Chapter 1

Generating Bags of Words from the Sums of their Word Embeddings

Abstract

Many methods have been proposed to generate sentence vector representations, such as recursive neural networks, latent distributed memory models, and the simple sum of word embeddings (SOWE). However, very few methods demonstrate the ability to reverse the process – recovering sentences from sentence embeddings. Amongst the many sentence embeddings, SOWE has been shown to maintain semantic meaning, so in this paper we introduce a method for moving from the SOWE representations back to the bag of words (BOW) for the original sentences. This is a part way step towards recovering the whole sentence and has useful theoretical and practical applications of its own. This is done using a greedy algorithm to convert the vector to a bag of words. To our knowledge this is the first such work. It demonstrates qualitatively the ability to recreate the words from a large corpus based on its sentence embeddings.

As well as practical applications for allowing classical information retrieval methods to be combined with more recent methods using the sums of word embeddings, the success of this method has theoretical implications on the degree of information maintained by the sum of embeddings representation. This lends some credence to the consideration of the SOWE as a dimensionality reduced, and meaning enhanced, data manifold for the bag of words.

1.1 Introduction

The task being tackled here is the *resynthesis* of bags of words (BOW) from sentence embedding representations. In particular the generation of BOW from vectors based on the sum of the sentence's constituent words' embeddings (SOWE). To the knowledge of the authors, this task has not been attempted before.

The motivations for this task are the same as in the related area of sentence generation. **Dinu2014CompositionalGeneration** observe that given a sentence has a given 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. A sub-step of this task is the unordered case (BOW), rather than true sentences, which we tackle in this paper. The success of the implementation does indicates the validity of this dual space theory, for the representations considered (where order is neglected). There are also some potential practical applications of such an implementation, often ranging around common vector space representations.

Given suitable bidirectional methods for converting between sentence embeddings and bags of words, the sentence embedding space can be employed as a *lingua franca* for translation between various forms of information – though with loss of word order information. The most obvious of which is literal translation between different natural languages; however the use extends beyond this.

Several approaches have been developed for representing images and sentences in a common vector space. This is then used to select a suitable caption from a list of candidates (**farhadi2010every**; **socherDTRNN**). Similar methods, creating a common space between images and SOWE of the keywords describing them, could be used to generate keyword descriptions using BOW resynthesis

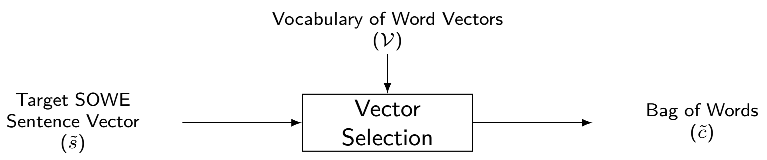

Figure 1.1: The process for the regenerating BOW from SOWE sentence embeddings.

– without any need for a list. This would allows classical word-based information retrieval and indexing techniques to be applied to images.

A similar use is the replacement of vector based extractive summarisation (**KaagebExtractiveSummaristation**; **yogatamaextractive**), with keyword based abstractive summarisation, which is the generation of a keyword summary from a document. The promising use of SOWE generation for all these applications is to have a separate model trained to take the source information (e.g. a picture for image description, or a cluster of sentences for abstract summarisation) as its input and train it to output a vector which is close to a target SOWE vector. This output can then be used to generate the sentence.

The method proposed in this paper has an input of a sum of word embeddings (SOWE) as the sentence embedding, and outputs the bag of word (BOW) which it corresponds to. The input is a vector for example $\tilde{s} = [-0.79, 1.27, 0.28, ..., -1.29]$, which approximates a SOWE vector, and outputs a BOW for example `{,: 1, best:1, it:2, of:2, the:2, times:2, was:2, worst:1}` – the BOW for the opening line of Dickens' *Tale of Two Cities*. Our method for BOW generation is shown in Figure 1.1, note that it takes as input only a word embedding vocabulary ($\mathcal{V}$) and the vector ($\tilde{s}$) to generate the BOW ($\tilde{c}$).

The rest of the paper is organized into the following sections. Section 1.2 introduces the area, discussing in general sentence models, and prior work on generation. Section 1.3 explains the problem in detail and our algorithm for solving it. Section 1.4 described the settings used for evaluation. Section 1.5 discusses the results of this evaluation. The paper presents its conclusions in Section 1.6, including a discussion of future work.

1.2 Background

The current state of the art for full sentence generation from sentence embeddings are the works of **iyyer2014generating** and **Bowman2015SmoothGeneration**. Both these advance beyond the earlier work of **Dinu2014CompositionalGeneration** which is only theorised to extend beyond short phrases. Iyyer et al. and Bowman et al. produce full sentences. These sentences are shown by examples to be loosely similar in meaning and structure to the original sentences. Neither works has produced quantitative evaluations, making it hard to determine between them. However, when applied to the various quantitative examples shown in both works neither is able to consistently reproduce exact matches. This motivates investigation on a simpler unordered task, converting a sum of word embeddings to bag of words, as investigated in this paper.

Bag of words is a classical natural language processing method for representing a text, sentence or document, commonly used in information retrieval. The text is represented as a multiset (or bag), this is an unordered count of how often each word occurs.

Word embeddings are vector representations of words. They have been shown to encode important syntactic and semantic properties. There are many different types of word embeddings (**Yin2015**). Two of the more notable are the SkipGrams of **mikolov2013efficient**; **mikolov2013linguisticsubstruct** and the Global Vector word representations (GloVe) of **pennington2014glove**. Beyond word representations are sentence embeddings.

Sentence embeddings represent sentences, which are often derived from word embeddings. Like word embeddings they can capture semantic and syntactic features. Sentence vector creation methods include the works of **le2014distributed** and **socher2014recursive**. Far simpler than those methods, is the sum of word embeddings (SOWE). SOWE, like BOW, draws significant criticism for not only disregarding sentence structure, but disregarding word order entirely when producing the sentence embedding. However, this weaknesses, may be offset by the improved discrimination allowed through words directly affecting the sentence embedding. It avoids the potential information loss through the indirection of more complex methods. Recent results suggest that this may allow it to be comparable overall to the more linguistically consistent embeddings when it comes to representing meaning.

White2015SentVecMeaning found that when classifying real-world sentences into groups of semantically equivalent paraphrases, that using SOWE as the input resulted in very accurate classifications. In that work White et al. partitioned the sentences into groups of paraphrases, then evaluated how well a linear SVM could classify unseen sentences into the class given by its meaning. They used this to evaluate a variety of different sentence embeddings techniques. They found that the classification accuracy when using SOWE as the input performed very similarly to the best performing methods – less than 0.6% worse on the harder task. From this they concluded that the mapping from the space of sentence meaning to the vector space of the SOWE, resulted in sentences with the same meaning going to distinct areas of the vector space.

RitterPosition presented a similar task on spacial-positional meaning, which used carefully constructed artificial data, for which the meanings of the words interacted non-simply – thus theoretically favouring the more complex sentence embeddings. In their evaluation the task was classification with a Naïve Bayes classifier into one of five categories of different spatial relationships. The best of the SOWE models they evaluated, outperformed the next best model by over 5%. These results suggest this simple method is still worth consideration for many sentence embedding representation based tasks. SOWE is therefore the basis of the work presented in this paper.

1.3 The Vector Selection Problem

At the core of this problem is what we call the Vector Selection Problem, to select word embedding vectors which sum to be closest to the target SOWE (the input). The word embeddings come from a known vector vocabulary, and are to be selected with potential repetition. Selecting the vectors equates to selecting the words, because there is a one to one correspondence between the word embedding vectors and their words. This relies on no two words having exactly the same embeddings – which is true for all current word embedding techniques.

The Vector Selection Problem is defined on $(\mathcal{V}, \tilde{s}, d)$
for a finite vocabulary of vectors $\mathcal{V}$, $\mathcal{V} \subset \mathbb{R}^n$, a target sentence embedding $\tilde{s}$, $\tilde{s} \in \mathbb{R}^n$, and any distance metric d, by:

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

$\tilde{x}_j$ is the vector embedding for the jth word in the vocabulary $\tilde{x}_j \in \mathcal{V}$ and c_j is the jth element of the count vector $\tilde{c}$ being optimised – it is the count of how many times the x_j occurs in approximation to the sum being assessed; and correspondingly it is the count of how many times the jth word from the vocabulary occurs in the bag of words. The selection problem is thus finding the right words with the right multiplicity, such that the sum of their vectors is as close to the input target vector, $\tilde{s}$, as possible.

1.3.1 NP-Hard Proof

The vector selection problem is NP-Hard. It is possible to reduce from any given instance of a *subset sum problem* to a vector selection problem. The *subset sum problem* is NP-complete (**karp1972reducibility**). It is defined: for some set of integers ($\mathcal{S} \subset \mathbb{Z}$), does there exist a subset ($\mathcal{L} \subseteq \mathcal{S}$) which sums to zero ($0 = \sum_{l_i \in \mathcal{L}} l_i$). A suitable metric, target vector and vocabulary of vectors corresponding to the elements $\mathcal{S}$ can be defined by a bijection; such that solving the vector selection problem will give the subset of vectors corresponding to a subset of $\mathcal{S}$ with the smallest sum; which if zero indicates that the subset sum does exists, and if nonzero indicates that no such subset ($\mathcal{L}$) exists. A fully detailed proof of the reduction from subset sum to the vector selection problem can be found on the first author's website. [1]

1.3.2 Selection Algorithm

The algorithm proposed here to solve the selection problem is a greedy iterative process. It is a fully deterministic method, requiring no training, beyond having the word embedding mapping provided. In each iteration, first a greedy search (Greedy Addition) for a path to the targeted sum point $\tilde{s}$ is done, followed by correction through substitution (n-Substitution). This process is repeated until no change is made to the path. The majority of the selection is done in the Greedy Addition step, while the n-substitution handles fine tuning.

[1] `http://white.ucc.asn.au/publications/White2015BOWgen/`

Greedy Addition

The greedy addition phase is characterised by adding the best vector to the bag at each step (see the pseudo-code in Algorithm 1). At each step, all the vectors in the current bag are summed, and then each vector in the vocabulary is added in turn to evaluate the new distance the new bag would have from the target, the bag which sums to be closest to the target becomes the current solution. This continues until there is no option to add any of the vectors without moving the sum away from the target. There is no bound on the size of the bag of vector (i.e. the length of the sentence) in this process, other than the greedy restriction against adding more vectors that do not get closer to the solution.

Greedy Addition works surprisingly well on its own, but it is enhanced with a fine tuning step, n-substitution, to decrease its greediness.

```
Data: the metric $d$
the target sum $\tilde{s}$
the vocabulary of vectors $\mathcal{V}$
the current best bag of vectors $bag_c$: initially $\emptyset$
Result: the modified $bag_c$ which sum to be as close as greedy search can get to the target $\tilde{s}$,
        under the metric $d$
begin
    $\tilde{t} \longleftarrow \sum_{x_i \in bag_c} x_i$
    while true do
        $\tilde{x}^* \longleftarrow \operatorname{argmin}_{x_j \in \mathcal{V}} d(\tilde{s}, \tilde{t} + \tilde{x}_j)$        /* exhaustive search of $\mathcal{V}$ */
        if $d(\tilde{s}, \tilde{t} + \tilde{x}^*) < d(\tilde{s}, \tilde{t})$ then
            $\tilde{t} \longleftarrow \tilde{t} + \tilde{x}^*$  $bag_c \longleftarrow bag_c \cup \{\tilde{x}^*\}$
        else
            return $bag_c$        /* No further improving step found */
        end
    end
end
```

Algorithm 1: Greedy Addition. In practical implementation, the bag of vectors can be represented as list of indices into columns of the embedding vocabulary matrix, and efficient matrix summation methods can be used.

n-Substitution

We define a new substitution based method for fine tuning solutions called n-substitution. It can be described as considering all subbags containing up to n elements, consider replacing them with a new sub-bag of up that size n from the vocabulary, including none at all, if that would result in the overall bag getting closer to the target $\tilde{s}$.

The reasoning behind performing the n-substitution is to correct for greedy mistakes. Consider the 1 dimensional case where $\mathcal{V} = 24, 25, 100$ and $\tilde{s} = 148$, $d(x, y) = |x - y|$. Greedy addition would give $bag_c = [100, 25, 24]$ for a distance of 1, but a perfect solution is $bag_c = [100, 24, 24]$ which is found using 1-substitution. This substitution method can be considered as re-evaluating past decisions in light of the future decisions. In this way it lessens the greed of the addition step.

The n-substitution phase has time complexity of $O(\binom{C}{n} V^n)$, for $C = \sum \tilde{c}$ i.e. current cardinality of bag_c. With large vocabularies it is only practical to consider 1-substitution. With the Brown Corpus, where $|\mathcal{V}| \approxeq 40,000$, it was found that 1-substitution provides a significant improvement over greedy addition alone. On a smaller trial corpora, where $|\mathcal{V}| \approxeq 1,000$, 2-substitution was used and found to give further improvement. In general it is possible to initially use 1-substitution, and if the overall algorithm converges to a poor solution (given the distance to the target is always known), then the selection algorithm can be retried from the converged solution, using 2-substitution and so forth. As n increases the greed overall decreases; at the limit the selection is not greedy at all, but is rather an exhaustive search.

1.4 Experimental Setup and Evaluations

1.4.1 Word Embeddings

GloVe representations of words (**pennington2014glove**) are used in our evaluations. There are many varieties of word embeddings which work with our algorithm. GloVe was chosen simply because of the availability of a large pre-trained vocabulary of vectors. The representations used for evaluation were pretrained on 2014 Wikipedia and Gigaword 5[2]. Preliminary results with SkipGrams from **mikolov2013efficient** suggested similar performance.

1.4.2 Corpora

The evaluation was performed on the Brown Corpus (**francis1979brown**) and on a subset of the Books Corpus (**moviebook**). The Brown Corpus was sourced with samples from a 500 fictional and non-fictional works from 1961. The Books Corpus was sourced from 11,038 unpublished novels. The Books Corpus is extremely large, containing roughly 74 million sentences. After preprocessing we randomly selected 0.1% of these for evaluation.

For simplicity of evaluation, sentences containing words not found in the pretrained vector vocabulary are excluded. These were generally rare mis-spellings and unique numbers (such as serial numbers). Similarly, words which are not used in the corpus are excluded from the vector vocabulary.

After the preprocessing the final corpora can be described as follows. The Brown Corpus has 42,004 sentences and a vocabulary of 40,485 words. Where-as, the Books Corpus has 66,464 sentences, and a vocabulary of 178,694 words. The vocabulary sizes are beyond what is suggested as necessary for most uses (**nation2006large**). These corpora remain sufficiently large and complex to quantitatively evaluate the algorithm.

1.4.3 Vector Selection

The Euclidean metric was used to measure how close potential solutions were to the target vector. The choice of distance metric controls the ranking of each vector by how close (or not) it brings the the partial sum to the target SOWE during the greedy selection process. Preliminary results on one-tenth of the Books Corpus used in the main evaluation found the Manhattan distance performed marginally worse than the Euclidean metric and took significantly longer to converge.

The commonly used cosine similarity, or the linked angular distance, have an issue of zero distances between distinct points – making them not true distance metrics. For example the SOWE of *"a can can can a can"* has a zero distance under those measures to the SOWE for *"a can can"*.[3] That example is a pathological, though valid sentence fragment. True metrics such as the Euclidean metric do not have this problem. Further investigation may find other better distance metrics for this step.

The Julia programming language (**Julia**), was used to create the implementation of the method, and the evaluation scripts for the results presented in the next section. This implementation, evaluation scripts, and the raw results are available online.[4]. Evaluation was carried out in parallel on a 12 core virtual machine, with 45Gb of RAM. Sufficient RAM is required to load the entire vector vocabulary in memory.

1.5 Results and Discussion

Table 1.1 shows examples of the output. Eight sentences which were used for demonstration of sentence generation in **iyyer2014generating**; **Bowman2015SmoothGeneration** have the BOW generation results shown. All examples except *(a)* and *(f)* are perfect. Example *(f)* is interesting as it seems that the contraction token *'re* was substituted for *are*, and *do* for *doing.* Inspections of the execution logs for running on the examples show that this was a greedy mistake that would be corrected using 2-substitution. Example *a* has many more mistakes.

The mistakes in Example *(a)* seem to be related to unusual nonword tokens, such as the three tokens with 13, 34, and 44 repetitions of the underscore character. These tokens appear in the very large Books corpus, and in the Wikipedia/Gigaword pretraining data used for word embeddings,

[2]Kindly made available online at `http://nlp.stanford.edu/projects/glove/`

[3]The same is true for any number of repetitions of the word *buffalo* – each of which forms a valid sentence as noted in **tymoczko1995sweet**

[4]`http://www.cicling.org/2016/data/97`

Table 1.1: Examples of the BOW Produced by our method using the Books Corpus vocabulary, compared to the Correct BOW from the reference sentences. The P and C columns show the the number of occurrences of each word in the Produced and Correct bags of words, respectively. **Bolded** lines highlight mistakes. Examples a-e were sourced from **iyyer2014generating**, Examples f-h from **Bowman2015SmoothGeneration**. Note that in example a, the *"___..._(n)"* represents *n* repeated underscores (without spaces).

(a) ralph waldo emerson dismissed this poet as the jingle man and james russell lowell called him three-fifths genius and two-fifths sheer fudge

Word	P	C
2008	**1**	**0**
___..._(13)	**1**	**0**
___..._(34)	**1**	**0**
___..._(44)	**1**	**0**
“	**1**	**0**
aldrick	**1**	**0**
and	2	2
as	**0**	**1**
both	**1**	**0**
called	**0**	**1**
dismissed	1	1
emerson	1	1
fudge	1	1
genius	1	1
hapless	**1**	**0**
him	1	1
hirsute	**1**	**0**
james	1	1
jingle	1	1
known	**1**	**0**
lowell	1	1
man	**0**	**1**
poet	1	1
ralph	1	1
russell	1	1
sheer	1	1
the	1	1
this	1	1
three-fifths	1	1
two-fifths	1	1
waldo	1	1
was	**1**	**0**

(b) thus she leaves her husband and child for aleksei vronsky but all ends sadly when she leaps in front of a train

Word	P	C
a	1	1
aleksei	1	1
all	1	1
and	1	1
but	1	1
child	1	1
ends	1	1
for	1	1
front	1	1
her	1	1
husband	1	1
in	1	1
leaps	1	1
leaves	1	1
of	1	1
sadly	1	1
she	2	2
thus	1	1
train	1	1
vronsky	1	1
when	1	1

(c) name this 1922 novel about leopold bloom written by james joyce

Word	P	C
1922	1	1
about	1	1
bloom	1	1
by	1	1
james	1	1
joyce	1	1
leopold	1	1
name	1	1
novel	1	1
this	1	1
written	1	1

(d) this is the basis of a comedy of manners first performed in 1892

Word	P	C
1892	1	1
a	1	1
basis	1	1
comedy	1	1
first	1	1
in	1	1
is	1	1
manners	1	1
of	2	2
performed	1	1
the	1	1
this	1	1

(e) in a third novel a sailor abandons the patna and meets marlow who in another novel meets kurtz in the congo

Word	P	C
a	2	2
abandons	1	1
and	1	1
another	1	1
congo	1	1
in	3	3
kurtz	1	1
marlow	1	1
meets	2	2
novel	2	2
patna	1	1
sailor	1	1
the	2	2
third	1	1
who	1	1

(f) how are you doing ?

Word	P	C
’re	**1**	**0**
?	1	1
are	**0**	**1**
do	**1**	**0**
doing	**0**	**1**
how	1	1
well	**1**	**0**
you	**0**	**1**

(g) we looked out at the setting sun .

Word	P	C
.	1	1
at	1	1
looked	1	1
out	1	1
setting	1	1
sun	1	1
the	1	1
we	1	1

(h) i went to the kitchen .

Word	P	C
.	1	1
i	1	1
kitchen	1	1
the	1	1
to	1	1
went	1	1

Table 1.2: The performance of the BOW generation method. Note the final line is for the Books Corpus, where-as the preceding are or the Brown Corpus.

Corpus	Embedding Dimensions	Portion Perfect	Mean Jaccard Score	Mean Precision	Mean Recall	Mean F1 Score
Brown	50	6.3%	0.175	0.242	0.274	0.265
Brown	100	19.4%	0.374	0.440	0.530	0.477
Brown	200	44.7%	0.639	0.695	0.753	0.720
Brown	300	70.4%	0.831	0.864	0.891	0.876
Books	300	75.6%	0.891	0.912	0.937	0.923

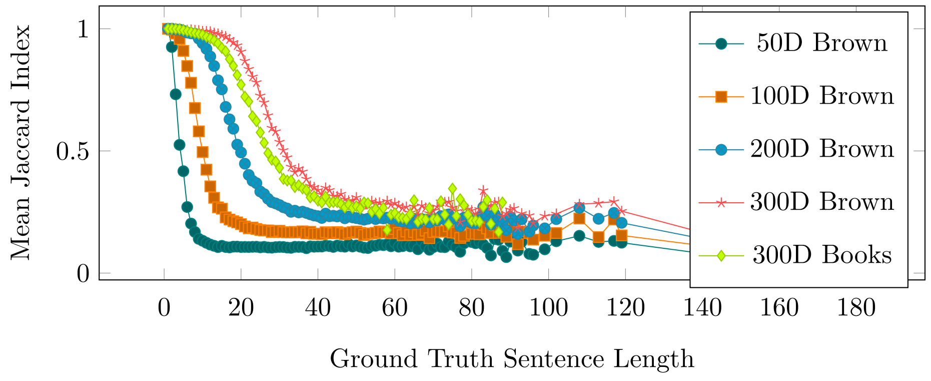

Figure 1.2: The mean Jaccard index achieved during the word selection step, shown against the ground truth length of the sentence. Note that the vast majority of sentences are in the far left end of the plot. The diminishing samples are also the cause of the roughness, as the sentence length increases.

but are generally devoid of meaning and are used as structural elements for formatting. We theorise that because of their rarity in the pre-training data they are assigned an unusual word-embedding by GloVE. There occurrence in this example suggests that better results may be obtained by pruning the vocabulary. Either manually, or via a minimum uni-gram frequency requirement. The examples overall highlight the generally high performance of the method, and evaluations on the full corpora confirm this.

Table 1.2 shows the quantitative performance of our method across both corpora. Five measures are reported. The most clear is the portion of exact matches – this is how often out of all the trials the method produced the exact correct bag of words. The remaining measures are all means across all the values of the measures in each trial. The Jaccard index is the portion of overlap between the reference BOW, and the output BOW – it is the cardinality of the intersection divided by that of the union. The precision is the portion of the output words that were correct; and the recall is the portion of all correct words which were output. For precision and recall word repetitions were treated as distinct. The F_1 score is the harmonic mean of precision and recall. The recall is higher than the precision, indicating that the method is more prone to producing additional incorrect words (lowering the precision), than to missing words out (which would lower the recall).

Initial investigation focused on the relationship between the number of dimensions in the word embedding and the performance. This was carried out on the smaller Brown corpus. Results confirmed the expectation that higher dimensional embeddings allow for better generation of words. The best performing embedding size (i.e. the largest) was then used to evaluate success on the Books Corpus. The increased accuracy when using higher dimensionality embeddings remains true at all sentence lengths.

As can be seen in Figure 1.2 sentence length is a very significant factor in the performance of our method. As the sentences increase in length, the number of mistakes increases. However, at higher embedding dimensionality the accuracy for most sentences is high. This is because most sentences are short. The third quartile on sentence length is 25 words for Brown, and 17 for the Books Corpus. This distribution difference is also responsible for the apparent better results on the Books Corpus, than on the Brown corpus.

While the results shown in Table 1.2 suggest that on the Books corpus the algorithm performs better, this is due to its much shorter average sentence length. When taken as a function of the sentence length, as shown in Figure 1.2, performance on the Books Corpus is worse than on the Brown Corpus. It can be concluded from this observation that increasing the size of the vocabulary does decrease success in BOW regeneration. Books Corpus vocabulary being over four times larger, while the other factors remained the same, resulted in lower performance. However, when taking all three factors into account, we note that increasing the *vocabulary size* has significantly less impact than increasing the *sentence length* or the *embedding dimensionality* on the performance.

1.6 Conclusion

A method was presented for how to regenerate a bag of words, from the sum of a sentence's word embeddings. This problem is NP-Hard. A greedy algorithm was found to perform well at the task, particularly for shorter sentences when high dimensional embeddings are used.

Resynthesis degraded as sentence length increased, but remained strong with higher dimensional models up to reasonable length. It also decreased as the vocabulary size increased, but significantly less so. The BOW generation method is functional with usefully large sentences and vocabulary.

From a theoretical basis the resolvability of the selection problem shows that adding up the word embeddings does preserve the information on which words were used; particularly for higher dimensional embeddings. This shows that collisions do not occur (at least not frequently) such that two unrelated sentences do not end up with the same SOWE representation.

This work did not investigate the performance under noisy input SOWEs – which occur in many potential applications. Noise may cause the input to better align with an unusual sum of word embeddings, than with its true value. For example it may be shifted to be very close a sentence embedding that is the sum of several hundred word embeddings. Investigating, and solving this may be required for applied uses of any technique that solves the vector selection problem.

More generally, future work in this area would be to use a stochastic language model to suggest suitable orderings for the bags of words. While this would not guarantee correct ordering every-time, we speculate that it could be used to find reasonable approximations often. Thus allowing this bag of words generation method to be used for full sentence generation, opening up a much wider range of applications.

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).