w_c, w_d\rangle]$$

Every district must be entered exactly once. i.e. every word must be placed in a single position in the sequence. $\forall w_i \in \mathcal{W} \backslash \{w_{\blacktriangleright}, w_{\blacktriangleleft}\}$:

$$\sum_{\substack{\forall\langle w_i, w_j\rangle \in S(w_i) \\ \forall\langle w_a, w_b\rangle \in \mathcal{W}^2}} \sum \tau[\langle w_a, w_b\rangle,\ \langle w_i, w_j\rangle] = 1$$

To allow the feasibility checker to detect if ordering the words is impossible, transitions of zero probability are also forbidden. i.e. if $P(w_n | w_{n-2}, w_{n-1}) = 0$ then $\tau[\langle w_{n-2}, w_{n-1}\rangle, \langle w_{n-1}, w_n\rangle] = 0$. These transitions, if not expressly forbidden, would never occur in an optimal solution in any case, as they have infinitely high cost.

3.2.3.1 Lazy Subtour Elimination Constraints: The problem as formulated above can be input into a MIPS solver. However, like similar formulations of the travelling salesman problem, some solutions will have subtours. As is usual callbacks are used to impose lazy constraints to forbid such solutions at run-time. However, the actual formulation of those constraints are different from a typical GA-TSP.

Given a potential solution τ meeting all other constraints, we proceed as follows.

The core path – which starts at $\langle w_{\blacktriangleright}, w_{\triangleright}\rangle$ and ends at $\langle w_{\triangleleft}, w_{\blacktriangleleft}\rangle$ can be found. This is done by practically following the links from the start node, and accumulating them into a set $T \subseteq \mathcal{W}^2$

From the core path, the set of words covered is given by $\mathcal{W}_T = \{w_i \mid \forall\langle w_i, w_j\rangle \in T\} \cup \{w_{\blacktriangleleft}\}$. If $\mathcal{W}_T = \mathcal{W}$ then there are no subtours and the core path is the complete path. Otherwise, there is a subtour to be eliminated.

If there is a subtour, then a constraint must be added to eliminate it. The constraint we define is that there must be a connection from at least one of the nodes in the district covered by the core path to one of the nodes in the districts not covered.

The districts covered by the tour are given by $S_T = \bigcup_{w_t \in \mathcal{W}_T} S(w_t)$. The subtour elimination constraint is given by

$$\sum_{\substack{\forall\langle w_{t1}, w_{t2}\rangle \in S_T \\ \forall\langle w_a, w_b\rangle \in \mathcal{W}^2 \backslash S_T}} \sum \tau[\langle w_{t1}, w_{t2}\rangle, \langle w_a, w_b\rangle] \geq 1$$

i.e. there must be a transition from one of the states featuring a word that is in the core path, to one of the states featuring a word not covered by the core path.

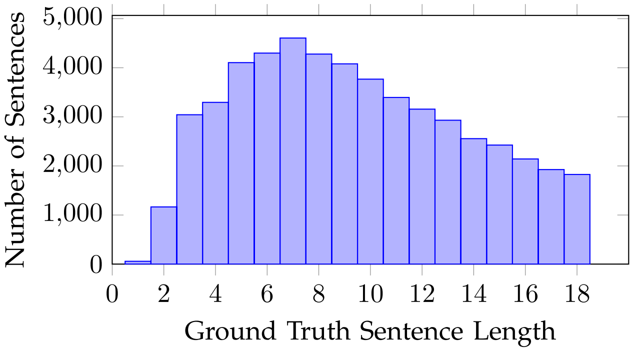

Figure 3. The distribution of the evaluation corpus after preprocessing.

This formulation around the notion of a core path that makes this different from typical subtour elimination in a GA-TSP. GA-TSP problems are not generally guaranteed to have any nodes which must occur. However, every word ordering problem is guaranteed to have such a node – the start and end nodes. Being able to identify the core path allows for reasonably simple subtour elimination constraint definition. Other subtour elimination constraints, however, also do exist.

4 Experimental Setup and Evaluations

This experimental data used in this evaluation was obtained from the data released with White, Togneri, Liu, *et al.* [7].[3]

4.1 Word Embeddings

GloVe representations of words are used in our evaluations [17]. GloVe was chosen because of the availability of a large pre-trained vocabulary of vectors.[4] The representations used for evaluation were pretrained on the 2014 Wikipedia and Gigaword 5. Other vector representations are presumed to function similarly. White, Togneri, Liu, *et al.* [7] showed that their word selection method significantly improves with higher dimensional embeddings. Due to their findings, we only evaluated 300 dimensional embeddings.

4.2 Corpus and Language Modelling

The evaluation was performed on a subset of the Books Corpus [18]. The corpus was preprocessed as in the work of White, Togneri, Liu, *et al.* [7]. This meant removing any sentences which used words not found in the embedding vocabulary.

After preprocessing, the base corpus, was split 90:10. 90% (59,694,016 sentences) of the corpus was used to fit a trigram model. This trigram language model was smoothed using the Knesler-Ney back-off method [19].

[3] Available online at http://white.ucc.asn.au/publications/White2016BOWgen/

[4] Available online at http://nlp.stanford.edu/projects/glove/

Process	Perfect Sentences	BLEU Score	Portion Feasible
Ref. BOW+Ord.	66.6%	0.806	99.6%
Sel. BOW+Ord.	62.2%	0.745	93.7%

Table 1
The overall performance of the Sel. BOW+Ord. sentence generation process when evaluated on the Books corpus.

Process	Perfect BOWs	Mean Precision	Mean Jaccard Index
Sel. BOW (only)	75.6%	0.912	0.891

Table 2
The performance of the word selection step, on the Books corpus. This table shows a subset of the results reported by White, Togneri, Liu, *et al.* [7].

The remaining 10% of the corpus was kept in reserve. From the 10%, 1% (66,464 sentences) were taken for testing. From this any sentences with length over 18 words were discarded – the time taken to evaluate longer sentences increases exponentially and becomes infeasible. This left a final test set of 53,055 sentences. Figure 3 shows the distribution of the evaluation corpus in terms of sentence length.

Note that the Books corpus contains many duplicate common sentences, as well as many duplicate books: according to the distribution site[5] only 7,087 out of 11,038 original books in the corpus are unique. We did not remove any further duplicates, which means there is a strong chance of a small overlap between the test set, and the set used to fit the trigrams.

4.3 Mixed Integer Programming

Gurobi version 6.5.0 was used to solve the MIP problems, invoked though the JuMP library [20]. During preliminary testing we found Gurobi to be significantly faster than the open source GLTK. Particularly for longer sentences, we found two orders of magnitude difference in speed for sentences of length 18. This is inline with the more extensive evaluations of Meindl and Templ [21]. Gurobi was run under default settings, other than being restricted to a single thread. Restricting the solver to a single thread allowed for parallel processing.

Implementation was in the Julia programming language [22]. The implementation, and non-summarised results are available for download.[6]

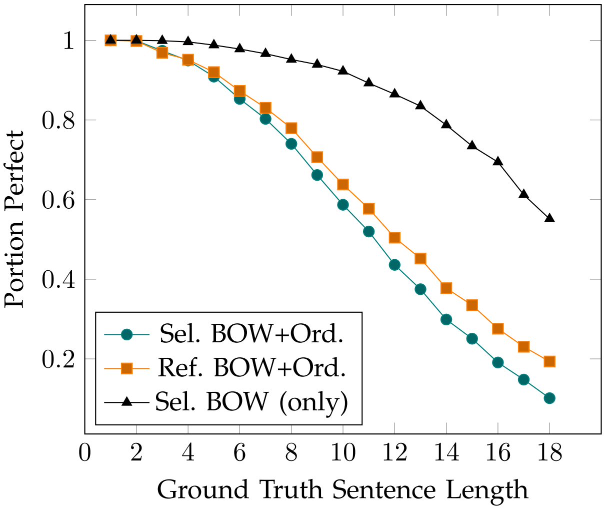

Figure 4. The portion of sentences reconstructed perfectly by the Sel. BOW+Ord. process. Shown also is the results on ordering only (Ref. BOW+Ord.), which orders the reference BOWS; and the portion of BOWs perfect from the word selection step only (Sel. BOW (only)) i.e. the input to the ordering step.

5 Results and Discussion

The overall results for our method (Sel. BOW+Ord.) sentence generation are shown in Table 1. Also shown are the results for just the ordering step, when the reference bag of words provided as the input (Ref. BOW+Ord.). The Perfect Sentences column shows the portion of the output sentences which exactly reproduce the input. The more forgiving BLEU Score [23] is shown to measure how close the generated sentence is to the original. The portion of cases for which there does exist a solution within the constraints of the MIP ordering problem is showin in Portion Feasible. In the other cases, where the MIP problem is unsolvable, for calculating the BLEU score, we order the BOW based on the order resulting from the word selection step, or in the reference case randomly.

Table 2 shows the results reported by [7] for the Word Selection step only (Sel. BOW (only)). The Perfect BOWs column reports the portion of the generated BOWs which perfectly match the reference BOWs. We also show the Mean Precision, averaged across all cases, this being the number of correct words generated, out of the total number of words generated. Similarly, the Mean Jaccard Index is shown, which is a measure of the similarities of the BOWs, being the size of the intersection of the generated BOW with the reference BOW, divided by the size of their union. We present these results to show how each step's performance impacts the overall system.

Both the Ref. BOW+Ord. and Sel. BOW (only) results place an upper bound on the performance of the overall approach (Sel. BOW+Ord.). The ordering only results (Ref. BOW+Ord.) show the best performance that can

[5] http://www.cs.toronto.edu/~mbweb/
[6] http://white.ucc.asn.au/publications/White2016SOWE2Sent/

be obtained in ordering with this language model, when no mistakes are made in selection. Similarly, the selection only results (Sel. BOW (only)) are bounding as no matter how good the word ordering method is, it cannot recreate perfectly accurate sentences using incorrect words.

It can be noted that Ref. BOW+Ord. and Sel. BOW+Ord. were significantly more accurate than the best results reported by Horvat and Byrne [8]. We attribute this to Horvat and Byrne preprocessing the evaluation corpora to remove the easier sentences with 4 or less words. We did not remove short sentences from the corpus. The performance on these sentences was particularly high, thus improving the overall results on ordering.

The overall resynthesis (Sel. BOW+Ord.) degrades as the sentence length increases as shown in Figure 4. It can be seen from the figure that sentence length is a critical factor in the performance. The performance drop is largely from the complexity in the ordering step when faced with long sentences. This is evident in Figure 4, as performance degrades at almost the same rate even when using the perfect BOW (compare Ref. BOW+Ord. vs Sel. BOW+Ord.); rather than being degraded by the failures in the word selection step (Sel. BOW (only)). We can conclude that sentences with word selection failures (Sel. BOW (only)) are also generally sentences which would have word ordering failures even with perfect BOW (Ref. BOW+Ord.). Thus improving word selection, without also improving ordering, would not have improved the overall results significantly.

From observing examples of the output of method we note that normally mistakes made in the word selection step result in an unorderable sentence. Failures in selection are likely to result in a BOW that cannot be grammatically combined e.g. missing conjunctions. This results in no feasible solutions to the word ordering problem.

Our method considers the word selection and word ordering as separate steps. This means that unorderable words can be selected if there is an error in the first step. This is not a problem for the existing methods of Iyyer, Boyd-Graber, and Daumé III [2] and of Bowman, Vilnis, Vinyals, *et al.* [3]. Iyyer, Boyd-Graber, and Daumé III [2] guarantees grammatical correctness, as the syntax tree must be provided as an input for resynthesis – thus key ordering information is indirectly provided and it is generated into. Bowman, Vilnis, Vinyals, *et al.* [3] on the other hand integrates the language model with the sentence embedding so that every point in the vector space includes information about word order. In general, it seems clear that incorporating knowledge about order, or at least co-occurrence probabilities, should be certain to improve the selection step. Even so the current simple approach has a strong capacity to get back the input, without such enhancement.

6 Conclusion

A method was presented for regenerating sentences, from the sum of a sentence's word embeddings. It uses sums of existing word embeddings, which are machine learnt to represent the sentences, and then generates natural language output, using only the embeddings and a simple trigram language model. Unlike existing methods, the generation method itself is deterministic rather than being based on machine-learnt encoder/decoder models. The method involved two steps, word selection and word ordering.

The first part is the word selection problem, of going from the sum of embeddings to a bag of words. To solve this we utilised the method presented in White, Togneri, Liu, *et al.* [7]. Their greedy algorithm was found to perform well at regenerating a BOW. The second part was word ordering. This was done through a MIP bases reformulation of the work of the graph-based work of Horvat and Byrne [8]. It was demonstrated that a probabilistic language model can be used to order the bag of words output to regenerate the original sentences. While it is certainly impossible to do this perfectly in every case, for many sentences the most likely ordering is correct.

From a theoretical basis the resolvability of the selection problem, presented by White, Togneri, Liu, *et al.* [7], shows that adding up the word embeddings does preserve the information on which words were used; particularly for higher dimensional embeddings. This shows clearly that collisions do not occur (at least with frequency) such that two unrelated sentences do not end up with the same SOWE representation. This work extends that by considering if the order can be recovered based on simple corpus statistics. Its recoverability is dependent, in part, on how frequent sentences with the same words in different order are in the corpus language – if they were very frequent then non-order preserving, non-compositional representations like SOWE would be poor at capturing meaning, and the ordering task would generally fail. As the method we presented generally does succeed, we can conclude that word order ambiguity is not a dominating problem. This supports the use of simple approaches like SOWE as a meaning representation for sentences – at least for sufficiently short sentences.

The technique was only evaluated on sentences with up to 18 words (inclusive), due to computational time limitations. Both accuracy and running time worsens exponentially as sentence length increases. With that said, short sentences are sufficient for many practical uses. For longer sentences, it is questionable as to the extent the information used is preserved by the SOWE representation – given they tend to have large substructures (like this one) compositional models are expected to be more useful. In evaluating such future representations, the method we present here is a useful baseline.

6.1 Acknowledgements

This research is supported by the Australian Postgraduate Award, and partially funded by Australian Research Council grants DP150102405 and LP110100050. Computational resources were provided by the National eResearch Collaboration Tools and Resources project (Nectar).

References

[1] G. Dinu and M. Baroni, "How to make words with vectors: Phrase generation in distributional semantics", in *Proceedings of ACL*, 2014, pp. 624–633.

[2] M. Iyyer, J. Boyd-Graber, and H. Daumé III, "Generating sentences from semantic vector space representations", in *NIPS Workshop on Learning Semantics*, 2014.

[3] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space", *ArXiv preprint arXiv:1511.06349*, 2015.

[4] S. Ritter, C. Long, D. Paperno, M. Baroni, M. Botvinick, and A. Goldberg, "Leveraging preposition ambiguity to assess compositional distributional models of semantics", *The Fourth Joint Conference on Lexical and Computational Semantics*, 2015.

[5] L. White, R. Togneri, W. Liu, and M. Bennamoun, "How well sentence embeddings capture meaning", in *Proceedings of the 20th Australasian Document Computing Symposium*, ser. ADCS '15, Parramatta, NSW, Australia: ACM, 2015, 9:1–9:8, ISBN: 978-1-4503-4040-3.

[6] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu, "Towards universal paraphrastic sentence embeddings", *International Conference on Learning Representations (ICLR)*, 2016.

[7] L. White, R. Togneri, W. Liu, and M. Bennamoun, "Generating bags of words from the sums of their word embeddings", in *17th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, 2016.

[8] M. Horvat and W. Byrne, "A graph-based approach to string regeneration.", in *EACL*, 2014, pp. 85–95.

[9] F. M. Zanzotto, I. Korkontzelos, F. Fallucchi, and S. Manandhar, "Estimating linear models for compositional distributional semantics", in *Proceedings of the 23rd International Conference on Computational Linguistics*, Association for Computational Linguistics, 2010, pp. 1263–1271.

[10] E. Guevara, "A regression model of adjective-noun compositionality in distributional semantics", in *Proceedings of the 2010 Workshop on Geometrical Models of Natural Language Semantics*, Association for Computational Linguistics, 2010, pp. 33–37.

[11] R. Socher, E. H. Huang, J. Pennington, A. Y. Ng, and C. D. Manning, "Dynamic pooling and unfolding recursive autoencoders for paraphrase detection", in *Advances in Neural Information Processing Systems 24*, 2011.

[12] D. P. Kingma and M. Welling, "Auto-encoding variational bayes", *ArXiv preprint arXiv:1312.6114*, 2013.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] J. Mitchell and M. Lapata, "Vector-based models of semantic composition.", in *ACL*, 2008, pp. 236–244.

[15] A. Schmaltz, A. M. Rush, and S. M. Shieber, "Word ordering without syntax", *ArXiv e-prints*, Apr. 2016. arXiv: 1604.08633 `[cs.CL]`.

[16] Y. Zhang and S. Clark, "Discriminative syntax-based word ordering for text generation", *Comput. Linguist.*, vol. 41, no. 3, pp. 503–538, Sep. 2015, ISSN: 0891-2017.

[17] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation", in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014, pp. 1532–1543.

[18] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books", in *ArXiv preprint arXiv:1506.06724*, 2015.

[19] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling", in *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, IEEE, vol. 1, 1995, pp. 181–184.

[20] M. Lubin and I. Dunning, "Computing in operations research using julia", *INFORMS Journal on Computing*, vol. 27, no. 2, pp. 238–248, 2015.

[21] B. Meindl and M. Templ, "Analysis of commercial and free and open source solvers for linear optimization problems", *Eurostat and Statistics Netherlands*, 2012.

[22] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing", 2014. arXiv: 1411.1607 `[cs.MS]`.

[23] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation", in *Proceedings of the 40th annual meeting on association for computational linguistics*, Association for Computational Linguistics, 2002, pp. 311–318.