Part I

Publications

How Well Sentence Embeddings Capture Meaning

Lyndon White
lyndon.white@research.uwa.edu.au

Roberto Togneri
roberto.togneri@uwa.edu.au

Wei Liu
wei.liu@uwa.edu.au

Mohammed Bennamoun
mohammed.bennamoun@uwa.edu.au

The University of Western Australia
35 Stirling Highway, Crawley, Western Australia

ABSTRACT

Several approaches for embedding a sentence into a vector space have been developed. However, it is unclear to what extent the sentence's position in the vector space reflects its semantic meaning, rather than other factors such as syntactic structure. Depending on the model used for the embeddings this will vary – different models are suited for different down-stream applications. For applications such as machine translation and automated summarization, it is highly desirable to have semantic meaning encoded in the embedding. We consider this to be the quality of *semantic localization* for the model – how well the sentences' meanings coincides with their embedding's position in vector space. Currently the semantic localization is assessed indirectly through practical benchmarks for specific applications.

In this paper, we ground the semantic localization problem through a semantic classification task. The task is to classify sentences according to their meaning. A SVM with a linear kernel is used to perform the classification using the sentence vectors as its input. The sentences from subsets of two corpora, the Microsoft Research Paraphrase corpus and the Opinosis corpus, were partitioned according to their semantic equivalence. These partitions give the target classes for the classification task. Several existing models, including URAE, PV–DM and PV–DBOW, were assessed against a bag of words benchmark.

General Terms

Measurement, Performance, Experimentation

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Abstracting methods, Linguistic processing*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Language parsing and understanding*

ADCS, December 08-09, 2015, Parramatta, NSW, Australia

DOI: http://dx.doi.org/10.1145/2838931.2838932

Keywords

Semantic vector space representations, semantic consistency evaluation, sentence embeddings, word embeddings

1. INTRODUCTION

Sentence embeddings are often referred to as semantic vector space representations [7]. Embedding the meaning of a sentence into a vector space is expected to be very useful for natural language tasks. Vector representation of natural languages enables discourse analysis to take advantage of the array of tools available for computation in vector spaces. However, the embeddings of a sentence may encode a number of factors including semantic meaning, syntactic structure and topic. Since many of these embeddings are learned unsupervised on textual corpora using various models with different training objectives, it is not entirely clear the emphasis placed on each factor in the encoding. For applications where encoding semantic meaning is particularly desirable, such as machine translation and automatic summarization, it is crucial to be able to assess how well the embeddings capture the sentence's semantics. In other words, for successful application to these areas it is required that the embeddings generated by the models correctly encode meaning such that sentences with the same meaning are co-located in the vector space, and sentences with differing meanings are further away. However, few current models are directly trained to optimize for this criteria.

Currently sentence embeddings are often generated as a byproduct of unsupervised, or semi-supervised, tasks. These tasks include: word prediction [10]; recreation of input, as in the auto-encoders of [22, 19] and [7]; and syntactic structural classification [18, 21]. As a result the vector representations of the input sentences learned by these models are tuned towards the chosen optimization task. When employing the embeddings produced as features for other tasks, the information captured by the embeddings often proved to be very useful: e.g. approaching or exceeding previous state-of-the-art results, in sentiment analysis [22, 20, 10] and paraphrase detection [19]. However these practical applications do not directly show how well meaning is captured by the embeddings.

This paper provides a new method to assess how well the models are capturing semantic information. A strict definition for the semantic equivalence of sentences is: that each sentence shall entail the other. Such mutually entailing sen-

tences are called *paraphrases*. In this paper we propose to use paraphrases to assess how well the true semantic space aligns with the vector space the models embed into. It thus assesses whether projecting a sentence via the models in to the vector space preserves meaning.

The evaluation corpora were prepared by grouping paraphrases from the Microsoft Research Paraphrase (MSRP) [3] and Opinosis [5] corpora. A semantic classification task was defined which assesses if the model's embeddings could be used to correctly classify sentences as belonging to the paraphrase group with semantically equivalent sentences. Ensuring that sentences of common meaning, but differing form are located in vector space together, is a challenging task and shows a model's semantic encoding strength. This assessment, together with out recent work in the area, allows for a better understanding of how these models work, and suggest new directions for the development in this area.

The assessment proposed in this paper adds to the recent work on semantic evaluation methods, such as the work of Gershamn and Tenenbaum [6] and of Ritter et. al. [17]. In particular, the real-world corpus based assessment in this paper is highly complementary to the structured artificial corpus based assessment of Ritter et. al. These methods are discussed in more detail in the next section.

The rest of the paper is organized into the following sections. Section §2 discusses the existing models being assessed, the traditional assessment methods, and the aforementioned more recent semantic correctness based assessments. Section §3 describes the processes by which the models are evaluated using our new method, and the parameters used in the evaluation. Section §4 continues into more details on the development of the evaluation corpora for the semantic classification evaluation task. Section §5 details the results from evaluating the models and discusses the implications for their semantic consistency. Section §6 closes the paper and suggests new directions for development.

2. BACKGROUND

2.1 Models

Three well known sentence embedding methods are evaluated in this work. The compositional distributed model of the Unfoldering Recussive Autoencoder (URAE) by Socher et. al. [19]; and the two word content predictive models, Distributed Memory (PV-DM) and Distributed Bag of Words by Le and Mikolov [10]. In addition to these advanced sentence embedding models, a simple average of word embeddings, from Mikolov et. al. [13], is also assessed. These models and their variant forms have been applied to a number of natural language processing tasks in the past, as detailed in the subsequent sections, but not to a real-sentence semantic classification task as described here.

2.1.1 Unfolding Recursive Auto-Encoder (URAE)

The Unfolding Recursive Autoencoder (URAE) [19] is an autoencoder based method. It functions by recursively using a single layer feedforward neural-network to combine embedded representations, following the parse tree. Its optimization target is to be be able to reverse (unfold) the merges and produce the original sentence. The central folding layer – where the whole sentence is collapsed to a single embedding vector – is the sentence representation.

2.1.2 PV-DM

The Distributed Memory Paragraph Vectors (PV-DM) [10] method is based on an extension of the Continuous Bag-of-Words word-embedding model [12]. It is trained using a sliding window of words to predict the next word. The softmax predictor network is fed a word-embedding for each word in the window, plus an additional sentence embedding vector which is reused for all words in the sentence – called the paragraph vector in [10]. These input embeddings can be concatenated or averaged; in the results below they were concatenated. During training both word and sentence vectors are allowed to vary, in evaluation (i.e. inference), the word vectors are locked and the sentence vector is trained until convergence on the prediction task occurs.

2.1.3 PV-DBOW

Distributed Bag of Words Paragraph Vectors (PV-DBOW) [10], is based on the Skip-gram model for word-embeddings, also from [12]. In PV-DBOW a sentence vector is used as the sole input to a neural net. That network is tasked with predicting the words in the sentence. At each training iteration, the network is tasked to predict a number of words from the sentence, selected with a specified window size, using the sentence vector being trained as the input. As with PV-DM to infer embedding the rest of the network is locked, and only the sentence vector input allowed to vary, it is then trained to convergence.

2.1.4 Sum and Mean of Word Embeddings (SOWE and MOWE)

Taking the element-wise sum or mean of the word embeddings over all words in the sentence also produces a vector with the potential to encode meaning. Like traditional bag of words no order information is encoded, but the model can take into consideration word relations such as synonymity as encoded by the word vectors. The mean was used as baseline in [10]. The sum of word embeddings first considered in [13] for short phrases, it was found to be an effective model for summarization in [9]. The cosine distance, as is commonly used when comparing distances between embeddings, is invariant between sum and mean of word embeddings. Both sum and mean of word embeddings are computationally cheap models, particularly given pretrained word embeddings are available.

2.2 General Evaluation Methods

As discussed in the introduction, current methods of evaluating the quality of embedding are on direct practical applications designed down-stream.

Evaluation on a Paraphrase Detection task takes the form of being presented with pairs of sentences and tasked with determining if the sentences are paraphrases or not. The MSRP Corpus, [3] which we used in the semantic classification task, is intended for such use. This pairwise check is valuable, and does indicate to a certain extent if the embeddings are capturing meaning, or not. However, by considering groups of paraphrases, a deeper intuition can be gained on the arrangement of meaning within the vector space.

Sentiment Analysis is very commonly used task for evaluating embeddings. It was used both for the recursive autoencoder in [22] and for the paragraph vector models in [10]. Sentiment Analysis is classifying a text as positive or negative, or assigning a score as in the Sentiment Treebank

[23]. Determining the sentiment of a sentence is partially a semantic task, but it is lacking in several areas that would be required for meaning. For example, there is only an indirect requirement for the model to process the subject at all. Sentiment Analysis is a key task in natural language processing, but it is distinct from semantic meaning.

Document Classification is a classic natural language processing task. A particular case of this is topic categorization. Early work in the area goes back to [11] and [1]. Much more recently it has been used to assess the convolution neural networks of [25], where the articles of several news corpora were classified into categories such as "Sports", "Business" and "Entertainment". A huge spectrum of different sentences are assigned to the same topic. It is thus too board and insufficiently specific to evaluate the consistency of meanings. Information retrieval can be seen as the inverse of the document classification task.

Information Retrieval is the task of identifying the documents which most match a query. Such document selection depends almost entirely on topic matching. Suitable results for information retrieval have no requirement to agree on meaning, though text with the same meaning are more likely to match the same queries.

The evaluation of semantic consistency requires a task which is fine grained, and preserving meaning. Document Classification and Information Retrieval are insufficiently fine-grained. Sentiment Analysis does not preserve meaning, only semantic orientation. Paraphrase Detection is directly relevant to evaluating semantic constancy, however it is a binary choice based on a pairwise comparison – a more spatial application is desirable for evaluating these vector spaces. Thus the current down-steam application tasks are not sufficient for assessing semantic consistency – more specialized methods are required.

2.3 Evaluations of Semantic Consistency

Semantic consistency for word embeddings is often measured using the analogy task. In an analogy the meta-relation: `A is to B as C is to D`. Mikolov et. al.[14] demonstrated that the word-embedding models are semantically consistent by showing that the semantic relations between words were reflected as a linear offset in the vector space. That is to say, for embeddings $\vec{x}_a$, $\vec{x}_b$, $\vec{x}_c$, $\vec{x}_d$ corresponding to words A, B, C and D, respectively; it was tested that if for a strong relationship matching between A/B and C/D, then the offset vector would be approximately equal: $\vec{x}_b - \vec{x}_a \approxeq \vec{x}_d - \vec{x}_c$. Rearranging this in word space gets the often quoted example of $\texttt{King} - \texttt{Man} + \texttt{Woman} \approxeq \texttt{Queen}$, As man is to woman, king is to queen. In the rating task as described by [8], the goal is to rank such analogous word pairs based on the degree the relation matches. Thus to evaluate the word-embedding model using this task, it was a matter of sorting closeness of the corresponding offset vectors. Surprisingly strong results were found on this task[14]. It was thus demonstrated that word embeddings were not simply semantically consistent, but more so that this consistency was displayed as local linearity. This result gives confidence in the semantic quality of the word embeddings. However, this relationship analogy test cannot be performed for sentence embeddings.

Gershman et. al. [6], compares the distances of modified sentences in vector space, to the semantic distances ascribed to them by human raters. Like the analogy task for word vectors, this task requires ranking the targets based on the vector distance, however instead of rating on the strength of relationships it measures simply the similarities of the sentences to an original base sentence for each group. In that evaluation 30 simple base sentences of the form `A [adjective1] [noun1] [prepositional phrase] [adjective2] [noun2]` were modified to produce 4 difference derived sentences. The derived sentences were produced by swapping the nouns, swapping the adjectives, reversing the positional phrase (so `behind` becomes `in front of`), and a paraphrase by doing all of the aforementioned changes. Human raters were tasked with sorting the transformed sentences in similarity to the base sentence. This evaluation found that the embedding models considered did not agree with the semantic similarity rankings placed by humans. While the sentence embedding models performed poorly on the distance ranking measure, it is also worth considering how they perform on a meaning classification task.

A meaning classification task was recently proposed by Ritter et. al. [17], to classify sentences based on which spatial relationship was described. The task was to classify the sentence as describing: *Adhesion to Vertical Surface*, *Support by Horizontal Surface*, *Full Containment*, *Partial Containment*, or *Support from Above*. In this evaluation also, the sentences took a very structured form: `There is a [noun1] [on/in] the [noun2]`. These highly structured sentences take advantage of the disconnection between word content and the positional relationship described to form a task that must be solved by a compositional understanding combining the understanding of the words. *"The apple is on the refrigerator"* and *"The magnet is on the refrigerator"* belong to two separate spatial categories, even though the word content is very similar. Surprisingly, the simple model of adding word vectors outperformed compositional models such as the recursive autoencoder. The result does have some limitation due to the highly artificial nature of the sentences, and the restriction to categorizing into a small number of classes based only on the meaning in terms of positional relationship. To generalize this task, in this paper we consider real world sentences being classed into groups according to their full semantic meaning.

3. METHODOLOGY

To evaluate how well a model's vectors capture the meaning of a sentence, a semantic classification task was defined. The task is to classify sentences into classes where each shares the same meaning. Each class is thus defined as a paraphrase groups. This is a far finer-grained task than topic classification. It is a multiclass classification problem, rather than the binary decision problem of paraphrase detection. Such multiclass classification requires the paraphrase groups to be projected into compact and distinct groups in the vector space. A model which produces such embeddings which are thus easily classifiable according to their meaning can been thus seen to have good semantic localization.

This semantic classification does not have direct practical application – it is rare that the need will be to quantify sentences into groups with the same prior known meaning. Rather it serves as a measure to assess the models general suitability for other tasks requiring a model with consistency between meaning and embedding.

To evaluate the success at the task three main processes are involved, as shown in Figure 1: Corpus Preparation,

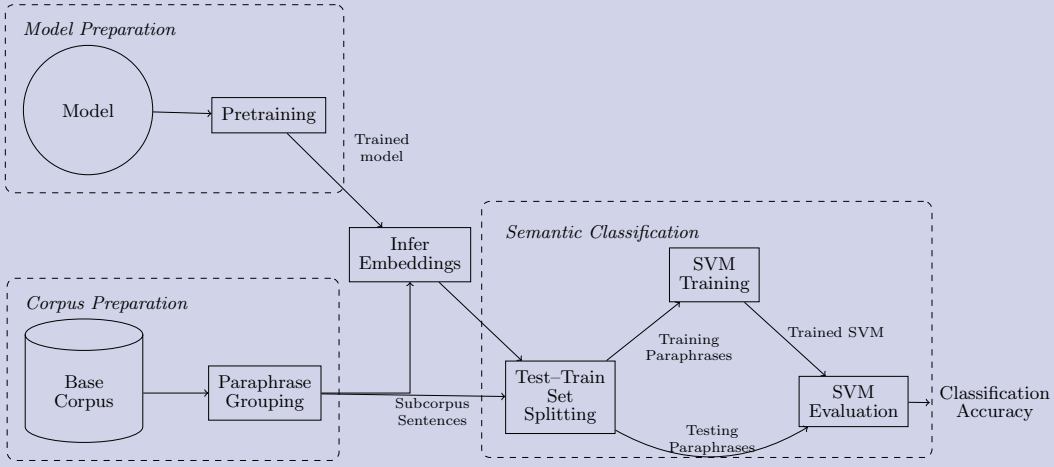


Figure 1: Process Diagram for the Evaluation of Semantic Consistency via our method

Model Preparation, and the Semantic Classification task itself.

3.1 Corpus Preparation

The construction of each of the corpora is detailed more fully in the next section. In brief: Two corpora were constructed by selecting subsets of the Microsoft Research Paraphrase (MSRP) [3] and of the Opinosis [5] corpora. The corpora were partitioned into groups of paraphrases – sentences with the same meaning. Any paraphrase groups with less than three sentences were discarded. The paraphrase grouping was carried out manually for Opinosis, and automatically for the MSRP corpus using the existing paraphrase pairings. The paraphrase groups divide the total semantic space of the corpora into discrete classes, where each class contains sentences sharing the same meaning.

It is by comparing the ability of the models to produce embeddings which can be classified back into these classes, that we can compare the real semantic space partitions to their corresponding vector embedding space regions.

3.2 Model Preparation and Inferring Vectors

Prior to application to semantic classification, as with any task the models had to be pretrained. Here we use the term *pretraining* to differentiate the model training from the classifier training. The pretraining is not done using the evaluation corpora as they are both very small. Instead other data are used, and the inference/evaluation procedure given for each method was then used to produce the vectors for each sentence. The model parameters used are detailed below.

3.2.1 Unfolding Recursive Auto-Encoder (URAE)

In this evaluation we make use of the pretrained network that Socher et. al. have graciously made available[1], full information is available in the paper[19]. It is initialized on the unsupervised Collobert and Weston word embeddings[2], and training on a subset of 150,000 sentences from the gigaword corpus. It produces embeddings with 200 dimensions. This pretrained model when used with dynamic pooling and other word based features performed very well on the MSRP corpus paraphrase detection. However in the evaluation below the dynamic pooling techniques are not used as they are only directly suitable for enhancing pairwise comparisons between sentences.

3.2.2 Paragraph Vector Methods (PV-DM and PV-DBOW)

Both PV-DM and PV-DBOW, were evaluated using the GenSim implementation [16] from the current *develop* branch[2]. Both were trained on approximately 1.2 million sentences from randomly selected Wikipedia articles, and the window size was set to 8 words, and the vectors were of 300 dimensions.

3.2.3 Sum and Mean of Word Embeddings (SOWE and MOWE)

The word embeddings used for MOWE were taken from the Google News pretrained model[3] based on the method described in [13]. This has been trained on 100 million sentences from Google News. A small portion of the evaluation corpus did not have embeddings in the Google News model. These tokens were largely numerals, punctuation symbols, proper nouns and unusual spellings, as well as the stop-words: “and”, “a” and “of”. These words were simply skipped. The resulting embeddings have 300 dimensions, like the word embeddings they were based on.

[1] http://www.socher.org/index.php/Main/DynamicPoolingAndUnfoldingRecursiveAutoencodersForParaphraseDetection

[2] https://github.com/piskvorky/gensim/tree/develop/

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

3.2.4 Bag of Words (BOW and PCA BOW)

A bag of words (BOW) model is also presented as a baseline. There is a dimension in each vector embedding for the count of each token, including punctuation, in the sentence. In the Opinosis and MSRP subcorpora there were a total of 1,085 and 2,976 unique tokens respectively, leading to BOW embeddings of corresponding dimensionality. As it is a distributional rather than distributed representation, the BOW model does not need any pretraining step. For comparison to the lower dimensional models Principle Component Analysis (PCA) was applied to the BOW embeddings to produce an additional baseline set of embeddings of 300 dimensions – in line with PV-DM, PV-DBOW, SOWE, and MOWE models. It does not quite follow the steps shown in Figure 1, as the PCA pretraining step is performed on the training embeddings only during the SVM classification process, and it is used to infer the PCA BOW embeddings during the testing step. This avoids unfair information transfer where the PCA would otherwise be about to choose representations optimized for the whole set, including the test data. It was found that when the PCA model was allowed to cheat in this way it performed a few percentage points better. The bag of words models do not have any outside knowledge.

3.3 Semantic Classification

The core of this evaluation procedure is in the semantic classification step. A support vector machine (SVM), with a linear kernel, and class weighting was applied to the task of predicting which paraphrase group each sentence belongs to. Classification was verified using 3-fold cross-validation across different splits of the testing/training data, the average results are shown in this section. The splits were in proportion to the class size. For the smallest groups this means there were two training cases and one test case to classify.

In this paper, only a linear kernel was used, because a more powerful kernel such as RBF may be able to compensate for irregularities in the vector space, which makes model comparison more difficult. Scikit-learn [15] was used to orchestrate the cross-validation and to interface with the LibLinear SVM implementation [4]. As the linear SVM's classification success depends on how linearly separable the input data is, thus this assessed the quality of the localization of the paraphrase groupings embeddings.

4. CORPUS CONSTRUCTION

4.1 Microsoft Research Paraphrased Grouped Subcorpus

The MSRP corpus is a very well established data set for the paraphrase detection task [3]. Sentences are presented as pairs which are either paraphrases, or not. A significant number of paraphrases appear in multiple different pairings. Using this information, groups of paraphrases can be formed.

The corpus was partitioned according to sentence meaning by taking the symmetric and transitive closures the set of paraphrase pairs. For example if sentences A, B, C and D were present in the original corpus as paraphrase pairs: A, B, D, A and B, C then the paraphrase group $\{A, B, C, D\}$ is found. Any paraphrase groups containing less than 3 phrases were discarded. The resulting sub-corpus has the breakdown as shown in Figure 2.

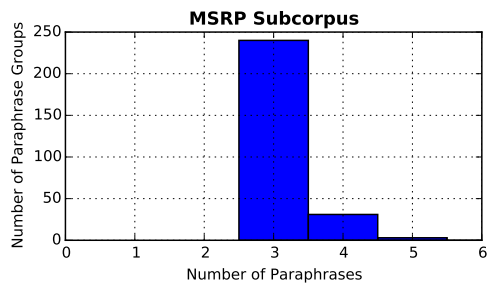


Figure 2: Break down of how many paraphrases groups are present in the MSRP subcorpus of which sizes.It contains a total of 859 unique sentences, broken up into 273 paraphrase groups.

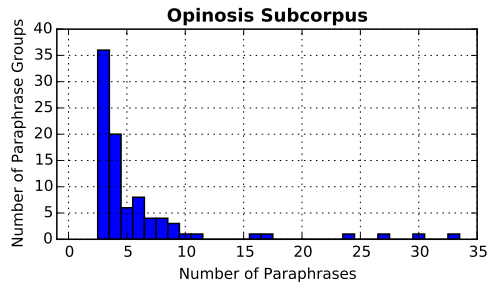


Figure 3: Break down of how many paraphrases groups are present in the Opinosis subcorpus of which sizes. It contains a total of 521 unique sentences, broken up into 89 paraphrase groups.

4.2 Opinosis Paraphrase Grouped Subcorpus

The Opinosis Corpus[5] was used as secondary source of original real-world text. It is sourced from several online review sites: Tripadvisor, Edmunds.com, and Amazon.com, and contains single sentence statements about hotels, cars and electronics. The advantage of this as a source for texts is that comments on the quality of services and products tend to be along similar lines. The review sentences are syntactically simpler than sentences from a news-wire corpus, and also contain less named entities. However, as they are from more casual communications, the adherence to grammar and spelling may be less formal.

Paraphrases were identified using the standard criterion: bidirectional entailment. For a paraphrase group $\mathcal{S}$ of sentences: $\forall s_1, s_2 \in \mathcal{S}, \quad s_1 \vDash s_2 \quad \wedge \; s_2 \vDash s_1$, every sentence in the group entails the every other sentence in the group. A stricter interpretation of bidirectional entailment was used, as compared to the "mostly bidirectional entailment" used in the MSRP corpus. The grouping was carried out manually. Where it was unclear as to the group a particular phrase should belong to it was left out of the corpus entirely. The general guidelines were as follows.

	MSRP Subcorpus	Opinosis Subcorpus
PV-DM	78.00%	38.26%
PV-DBOW	89.93%	32.19%
URAE	51.14%	20.86%
MOWE	97.91%	**69.30%**
SOWE	98.02%	68.75%
BOW	**98.37%**	65.23%
PCA BOW	97.96%	54.43%

Table 1: The semantic classification accuracy of the various models across the two evaluation corpora.

- Tense, Transitional Phrases, and Discourse and Pragmatic Markers were ignored.
- Statement intensity was coarsely quantized.
- Approximately equal quantitative and qualitative values were treated as synonymous.
- Sentences with entities mentioned explicitly were grouped separately from similar statements where they were implied.
- Sentences with additional information were grouped separately from those without that information.

The final point is the most significant change from the practices apparent in the construction of the MSRP corpus. Sentences with differing or additional information were classified as non-paraphrases. This requirement comes from the definition of bidirectional entailment. For example, *"The staff were friendly and polite."*, *"The staff were polite."* and *"The staff were friendly."* are in three separate paraphrase groups. The creators of the MSRP corpus, however, note "...the majority of the equivalent pairs in this dataset exhibit 'mostly bidirectional entailments', with one sentence containing information 'that differs' from or is not contained in the other." [3]. While this does lead to more varied paraphrases; it strays from the strict linguistic definition of a paraphrase, which complicates the evaluation of the semantic space attempted here. This stricter adherence to bidirectional entailment resulted in finer separation of groups, which makes this a more challenging corpus.

After the corpus had been broken into paraphrase groups some simple post-processing was done. Several artifacts present in the original corpus were removed, such as substituting the ampersand symbol for `&amp`. Any paraphrase groups containing identical sentences were merged, and duplicates removed. Finally, any group with less than three phrases was discarded. With this complete the breakdown is as in Figure 3.

Further information on the construction of the corpora in this section, and download links are available online.[4]

5. RESULTS AND DISCUSSION

5.1 Classification Results and Discussion

The results of performing the evaluation method described in Section §3 are shown in Table 1.

[4]http://white.ucc.asn.au/resources/paraphrase_grouped_corpora/

While the relative performance of the models is similar between the corpora, the absolute performance differs. On the absolute scale, all the models perform much better on the MSRP subcorpus than on the Opinosis subcorpus. This can be attributed to the significantly more distinct classes in the MSRP subcorpus. The Opinosis subcorpus draws a finer line between sentences with similar meanings. As discussed earlier, for example there is a paraphrase group for *"The staff were polite."*, another for *"The staff were friendly."*, and a third for *"The staff were friendly and polite."*. Under the guidelines used for paraphrases in MSRP, these would all have been considered the same group. Secondly, there is a much wider range of topics in the MSRP. Thus the paraphrase groups with different meanings in MSRP corpus are also more likely to have different topic entirely than those from Opinosis. Thus the the ground truth of the semantics separability of phrases from the MSRP corpus is higher than for Opinosis, making the semantic classification of the Opinosis subcorpus is a more challenging task.

The URAE model performs the worst of all models evaluated. In [9] is was suggested that the URAE's poor performance at summarizing the Opinosis corpus could potentially be attributed to the less formally structured product reviews – the URAE being a highly structured compositional model. However, here it also performed poorly on the MSRP – which it was created for [19]. The exact same model from [19] was used here – though this did put it at a dimensional disadvantage over the other models having 200 dimensions to the other's 300. The key difference from [19], beyond the changing to a multiclass classification problem, was the lack of the complementary word-level features as used in the dynamic pooling layer. This suggests the model could benefit from such world level features – as the very strong performance of the word-based model indicates.

The word based models, MOWE, SOWE, BOW and PCA BOW, performed very well. This suggests that word choice is a very significant factor in determining meaning; so much so that the models which can make use of word order information, URAE and PV-DM, were significantly outperformed by methods which made more direct use of the word content.

The very high performance of the BOW maybe attributed to its very high dimensionality, though the MOWE and SOWE performed similarly. The PCA step can be considered as being similar to choosing an optimal set of words to keep so as to maximum variability in the bag of words. It loses little performance, even though decreasing vector size by an order of magnitude – particularly on the easier MSRP dataset.

5.2 Model Agreement

The misclassifications of the models can be compared. By selecting one of the test/train folds from the classification task above, and comparing the predicted classifications for each test-set sentence, the similarities of the models were assessed. The heatmaps in Figure 4 show the agreement in errors. Here misclassification agreement is given as an approximation to $P(m_1(x) = m_2(x) \,|\, m_1(x) \neq y \wedge m_2(x) \neq y)$, for a randomly selected sentence x, with ground truth classification y, where the models m_1 and m_2 are used to produce classifications. Only considering the cases where both models were incorrect, rather than simple agreement, avoids the analysis being entirely dominated by the agreement of the

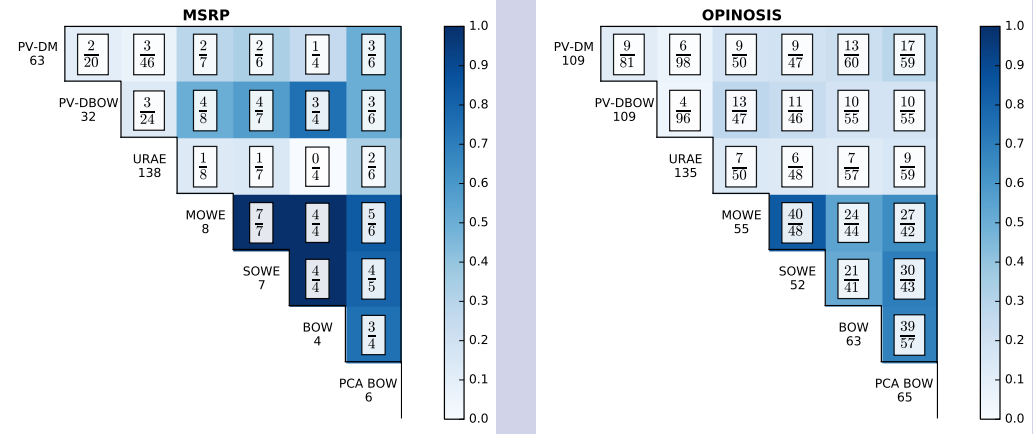


Figure 4: The misclassification agreement between each of the models for the MSRP (left) and Opinosis (right) subcorpora. Below each model name is the total mistakes made. The denominator of each fraction is the number of test cases incorrectly classified by both models. The numerator is the portion of those misclassifications which were classified in the same (incorrect) way by both models. The shading is in-proportion to that fraction.

models with the ground truth.

The word based models showed significant agreement. Unsurprisingly MOWE and SOWE have almost complete agreement in both evaluations. The other models showed less agreement – while they got many of the same cases wrong the models produced different misclassifications. This overall suggests that the various full sentence models are producing substantially dissimilar maps from meaning to vector space. Thus it seems reasonable that using a ensemble approach between multiple sentence models and one word-based model would produce strong results. Yin and Schütze [24] found this successful when combining different word embedding models.

5.3 Limitations

This evaluation has some limitations. As with all such empirical evaluations of machine learning models, a more optimal choice of hyper-parameters and training data will have an impact on the performance. In particular, if the model training was on the evaluation data the models would be expected to be better able to position their embedding. This was however unfeasible due to the small sizes of the datasets used for evaluation, and would not reflect real word application of the models to data not prior seen. Beyond the limitation of the use of the datasets is their contents.

The paraphrase groups were not selected to be independent of the word content overlap – they were simply collected on commonality of meaning from real world sourced corpora. This is a distinct contrast to the the work of Ritter et. al.[17] discussed in section 2.3 where the classes were chosen to not have meaningful word overlap. However our work is complementary to theirs, and our findings are well aligned. The key difference in performance is the magnitude of the performance of the sum of word embeddings (comparable to the mean of word embeddings evaluated here). In [17] the word embedding model performed similarly to the best of the more complex models. In the results presented above we find that the word embedding based model performs significantly beyond the more complex models. This can be attributed to the word overlap in the paraphrase groups – in real-world speech people trying to say the same thing do in-fact use the same words very often.

6. CONCLUSION

A method was presented, to evaluate the semantic localization of sentence embedding models. Semantically equivalent sentences are those which exhibit bidirectional entailment – they each imply the truth of the other. Paraphrases are semantically equivalent. The evaluation method is a semantic classification task – to classify sentences as belonging to a paraphrase group of semantically equivalent sentences. The datasets used were derived from subsets of existing sources, the MRSP and the Opinosis corpora. The relative performance of various models was consistent across the two tasks, though differed on an absolute scale.

The word embedding and bag of word models performed best, followed by the paragraph vector models, with the URAE trailing in both tests. The strong performance of the sum and mean of word embeddings (SOWE and MOWE) compared to the more advanced models aligned with the results of Ritter et. al.[17]. The difference in performance presented here for real-word sentences, were more marked than for the synthetic sentence used by Ritter et. al. This may be attributed to real-world sentences often having meaning overlap correspondent to word overlap – as seen also in the very strong performance of bag of words. Combining the result of this work with those of Ritter et. al., it can be concluded that summing word vector representations is a practical and surprisingly effective method for encoding the meaning of a sentence.

Acknowledgement

This research is supported by the Australian Postgraduate Award, and partially funded by Australian Research Council DP150102405 and LP110100050.

7. REFERENCES

[1] H. Borko and M. Bernick. Automatic document classification. *Journal of the ACM (JACM)*, 10(2):151–162, 1963.

[2] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[3] W. B. Dolan and C. Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*. Asia Federation of Natural Language Processing, 2005.

[4] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[5] K. Ganesan, C. Zhai, and J. Han. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 340–348. Association for Computational Linguistics, 2010.

[6] S. J. Gershman and J. B. Tenenbaum. Phrase similarity in humans and machines. *Proceedings of the 37th Annual Conference of the Cognitive Science Society*, 2015.

[7] M. Iyyer, J. Boyd-Graber, and H. D. III. Generating sentences from semantic vector space representations. In *NIPS Workshop on Learning Semantics*, 2014.

[8] D. A. Jurgens, P. D. Turney, S. M. Mohammad, and K. J. Holyoak. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 356–364. Association for Computational Linguistics, 2012.

[9] M. Kågebäck, O. Mogren, N. Tahmasebi, and D. Dubhashi. Extractive summarization using continuous vector space models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)@ EACL*, pages 31–39, 2014.

[10] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014.

[11] M. E. Maron. Automatic indexing: an experimental inquiry. *Journal of the ACM (JACM)*, 8(3):404–417, 1961.

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

[13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

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

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[16] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

[17] 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.

[18] R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng. Parsing with compositional vector grammars. In *ACL*. 2013.

[19] 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.

[20] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics, 2012.

[21] R. Socher, C. D. Manning, and A. Y. Ng. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pages 1–9, 2010.

[22] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2011.

[23] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer, 2013.

[24] W. Yin and H. Schütze. Learning word meta-embeddings by using ensembles of embedding sets. Aug. 2015.

[25] X. Zhang and Y. LeCun. Text understanding from scratch. *CoRR*, Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, 2015.

Learning of Colors from Color Names: Distribution and Point Estimation

Lyndon White*
The University of Western Australia

Roberto Togneri**
The University of Western Australia

Wei Liu†
The University of Western Australia

Mohammed Bennamoun‡
The University of Western Australia§

Color names are often made up of multiple words, as a task in natural language understanding we investigate in depth the capacity of neural networks based on sums of word embeddings (SOWE), recurrence (RNN) and convolution (CNN), to estimate colors from sequences of terms. We consider both point as well as distribution estimates of color. We argue that the latter has a particular value as there is no clear agreement between people as to what a particular color describes – different people have a different idea of what it means to be "very dark orange". Surprisingly, the sum of word embeddings generally performs the best on almost all evaluations.

1. Introduction

Consider that the word `tan` may mean one of many colors for different people in different circumstances: ranging from the bronze of a tanned sunbather, to the brown of tanned leather; `green` may mean anything from `aquamarine` to `forest green`; and even `forest green` may mean the rich shades of a rain-forest, or the near grey of Australian bushland. Thus the *color intended* cannot be uniquely inferred from a color name. Without further context, it does nevertheless remain possible to estimate likelihoods of which colors are intended based on the population's use of the words.

Color understanding, that is, generating color from text, is an important subtask in natural language understanding, indispensable for *executable semantic parsing*. For example, in a natural language enabled human-machine interface, when asked to select

* E-mail: lyndon.white@research.uwa.edu.au
** Email: roberto.togneri@uwa.edu.au
† Email: wei.liu@uwa.edu.au
‡ Email: mohammed.bennamoun@uwa.edu.au
§ The University of Western Australia, 35 Stirling Highway, Crawley, Western Australia.

the `dark bluish green` object, it would be much useful if we could rank each object based on how likely its color matches against a learned distribution of the color name `dark bluish green`. This way if the most-likely object is eliminated (via another factor), the second most likely one can be considered. A threshold can be set to terminate the search. This kind of likelihood based approach is not possible when we have only exact semantics based on point estimates.

Color understanding is a challenging domain, due to high levels of ambiguity, the multiple roles taken by the same words, the many modifiers, and the many shades of meaning. In many ways it is a grounded microcosm of natural language understanding. Due to its difficulty, texts containing color descriptions such as `the flower has petals that are bright pinkish purple with white stigma` are used to demonstrate the capability of the-state-of-the-art image generation systems (Reed et al. 2016; Mansimov et al. 2015). The core focus of the work we present here is to address these linguistic phenomena around the short-phrase descriptions of a color, which can be represented in a color-space such as HSV (Smith 1978). Issues of illumination and perceived color based on context are considered out of the scope.

1.1 Distribution vs. Point Estimation

As illustrated, proper understanding of color names requires considering *the color intended* as a random variable. In other words, a color name should map to a distribution, not just a single point or region. For a given color name, any number of points in the color-space could be intended, with some being more or less likely than others. Or equivalently, up to interpretation, it may intend a region but the likelihood of what points are covered is variable and uncertain. This distribution is often multimodal and has a high and asymmetrical variance, which further renders regression to a single point unsuitable.

Having said that, we do produce point estimate results for completeness, though we argue the true usefulness of such estimates is limited.

A single point estimate, does not capture the diverse nature of the color names adequately. Moreover, it is impossible to find the single best point estimation method. For example: for a bimodal distribution, using the distribution mean as a point estimate will select a point in the valley between the peaks, which is less likely. Similarly for an asymmetrical distribution, where the mean will be off to one side of the peak. Conversely, using the distribution mode, will select the highest (most likely) peaks, but will on average be more incorrect (as measured by the mean squared error). The correct trade-off, if a point estimate is required, is dependent on the final use of the system. Another problem is that point estimates do not capture the sensitivity. In an asymmetrical distribution, having a point slightly off-centre in one direction may result in a very different probability, this more generally holds for a narrow variance distribution. Conversely for a very wide variance distribution (for example one approaching the uniform distribution) the point estimate value may matter very little with all points providing similar probabilities. Color distributions are almost always multimodal or asymmetrical, and exhibit widely differing variances for different names (this can be seen in the histograms of the training data shown in Section 6.1). While by definition the mean color point minimizes the squared error, it may not actually be a meaningful point estimate. Given these issues, producing a point estimate has only limited value and estimating a distribution is more general task. However we do consider the point estimation task, as it allows contrast in assessing the input module (SOWE/CNN/RNN) of our proposed methods across the two different output modules (distribution/point estimation).

The generation of color from text has not received much attention in prior work. To the best of our knowledge, the only similar work is Kawakami et al. (2016); which only considers point estimation, and uses a dataset containing far too few observations to allow for learning probability distributions from population usages of the color names. To our knowledge The generation of probability distributions in color-space based on sequences of terms making up the color name, has not been considered at all by any prior work. There has been several works on the reverse problem (McMahan and Stone 2015; Meo, McMahan, and Stone 2014; Monroe, Goodman, and Potts 2016): the generation of a textual name for a color from a point in a color-space. From the set of work on the reverse problem there is a clear trend on data-driven approaches in recent years where more color names and observations are used.

1.2 Contributions

Problem statement: given a set of $\langle$color-name, $(h, s, v)\rangle$ pairs, we need to learn a mapping from any color-name, seen or unseen to a color-value or a distribution in HSV space.

We propose a neural network based architecture that can be broken down into an **input module**, which learns a vector representation of color-names, and a connected **output module**, which produces either a probability distribution or a point estimate. The **output module** uses a softmax output layer for probability distribution estimation, or a novel HSV output layer for point estimation. To carry out the representational learning of color-names, three different color-name embedding learning models are investigated for use in the **input module**: Sum Of Word Embeddings (SOWE), Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN). The capacity of these input models is of primary interest to this work.

To evaluate and compare the three learning models, we designed a series of experiments to assess their capability in capturing compositionality of language used in color names. These include: **(1)** evaluation on all color names (full task); **(2)** evaluation on color names when the order of the words matters (order task); **(3)** evaluation on color names which never occur in the training data in that exact form, but for which all terms occur in the training data (unseen combination task); **(4)** qualitative demonstration of outputs for color names with terms which do not occur in the training data at all, but for which we know their word embeddings (embedding only task).

An interesting challenge when considering this discretization is the smoothness of the estimate. The true space is continuous, even if we are discretizing it at a resolution as high as the original color displays which were used to collect the data. Being continuous means that a small change in the point location in the color-space should correspond to a small change in how likely that point is according to the probability distribution. Informally, this means the histograms should look smooth, and not spiky. We investigated using a Kernel Density Estimation (KDE) based method for smoothing the training data, and further we conclude that the neural networks learn this smoothness. To qualify our estimate of the distribution, we discretize the HSV color-space to produce a histogram. This allows us to take advantage of the well-known softmax based methods for the estimation of a probability mass distribution using a neural network.

We conclude that the SOWE model is generally the best model for all tasks both for distribution and point estimation. It is followed closely by the CNN; with the RNN performing significantly worse (see Section 6). We believe that due to the nature of color understanding as a microcosm of natural language understanding, the results of our investigations have some implications for the capacity of the models for their general use in short phrase understanding.

To the best of knowledge, this is the first of such investigation in term-wise mapping color names to values that can generate a visual representation, which bring the future of natural language controlled computer graphics generation and executable semantic parsing one step closer to reality.

2. Related Work

The understanding of color names has long been a concern of psycholinguistics and anthropologists (Berlin and Kay 1969; Heider 1972; Heider and Olivier 1972; Mylonas et al. 2015). It is thus no surprise that there should be a corresponding field of research in natural language processing.

The earliest works revolve around explicit color dictionaries. This includes the ISCC-NBS color system (Kelly et al. 1955) of 26 words, that are composed according to a context free grammar, such that phrases are mapped to single points in the color-space; and the simpler, non-compositional, 11 basic colors of Berlin and Kay (1969). Works including Berk, Kaufman, and Brownston (1982); Conway (1992); Lammens (1994); Mojsilovic (2005); Menegaz et al. (2007); Van De Weijer et al. (2009) which propose methods for the automatic mapping of colors to and from these small manually defined sets of colors. We note that Menegaz et al. (2007); Van De Weijer et al. (2009) both propose systems that discretize the color-space, though to a much coarser level than we consider in this work.

More recent works, including the work presented in this article, function with a much larger number of colors, larger vocabularies, and larger pools of respondents. In particular making uses of the large Munroe dataset Munroe (2010), as we do here. This allows a data driven approach towards the modelling.

McMahan and Stone (2015) and Meo, McMahan, and Stone (2014) present color naming methods. Their work primarily focuses on mapping from colors to to their exact names, the reverse of our task. These works are based on defining fuzzy rectangular

distributions in the color-space to cover the distribution estimated from the data, which are used in a Bayesian system to non-compositionally determine the color name.

Monroe, Goodman, and Potts (2016) map a point in the color-space, to a sequence of probability estimates over color words. They extend beyond, all prior color naming systems to produce a compositional color namer based on the Munroe dataset. Their method uses a recurrent neural network (RNN), which takes as input a color-space point, and the previous output word, and gives a probability of the next word to output – this is a conditional language model. We tackle the inverse problem to the creation of a conditional language model. Our distribution estimation models map from a sequence of terms, to a distribution in color-space. Similarly, our point estimation models map from a sequence of terms to a single point in color-space.

Kawakami et al. (2016) proposes another compositional color naming model. They use a per-character RNN and a variational autoencoder approach. It is in principle very similar to Monroe, Goodman, and Potts (2016), but functioning on a character, rather than a word level. The work by Kawakami et al. also includes a method for generating colors. However they only consider the generation of point estimated, rather than distributions. The primary focus of our work is on generating distributions. The datasets used by Kawakami et al. contain only very small numbers of observations for each color name (often just one). These datasets are thus not suitable for modelling the distribution in color-space as interpreted by a population. Further, given the very small number of examples they are not well suited for use with word-based modelling: the character based modelling employed by Kawakami et al. is much more suitable. As such, we do not attempt to compare with their work.

Monroe et al. (2017) present a neural network solution to a communication game, where a speaker is presented with three colors and asked to describe one of them, and

the listener is to work out which color is being described. Speaker and listener models are trained, using LSTM-based decoders and encoders, respectively. The final time-step of their model produces a 100 dimensional representation of the description provided. From this, a Gaussian distributed score function is calculated, over a high dimensional color-space from Monroe, Goodman, and Potts (2016), which is then used to score each of the three options. While this method does work with a probability distribution, as a step in its goal, this distribution is always both symmetric and unimodal – albeit in a high-dimensional color-space.

Winn and Muresan (2018) demonstrates a neural network for producing point estimates of how colors change based on the use of comparative terms such as `lighter`. The network's input is word embedding for a comparative adjective, and a point in RGB color-space; the model outputs a point in predicted RGB space that is the modified version of the input color point according to the given adjective. For example mapping from green to a darker green is: $((164, 227, 77), \text{darker}) \mapsto (141, 190, 61)$. The color adjectives may have up to two words, to allow for expressions such as `more neon`. This is allowed by taking as the fixed sized input two embeddings – when only one input is required, the other is replaced by zero vector. Their training and evaluation is based on data sourced from the Munroe dataset.

The work presented here closes the gap, that while we have language models conditional upon color, we do not have color models conditional upon language.

3. Method

3.1 System Architecture

Our overall system architecture for all models is shown in Figure 1. This shows how color names are transformed into distribution or point estimates over the HSV color-space.

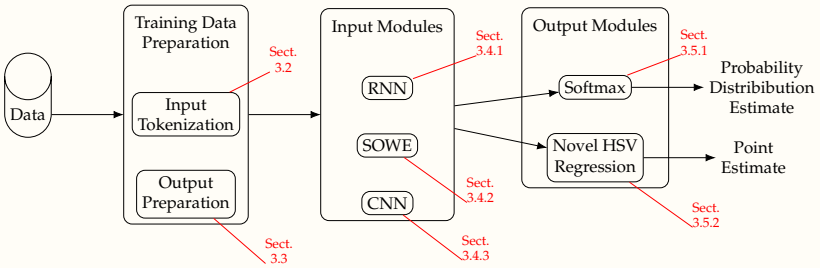


Figure 1
The overall architecture of our systems

3.2 Input Data Preparation

We desire a color prediction model which takes as input a sequence of words that make up the color's name; rather than simply mapping from the whole phrase (whole phrase mapping does not scale to new user input, given the combinatorial nature of language). Towards this end, color names are first tokenized into individual words. For the input into our neural network based models, these words are represented with pretrained word embeddings.

3.2.1 Tokenization. During tokenization a color name is split into terms with consistent spelling. For example, `bluish kahki` would become the sequence of 3 tokens: [`blue`, `ish`, `khaki`]. Other than spelling, the tokenization results in the splitting of affixes and combining tokens (such as hyphens). Combining tokens and related affixes affect how multiple colors can be combined. The full list of tokenization rules can be found in the accompanying source code. Some further examples showing how combining tokens and affixes are used and tokenized:

- `blue purple` $\mapsto$ [`blue`, `purple`].
- `blue-purple` $\mapsto$ [`blue`, `-`, `purple`].

- `bluish purple` ↦ [`blue`, `ish`, `purple`]
- `bluy purple` ↦ [`blue`, `y`, `purple`]
- `blurple` ↦ [`blue-purple`]

The final example of `blurple` is a special-case. It is the only portmanteau in the dataset, and we do not have a clear way to tokenize it into a series of terms which occur in our pretrained embedding's vocabulary (see Section 3.2.2). The portmanteau `blurple` is not in common use in any training set used for creating word embeddings, so no pretrained embedding is available.[1] As such we handle it by treating it as the single token `blue-purple` for purposes of finding an embedding. There are many similar hyphenated tokens in the pretrained embeddings vocabulary, however (with that exception) we do not use them as it reduces the sequential modelling task to the point of being uninteresting.

3.2.2 Embeddings. All our neural network based solutions incorporate an embedding layer. This embedding layer maps from tokenized words to vectors. We make use of 300 dimensional pretrained FastText embeddings (Bojanowski et al. 2017)[2].

The embeddings are not trained during the task, but are kept fixed. As per the universal approximation theorem (Leshno et al. 1993; Sonoda and Murata 2017) the layers above the embedding layer allow for arbitrary continuous transformations. By fixing the embeddings, and learning this transformation, we can produce estimates of colors from embeddings alone, without any training data at all, as shown in Section 6.4.

1 Methods do exist to generate embeddings for out of vocabulary words (like `blurple`), particularly with FastText embeddings (Bojanowski et al. 2017). But we do not investigate those here.
2 Available from `https://fasttext.cc/docs/en/english-vectors.html`

3.3 Output Data Preparation for Training Distribution Estimation Models

To train the distribution estimation models we need to preprocess the training data into a distribution. The raw training data itself, is just a collection of $\langle$color-name, $(h, s, v)\rangle$ pairs – samples from the distributions for each named-color. This is suitable for training the point estimation models, but not for the distribution estimation models . We thus convert it into a tractable form, of one histogram per output channel – by assuming the output channels are conditionality independent of each other.

3.3.1 Conditional Independence Assumption. We make the assumption that given the name of the color, the distribution of the hue, saturation and value channels are independent. That is to say, it is assumed if the color name is known, then knowing the value of one channel would not provide any additional information as to the value of the other two channels. The same assumption is made, though not remarked upon, in Meo, McMahan, and Stone (2014) and McMahan and Stone (2015). This assumption of conditional independence allows considerable saving in computational resources. Approximating the 3D joint distribution as the product of three 1D distributions decreases the space complexity from $O(n^3)$ to $O(n)$ in the discretized step that follows.

Superficial checks were carried out on the accuracy of this assumption. Spearman's correlation on the training data suggests that for over three quarters of all color names, there is only weak correlation between the channels for most colors (Q3 = 0.187). However, this measure underestimates correlation for values which have a circular relative value, such as hue. Of the 16 color-spaces evaluated, HSV had the lowest correlation by a large margin. Full details, including the table of correlations, are available in supplementary materials (Appendix 1). These results are suggestive, rather than solidly indicative, on the degree of correctness of the conditional independence assumption. We

consider the assumption sufficient for this investigation; as it does not impact on the correctness of results. A method that does not make this assumption may perform better when evaluated using the same metrics we use here.

3.3.2 Discretization. For distribution estimation, our models are trained to output histograms. This is a discretized representation of the continuous distribution. Following standard practice, interpolation-based methods can be used to handle it as a continuous distribution. By making use of the conditional independence assumption (see Section 3.3.1), we output one 256-bin histogram per channel. We note that 24-bit color (as was used in the survey that collected the dataset) can have all the information captured by a 256 bin discretization per channel. 24 bit color allows for a total of 2^{24} colors to be represented, and even one-hot encoding for each of the 256 bin discretization channels allows for the same. As such there is no meaningful loss of information when using histograms over a truely continuous estimation method, such as a Gaussian mixture model. Although such models may have other advantages (such as the a priori information added by specifying the distribution), we do not investigate them here, instead considering the simplest non-parametric estimation model (the histogram), which has the simple implementation in a neural network using a softmax output layer.

Discretizing the data in this way is a useful solution used in several other machine learning systems. Oord, Kalchbrenner, and Kavukcuoglu (2016); van den Oord et al. (2016) apply a similar discretization step and found that their method to outperform the more complex truly continuous distribution outputting systems.

For training purposes we thus convert all the observations into histograms. One set of training histograms is produced per color description present in the dataset – that is to say a training histogram is create for `brownish green`, `greenish brown`, `green` etc. We perform uniform weight attribution of points to bins as described by Jones and

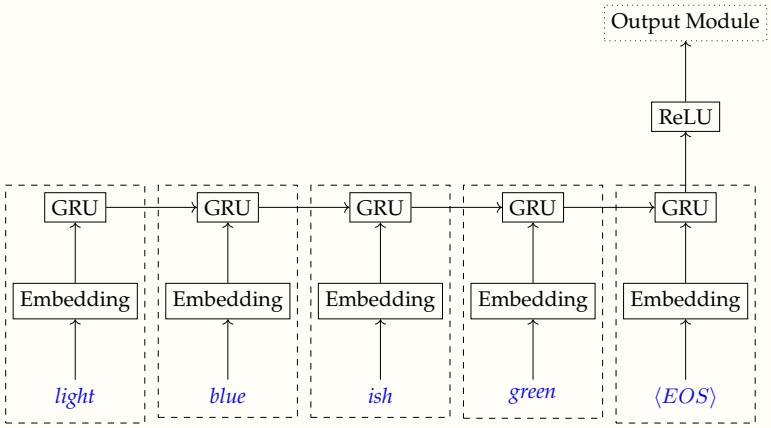


Figure 2
The RNN Input module for the example input `light greenish blue`. Each dashed box represents 1 time-step.

Lotwick (1984). In-short, this method of tabulation is to define the bins by their midpoints, and to allocate probability mass to each bin based on how close an observe point is to the adjacent midpoints. A point precisely on a midpoint would have all its probability mass allocated to that bin; whereas a point halfway between midpoints would have 50% of its mass allocated to each. This is the form of tabulation commonly used during the first step of performing kernel density estimation, prior to the application of the kernel.

3.4 Color Name Representation Learning (Input Modules)

For each of the models we investigate we define an input module. This module handles the input of the color name, which is provided as a sequence of tokens. It produces a fixed sized dense representation of the color name, which is the input to the output module Section 3.5). In all models the input and output modules are trained concurrently as a single system.

3.4.1 Recurrent Neural Network(RNN). A Recurrent Neural Network is a common choice for this kind of task, due to the variable length of the input. The general structure of this network, shown in Figure 2 is similar to Monroe, Goodman, and Potts (2016), or indeed to most other word sequence learning models. Each word is first transformed to an embedding representation. This representation is trained with the rest of the network allowing per word information to be efficiently learned. The embedding is used as the input for a Gated Recurrent Unit (GRU). The stream was terminated with an End of Stream (<EOS>) pseudo-token, represented using a zero-vector. The output of the last time-step is fed to a Rectified Linear Unit (ReLU).

We make use of a GRU (Cho et al. 2014), which we found to marginally out-perform the basic RNN in preliminary testing. The small improvement is unsurprising, as the color names have at most 5 terms, so longer short term memory is not required.

3.4.2 Sum of Word Embeddings (SOWE). Using a simple sum of word embeddings as a layer in a neural network is less typical than an RNN structure. Though it is well established as a useful representation, and has been used as an input to other classifiers such as support vector machines (e.g. as in White et al. (2015, 2018)). Any number of word embeddings can be added to the sum, thus allowing it to handle sequences of any length. However, it has no representation of the order. The structure we used is shown in Figure 3.

3.4.3 Convolutional Neural Network(CNN). A convolutional neural network (shown in Figure 4) can be applied to the task by applying 2D convolution over the stacked word embeddings. We use 64 filters of size between one and five. Five is number of tokens in the longest color-name, so this allows it to learn full length relations.

Lyndon White et. al. Learning of Colors from Color Names: Distribution and Point Estimation

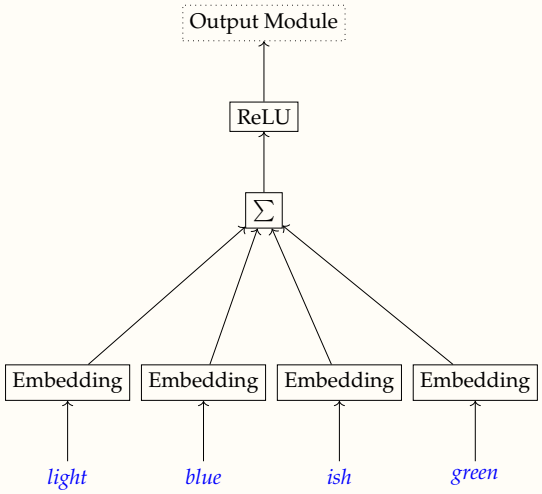


Figure 3
The SOWE input module for the example input `light bluish green`

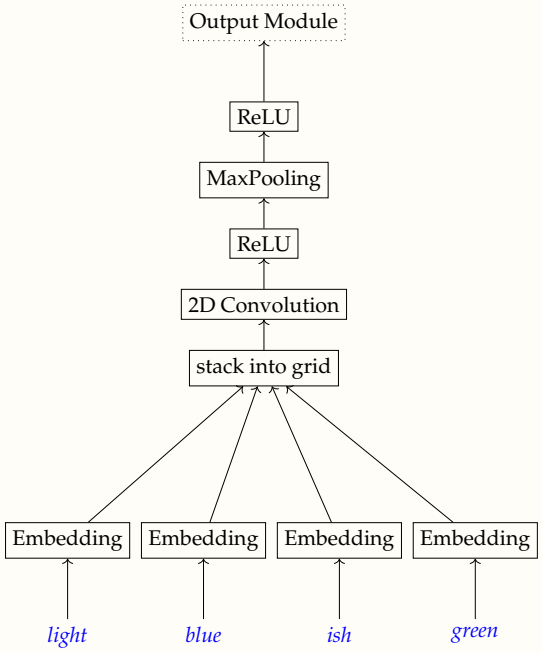


Figure 4
The CNN input module for the example input `light greenish blue`

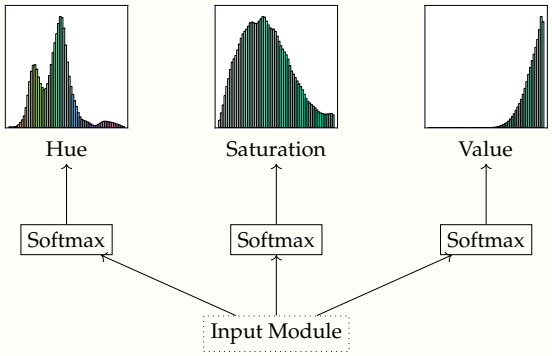


Figure 5
The Distribution Output Module

3.5 Distribution and Point Estimation (Output Modules)

On top of the input module, we define an output module to suit the neural network for the task of either distribution estimation or point estimation. The input module defines how the terms are composed into the network. The output module defines how the network takes its hidden representation and produces an output.

3.5.1 Distribution Estimation. The distributions are trained to produce the discretized representation as discussed in Section 3.3.2. Making use of the conditional independence assumption (see Section 3.3.1), we output the three discretized distributions. As shown in Figure 5, this is done using three softmax output layers – one per channel. They share a common input, but have separate weights and biases. The loss function is given by the sum of the cross-entropy for each of the three softmax outputs.

3.5.2 Point Estimation. Our point estimation output module is shown in Figure 6. The hidden-layer from the top of the input module is fed to four single output neurons.[3] Two

3 Equivalently these four single neurons can be expressed as a layer with four outputs and two different activation functions.

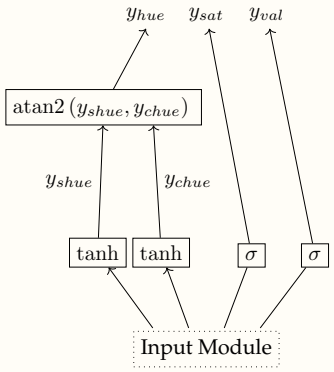


Figure 6
The Point Estimate Output Module. Here atan2 is the quadrant preserving arctangent, outputting the angle in turns.

of these use the sigmoid activation function (range 0:1) to produce the outputs for the saturation and value channels. The other two use the tanh activation function (range -1:1), and produce the intermediate output that we call y_{shue} and y_{chue} for the sine and cosine of the hue channel respectively. The hue can be found as $y_{hue} = \text{atan2}\,(y_{shue}, y_{chue})$. We use the intermediate values when calculating the loss function. During training we use the following loss function for each observation $y^\star$, and each corresponding prediction y.

$$\begin{aligned} loss = &\frac{1}{2}\left(\sin(y^\star_{hue}) - y_{shue}\right)^2 \\ &+ \frac{1}{2}\left(\cos(y^\star_{hue}) - y_{chue}\right)^2 \\ &+ \left(y^\star_{sat} - y_{sat}\right)^2 \\ &+ \left(y^\star_{val} - y_{val}\right)^2 \end{aligned} \tag{1}$$

This mean of this loss is taken over all observations in each mini-batch during training. This loss function is continuous and correctly handles the wrap-around nature of the hue channel (White 2016).

4. Evaluation

4.1 Perplexity in Color-Space

Perplexity is a measure of how well the distribution, estimated by the model, matches the reality according to the observations in the test set. Perplexity is commonly used for evaluating language models. Here however, it is being used to evaluate the discretized distribution estimate. It can be loosely thought of as to how well the model's distribution does in terms of the size of an equivalent uniform distribution. Note that this metric does not assume conditional independence of the color channels.

Here τ is the test-set made up of pairs consisting of a color name t, and a color-space point $\tilde{x}$; and $p(\tilde{x} \mid t)$ is the output of the evaluated model. Perplexity is defined as:

$$PP(\tau) = \exp_2 \left(\frac{-1}{|\tau|} \sum_{\forall (t,(\tilde{x})) \in \tau} \log_2 p(\tilde{x} \mid t) \right) \tag{2}$$

As the perplexity for a high-resolution discretized model will inherently be very large and difficult to read, we define the standardized perplexity: $\frac{PP(\tau)}{n_{res}}$, where n_{res} is the total number of bins in the discretization scheme. For all the results we present here $n_{res} = 256^3$. This standardized perplexity gives the easily interpretable values *usually* between zero and one. It is equivalent to comparing the relative performance of the model to that of a uniform distribution of the same total resolution. $\frac{PP(\tau)}{n_{res}} = 1$ means that the result is equal to what we would see if we had distributed the probability mass uniformly into all bins in a 3D histogram. $\frac{PP(\tau)}{n_{res}} = 0.5$ means the result is twice as good as if we were to simply use a uniform distribution: it is equivalent to saying that the correct bin is selected as often as it would be had a uniform distribution with half as many bins been used (i.e. larger bins with twice the area). The standardised perplexity is also invariant under different output resolutions. Though for brevity we only present

results with 256 bins per channel, our preliminary results for using other resolutions are similar under standardized perplexity.

4.2 Angularly Correct Calculations on HSV

We use the HSV color-space (Smith 1978) through-out this work. In this format: hue, saturation and value all range between zero and one. Note that we measure hue in *turns*, rather than the more traditional degrees, or radians. Having hue measured between zero and one, like the other channels, makes the modelling task more consistent. Were the hue to range between 0 and 2π (radians) or between 0 and 360 (degrees) it would be over-weighted in the loss function and evaluation metrics compared to the other channels. This regular space means that errors on all channels can be considered equally. Unlike many other colors spaces (CIELab, Luv etc.) the gamut is square and all combinations of values from the different channels correspond to realizable colors.

When performing calculations with the HSV color-space, it is important to take into account that hue is an angle. As we are working with the color-space regularized to range between zero and one for all channels, this means that a hue of one and a hue of zero are equivalent (as we measure in turns, in radians this would be 0 and 2π).

The square error of two hue values is thus calculated as:

$$SE(h_1, h_2) = \min\left((h_1 - h_2)^2, (h_1 - h_2 - 1)^2 \right) \tag{3}$$

This takes into account that the error can be calculated clockwise or counter-clockwise; and should be the smaller. Note that the -1 term is related to using units of turns, were we using radians it would be -2π

The mean of a set of hues ($\{h_1, \ldots, h_N\}$) is calculated as:

$$\bar{h} = \operatorname{atan2}\left(\frac{1}{N}\sum_{i=1}^{i=N} \sin(h_i), \frac{1}{N}\sum_{i=1}^{i=N} \cos(h_i) \right) \tag{4}$$

This gives the mean angle.

4.3 Operational Upper-bounds

To establish a rough upper-limit on the modelling results we evaluate a direct method which bypasses the language understanding component of the task. These direct methods do not process each term in the name:, they do not work with the language at all. They simply map from the exact input text (no tokenization) to the pre-calculated distribution or mean of the training data for the exact color name. This operational upper bound bypass the compositional language understanding part of the process. It is as if the input module (as discussed in Section 3.4) would perfectly resolve the sequence of terms into a single item.

They represent an approximate upper bound, as they fully exploit all information in the training data for each input. There is no attempt to determine how each term affects the result. We say approximate upper-bound, as it is not strictly impossible that the term-based methods may happen to model the test data better than can be directly determined by training data as processed per exact color name. This would require learning how the terms in the color name combine in a way that exceeds the information directly present in the training data per class. It is this capacity of learning how the terms combine that allow for the models to predict the outputs for combinations of terms that never occur in the training data (Section 4.4.2). Doing this in a way that generalizes to get better results than the direct exploitation of the training data, would require very well calibrated control of (over/under)fitting.

4.3.1 Operational Upper-bound for Distribution Estimation: KDE. To determine an operational upper-bound for the distribution estimation tasks, we use kernel-density estimation (KDE) in a formulation for non-parametric estimation (Silverman 1986) . The

KDE effectively produces a smoothed histogram from the training data as processed in Section 3.3.2. It causes adjacent bins to have most similar probabilities, thus matching to the mathematical notion of a continuous random variable. This is applied on-top of the histogram used for the training data. We use the Fast Fourier Transform (FFT) based KDE method of the Silverman (1982). We use a Gaussian kernel, and select the bandwidth per color description based on leave-one-out cross validation on the training data. A known issue with the FFT-based KDE method is that it has a wrap-around effect near the boundaries, where the mass that would be assigned outside the boundaries is instead assigned to the bin on the other side. For the value and saturation channels we follow the standard solution of initially defining additional bins outside the true boundaries, then discarding those bins and rescaling the probability to one. For the hue channel this wrap-around effect is exactly as desired.

In our evaluations using KDE rather than just the training histograms directly proved much more successful on all distribution estimation tasks. This is because it avoids empty bins, and effectually interpolates probabilities between observations. We found in preliminary investigations that using KDE-based method to be much better than add-one smoothing.

We also investigated the application of KDE to the training data, before training our term-based neural network based distribution models. Results for this can be found in Appendix 2. In brief, we found that smoothing the training data does not significantly affect the result of the neural network based models. As discussed in Section 6.2, this is because the neural networks are able to learn the smoothness relationship of adjacent bins.

Our KDE-based operational upper bound for distribution estimation bypasses the natural language understanding part of the task, and directly uses the standard non-

parametric probability estimation method to focus solely on modelling the distributions. Matching its performance indicates that a model is effectively succeeding well at both the natural language understanding component and the distribution estimation component.

4.3.2 Operational Upper-bound for Point Estimation: Mean-point. In a similar approach, we also propose a method that directly produces a point estimate from a color name. We define this by taking the mean of all the training observations for a given exact color name. The mean is taken in the angularly correct way (as discussed in Section 4.2). Taking the mean of all the observations gives the theoretically optimal solution to minimize the squared error on the training data set. As with our direct distribution estimation method, this bypasses the term based language understanding, and directly exploits the training data. It thus represents an approximate upper bound on the point estimation performance of the term based models. Though, as discussed in Section 1, the notion of mean and of minimizing the square error is not necessarily the correct way to characterize selecting the optimal point estimate for colors. It is however a consistent way to do so, and so we use it for our evaluations.

4.4 Evaluation Strategies

4.4.1 Full Task. We make use of the Munroe dataset as prepared by McMahan and Stone (2015) from the results of the XKCD color survey. The XKCD color survey (Munroe 2010), collected over 3.4 million observations from over 222,500 respondents. McMahan and Stone take a subset from Munroe's full survey, by restricting it to the responses from native English speakers, and removing very rare color names with less than 100 uses. This gives a total of 2,176,417 observations and 829 color names. They also define a standard test, development and train split.

4.4.2 Unseen combination Task. A primary interest in using the term based models is to be able to make predictions for never before seen descriptions of colors. For example, based on the learned understanding of `salmon` and of `bright`, from examples like `bright green` and `bright red`, we wish for the system to make predictions about `bright salmon`, even though that description never occurs in the training data. The ability to make predictions, such as these, illustrates term-based natural language understanding. This cannot be done with the operational upper bound models, which bypasses the term processing step. To evaluate this generalisation capacity, we define new sub-datasets for both testing and training. We select the rarest 100 color descriptions from the full dataset, with the restriction that every token in a selected description must still have at least 8 uses in other descriptions in the training set. The selected examples include multi-token descriptions such as: `bright yellow green` and also single tokens that occur more commonly as modifiers than as stand-alone descriptions such as `pale`.

The unseen combination testing set has only observations from the full test set that do use those rare descriptions. We define a corresponding restricted training set made up of the data from the full training set, excluding those corresponding to the rare descriptions. Similar restriction is done to create a restricted development set, so that no direct knowledge of the combined terms can leak during early-stopping.

By training on the restricted training set and testing on the unseen combination, we can assess the capacity of the models to make predictions for color descriptions not seen during training. A similar approach was used in Winn and Muresan (2018) and in Atzmon et al. (2016). We contrast this to the same models when trained on the full training set to see how much accuracy was lost.

4.4.3 Order Task. It is believed that the order of words in a color description matters, at least to some extent, for it's meaning. For example, `greenish brown` and `brownish`

`green` are distinct, if similar, colors. To assess the models on their ability to make predictions when order matters we construct the order test set. This is a subset of the full test set containing only descriptions with terms that occur in multiple different orders. There are 76 such descriptions in the full dataset. Each of which has exactly one alternate ordering. This is unsurprising as while color descriptions may have more than 2 terms, normally one or more of the terms is a joining token such as `ish` or `-`. We only construct an order testing set, and not a corresponding training set, as this is an evaluation using the model trained on the full training data.

5. Experimental Setup

5.1 Implementation

The implementation of all the models was in the julia programming language (Bezanson et al. 2014). The full implementation can be downloaded from the GitHub repository.[4] The machine learning components make heavy use of the MLDataUtils.jl[5] and TensorFlow.jl,[6] packages. The latter of which we enhanced significantly to allow for this work to be carried out. The discretization and the KDE for the Operational Upper Bound is done using KernalDensityEstimation.jl.[7]

5.2 Common Network Features

Drop-out(Srivastava et al. 2014) is used on all ReLU layers and on the GRU in the RNN, with threshold of 0.5 during training. The network is optimized using Adam (Kingma and Ba 2014), and a learning rate of 0.001. Early stopping is checked every 10 epochs using the development dataset. Distribution estimation methods are trained using the

4 Implementation source is at `https://github.com/oxinabox/ColoringNames.jl`
5 MLDataUtils.jl is available from `https://github.com/JuliaML/MLDataUtils.jl`
6 TensorFlow.jl is available from `https://github.com/malmaud/TensorFlow.jl`
7 KernalDensityEstimation.jl is available from `https://github.com/JuliaStats/KernelDensity.jl`

full batch (where each observation is a distribution) for every epoch. Point Estimation methods are trained using randomized mini-batches of 2^{16} observations (which are each color-space triples). All hidden-layers, except as otherwise precluded (inside the convolution, and in the penultimate layer of the point estimation networks) have the same width 300, as does the embedding layer.

6. Results

6.1 Qualitative Results

To get an understanding of the problem and how the models are performing, we consider some of the outputs of the model for particular cases. For models trained on the full dataset Figure 7 shows examples of distribution estimates, and Figure 8 shows similar examples for point estimates. It can be seen that the model's outputs using term based estimations are generally similar to the non-term-based Operational Upper-Bound, as is intended. This shows models are correctly fitting to estimate the colors. This aligns with the strong results found in that quantitative evaluations discussed in Section 6.2.

6.1.1 Qualitative results on word-order. The different input modules have different capacity to leverage word-order. This is reflected in Figures 7 and 8, when considering the pairs of outputs that differ only in word order, such as `purple-pink` and `pink-purple`. The plots presented for the training data and for the Operational Upper-Bound show that such color name pairs are subtly different but similar. The SOWE model is unable to take into account word order at all, and so produces identical outputs for all orders The CNN models produce very similar outputs but not strictly identical – spotting the difference requires very close observation. This is in-line with the different filter sizes allowing the CNN to use effectively use n-gram features, and finding that the unigram features are the most useful. The RNN produces the most strikingly different results.

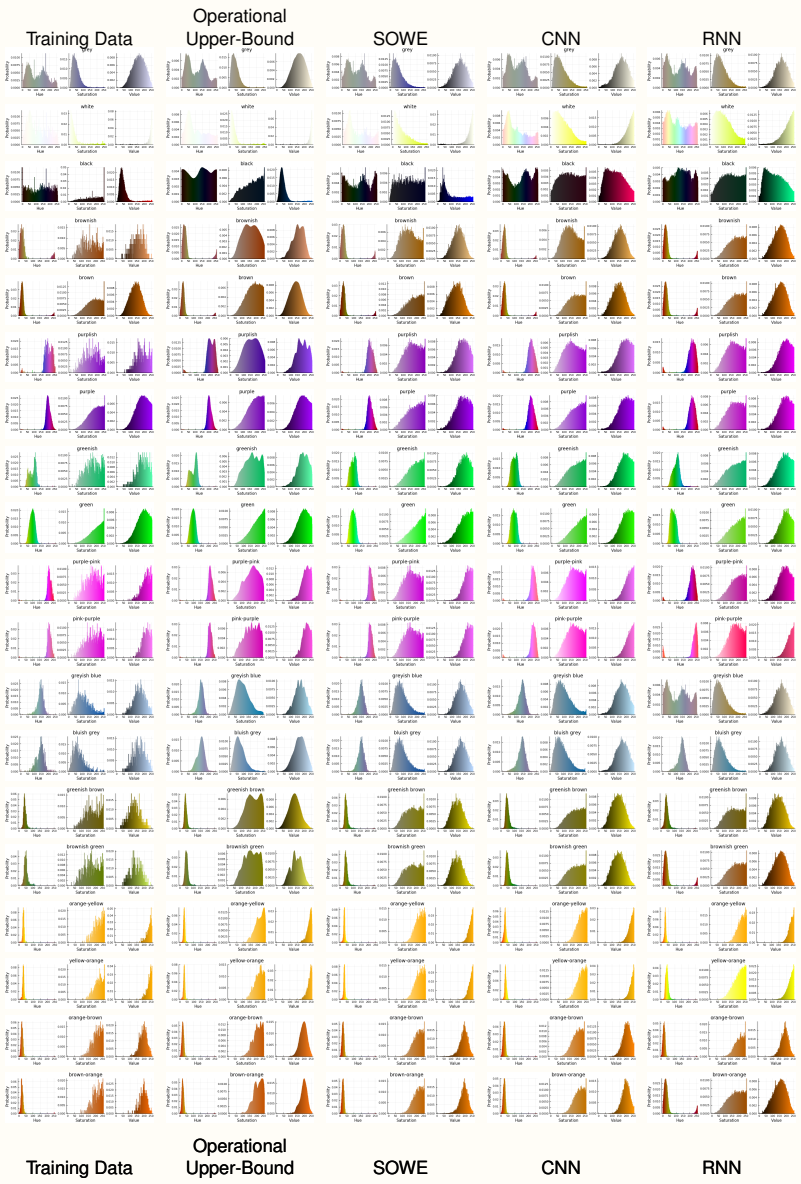


Figure 7
Some examples of the output distribution estimations from the models trained on the full dataset

Lyndon White et. al. Learning of Colors from Color Names: Distribution and Point Estimation

Figure 8
Some examples of the output point estimates from the models trained on the full dataset

27

It seems that the first term dominates the final output: `purple-pink` is more purple, and `pink-purple` is more pink. We can see that the first term is not solely responsible for the final output however, as `purple-pink`, `purple` and `purplish` (tokenized as `purple`, `ish`) are all different. It is surprising that the RNN is dominated by the first term and not the latter terms.[8] This shows the GRU is functioning to remember the earlier inputs. This is happening too strongly however, as it is causing incorrect outputs.

6.1.2 On smoothness in learned distribution estimates. In Figure 7 it can be seen that the term-based distribution estimation models are much smoother than the corresponding histograms taken from the training data. They are not as smooth as the Operational Upper-Bound which explicitly uses KDE. However, they are much smoother than would be expected had the output bins been treated independently. Thus it is clear that the models are learning that adjacent bins should have similar output values. This is a common feature of all the training data, no matter which color is being described. This learned effect is in line with the fact that color is continuous, and is only being represented here as discrete. We note in relation to this learned smoothness: that while the models capture the highly asymmetrical shapes of most distributions well, they do not do well at capturing small dips. Larger multi-modes as seen in the achromatic colors such as `white`, `grey`, `black`, `white`, are captured; but smaller dips such as the hue of `greenish` being most likely to be either side of the green spectrum are largely filled in. In general, it seems clear that additional smoothing of the training data is not required for the neural network based models. This aligns with the results presented in Appendix 2.

8 So much so that we double checked our implementation to be sure it wasn't processing the inputs backwards.

6.2 Quantitative Results

Overall, we see that our models are able to learn to estimate colors based on sequences of terms. From the consideration of all results Tables 1 to 6. The CNN and SOWE models perform almost as well as the Operational Upper-Bound. With the SOWE having a marginal lead for distribution estimation, and the CNN and SOWE being nearly exactly equal for most point estimation tasks. We believe the reason for this is the SOWE is an easier to learn model from a gradient descent perspective: it is a shallow model with only one true hidden layer. The RNN did not perform as well at these tasks. While it is only marginally behind the SOWE and CNN on the full point estimation task (Table 2), on all other tasks for both point estimation and distribution estimation is is significantly worse. This may indicate that it is hard to capture the significant relationship between terms in the sequence. However, as shown Section 6.2 it did learn generally acceptable colors, however it is not as close a match to the population's expectation.

6.2.1 Ordered Task. The performance of SOWE on the order tasks (Tables 3 and 4) is surprising. For distribution estimation it outperforms the CNN, and for point estimation ties with the CNN. The CNN (and RNN) can take into account word order, but the SOWE model can not. Its good results suggest that the word-order is not very significant for color names. While word order matters, different colors with the same terms in different order are similar enough that it still performs very well. In theory the models capable of using word order have the capacity to ignore it, and thus could achieve a similar result. An RNN can learn to perform a sum of its inputs (the word embeddings), and the CNN can learn to weight all non-unigram filters to zero. In practice we see that for the RNN in particular this clearly did not occur. This can be attributed to the more complex networks

being more challenging to train via gradient descent. It seems color-naming is not a task where word order substantially matters, and thus the simpler SOWE model excels.

6.3 Unseen Combinations of Terms

The SOWE and CNN models are able to generalize well to making estimates for combinations of color terms not seen in training. Tables 5 and 6 show the results of the model on the test set made up of rare combinations of color names (as described in Section 4.4.2) for the restricted training set (which does not contain those terms). These results on this test set are compared with the same models when trained on the full training set. The Operation Upper Bound models are unable to produce estimations from the unseen combinations testing set as they do not process the color names term-wise. The RNN models continue to perform badly on the unseen combination of terms task for both point and distribution estimation.

For distribution estimation (Table 5) the SOWE results are only marginally worse for the restricted training set as they are for the full training set. The CNN results are worse again, but still are better than the results on the full test-set. The distributions estimated are good on absolute terms getting low perplexity

In the point estimation (Table 6) task the order is flipped with the CNN outperforming the SOWE model. In-fact the CNN actually performs better with the restricted training set. This may be due to the CNN not fitting to the training data as well as the SOWE, as on the full training set (for this test set) we see the SOWE out performs the CNN. This suggests that the CNN point estimation model may be better capturing the shared information about term usages, at the expense of fitting to the final point. Unlike for distribution estimates the unseen color point estimates are worse than the overall results from the full task (Table 2), though the errors are still small on an absolute scale.

6.3.1 Extracting the mean from the distribution estimates. In the point estimation results discussed so far have been from models trained specifically for point estimation (as described by Section 3.5.2). However, it is also possible to derive the mean from the distribution estimation models. Those results are also presented in Tables 2, 4 and 6. In general these results perform marginally worse (using the MSE metric) than their corresponding modules using the point estimation output module. We note that for the Operational Upper Bounds, the distributions mean is almost identical to the true mean of points, as is expected.

Beyond the output module there are a few key differences between the point estimation modules and the distribution estimate modules. When training distribution estimates all examples of a particular color name is grouped into a single high information training observation using the histogram as the output. Whereas when training for point estimates we process each example individually, while using minibatches This means that that distribution estimating models fits to all color names with equal priority.Whereas for point estimates, more frequently used color names have more examples, and so more frequent color names are fit with priority over rarer ones. Another consequence of using training per example, rather than aggregating is that training via random minibatch is more resistant to local minima. One of the upsides of the aggregated training used in distribution estimation is that it trains much faster as only a small number of high-information training examples must be processed, rather than a much larger number of individual observations. It may be interesting in future work to consider training the distribution estimates per example using one-hot output representations; thus making the process similar to that used in the point estimate training. We suspect that such a method may have trouble learning the smoothness of the output space (as discussed in Section 6.1.2).

Method	$\frac{PP}{256^3}$
Operational Upper Bound	0.071
SOWE	**0.075**
CNN	0.078
RNN	0.089

Table 1
The results for the **full distribution estimation task**. Lower perplexity (PP) is better.

Method	MSE
Operational Upper Bound	0.066
SOWE	**0.067**
CNN	**0.067**
RNN	0.071
Distribution Mean Operational Upper Bound	0.066
Distribution Mean SOWE	0.068
Distribution Mean CNN	0.069
Distribution Mean RNN	0.077

Table 2
The results for the **full point estimation task**. Lower mean squared error (MSE) is better.

Method	$\frac{PP}{256^3}$
Operational Upper Bound	0.053
SOWE	**0.055**
CNN	0.057
RNN	0.124

Table 3
The results for the **order distribution estimation task**. Lower perplexity (PP) is better. This is a subset of the full test set containing only tests where the order of the words matters.

Method	MSE
Operational Upper Bound	0.065
SOWE	**0.066**
CNN	**0.066**
RNN	0.096
Distribution Mean Operational Upper Bound	0.065
Distribution Mean SOWE	**0.066**
Distribution Mean CNN	**0.066**
Distribution Mean RNN	0.095

Table 4
The results for the **order point estimation task**. Lower mean squared error (MSE) is better. This is a subset of the full test set containing only tests where the order of the words matters.

Method	Full Training Set $\frac{PP}{256^3}$	Restricted Training Set $\frac{PP}{256^3}$
Operational Upper Bound	0.050	–
SOWE	**0.050**	**0.055**
CNN	0.052	0.065
RNN	0.117	0.182

Table 5
The results for the **unseen combinations distribution estimation task**. Lower perplexity (PP) is better. This uses the extrapolation subset of the full test set. In the extrapolating results certain rare word combinations were removed from the training and development sets. In the non-extrapolating results the full training and development stet was used.

Method	Full Training Set MSE	Restricted Training Set MSE
Operational Upper Bound	0.062	–
SOWE	**0.065**	0.079
CNN	0.072	**0.070**
RNN	0.138	0.142
Distribution Mean Operational Upper Bound	0.062	–
Distribution Mean SOWE	0.073	0.076
Distribution Mean CNN	0.073	0.084
Distribution Mean RNN	0.105	0.152

Table 6
The results for the **unseen combinations point estimation task**. Lower mean squared error (MSE) is better. This uses the extrapolation subset of the full test set. In the extrapolating results certain rare word combinations were removed from the training and development sets. In the non-extrapolating results the full training and development stet was used.

6.4 Completely Unseen Color Estimation From Embedding Only

As an interesting demonstration of how the models function by learning the transformation from the embedding space to the output we briefly considered the outputs for color-names that do not occur in the training data (or testing) at all. This is even more extreme than the unseen combination task considered in Table 5 and Table 6 where the terms appeared in training, but not the combination of terms. In the examples shown in Figure 9 and Figure 10, the terms never occurred in the training data at all, but our models exploit the fact that they work by transforming the word-embedding space to predict

the colors. There is no equivalent for this in the direct models. While `Grey` and `gray` never occur in the training data; `grey` does, and it is near-by in the word-embedding space. Similar is true for the other colors that vary by capitalization. We only present a few examples of single term colors here, and no quantitative investigation, as this is merely a matter of interest.

It is particularly interesting to note that that all the models make similar estimations for each color. This occurs both for point estimation and for distribution estimation. They do well on the same colors and make similar mistakes on the colors they do poorly at. The saturation of `Gray` is estimated too high, making it appear too blue/purple, this is also true of `grey` though to a much lesser extent. `Purple` and `Green` produce generally reasonable estimates. The hue for `Brown` is estimated as having too much variance, allowing the color to swing into the red or yellowish-green parts of the spectrum. This suggests that in general all models are learning a more generally similar transformation of the space. In general the overall quality of each model seems to be in line with that found in the results for the full tests.

7. Conclusion

We have presented three input modules (SOWE, CNN, RNN), and two output modules (distribution estimate, and point estimate) suitable for using machine learning to make estimates about color based on the terms making up its name. We contrasted these to an operational upper bound model for each task which bypassed the term-wise natural language understanding component of the problem. We found the results for SOWE, and CNN were very strong, approaching this upper bound.

A key take away from our results is that using a SOWE should be preferred over an RNN for short phrase natural language understanding tasks when order is not a very significant factor. The RNN is the standard type of model for problems with sequential

Lyndon White et. al. Learning of Colors from Color Names: Distribution and Point Estimation

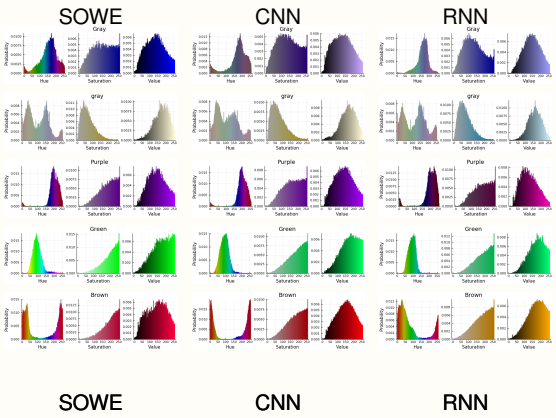


Figure 9
Some example distribution estimations for colors names which are completely outside the training data. The terms: `Brown`, `gray`, `Gray`, `Green`, and `Purple`, do not occur in any of the color data; however `brown`, `grey green`, and `purple` do occur.

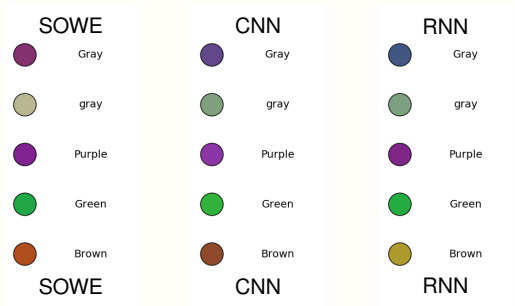


Figure 10
Some example point estimates for colors names which are completely outside the training data. The terms: `Brown`, `gray`, `Gray`, `Green`, and `Purple`, do not occur in any of the color data; however `brown`, `grey green`, and `purple` do occur.

input, such as color names made up of multiple words as we considered here. However, we find its performance to be significantly exceeded by the SOWE and CNN. The SOWE is an unordered model roughly corresponding to a bag of words. The CNN over word embeddings similarly roughly correspondents to a bag of ngrams, in our case a bag of all 1,2,3,4 and 5-grams. This means the CNN can readily take advantage of both fully

ordered information, using the filters of length 5, down to unordered information using the filters of length one. The RNN however must fully process the ordered nature of its inputs, as its output comes only from the final node. It would be interesting to further contrast a bidirectional RNN.

In a broader context, we envisage the distribution learned for a color name can be used as a prior probability and when combining with additional context information, this can be used for better prediction in areas such as document classification and sentiment detection.

A further interesting avenue for investigation would condition the model not only on the words used but also on the speaker. The original source of the data Munroe (2010), includes some demographic information which is not exploited by any known methods. It is expected that color-term usage may vary with subcultures.

References

Atzmon, Yuval, Jonathan Berant, Vahid Kezami, Amir Globerson, and Gal Chechik. 2016. Learning to generalize to new compositions in image understanding. *CoRR*, abs/1608.07639.

Berk, Toby, Arie Kaufman, and Lee Brownston. 1982. A human factors study of color notation systems for computer graphics. *Commun. ACM*, 25(8):547–550.

Berlin, Brent and Paul Kay. 1969. *Basic color terms: Their university and evolution*. California UP.

Bezanson, Jeff, Alan Edelman, Stefan Karpinski, and Viral B. Shah. 2014. Julia: A fresh approach to numerical computing.

Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Cho, Kyunghyun, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*.

Conway, Damian. 1992. An experimental comparison of three natural language colour naming models. In *Proc. east-west int. conf. on human-computer interaction*, pages 328–339.

Heider, Eleanor R. 1972. Universals in color naming and memory. *Journal of experimental psychology*, 93(1):10.

Heider, Eleanor Rosch and Donald C. Olivier. 1972. The structure of the color space in naming and memory for two languages. *Cognitive Psychology*, 3(2):337 – 354.

Jones, MC and HW Lotwick. 1984. Remark as r50: a remark on algorithm as 176. kernal density estimation using the fast fourier transform. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 33(1):120–122.

Kawakami, Kazuya, Chris Dyer, Bryan R. Routledge, and Noah A. Smith. 2016. Character sequence models for colorfulwords. *CoRR*, abs/1609.08777.

Kelly, Kenneth Low et al. 1955. Iscc-nbs method of designating colors and a dictionary of color names.

Kingma, Diederik and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lammens, Johan Maurice Gisele. 1994. *A Computational Model of Color Perception and Color Naming*. Ph.D. thesis, State University of New York.

Leshno, Moshe, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. 1993. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867.

Mansimov, E., E. Parisotto, J. Lei Ba, and R. Salakhutdinov. 2015. Generating Images from Captions with Attention. *ArXiv e-prints*.

McMahan, Brian and Matthew Stone. 2015. A bayesian model of grounded color semantics. *Transactions of the Association for Computational Linguistics*, 3:103–115.

Menegaz, Gloria, Arnaud Le Troter, Jean Sequeira, and Jean-Marc Boi. 2007. A discrete model for color naming. *EURASIP Journal on Applied Signal Processing*, 2007(1):113–113.

Meo, T., B. McMahan, and M. Stone. 2014. Generating and resolving vague color reference. *Proc. 18th Workshop Semantics and Pragmatics of Dialogue (SemDial)*.

Mojsilovic, Aleksandra. 2005. A computational model for color naming and describing color composition of images. *IEEE Transactions on Image Processing*, 14(5):690–699.

Monroe, W., N. D. Goodman, and C. Potts. 2016. Learning to Generate Compositional Color Descriptions. *ArXiv e-prints*.

Monroe, Will, Robert X. D. Hawkins, Noah D. Goodman, and Christopher Potts. 2017. Colors in context: A pragmatic neural model for grounded language understanding. *CoRR*, abs/1703.10186.

Munroe, Randall. 2010. Xkcd: Color survey results.

Mylonas, Dimitris, Matthew Purver, Mehrnoosh Sadrzadeh, Lindsay MacDonald, and Lewis Griffin. 2015. The use of english colour terms in big data. The Color Science Association of Japan.

van den Oord, Aäron, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499.

Oord, Aaron van den, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*.

Reed, Scott, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. 2016. Generative adversarial text to image synthesis. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 3.
Silverman, BW. 1982. Algorithm as 176: Kernel density estimation using the fast fourier transform. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 31(1):93–99.
Silverman, B.W. 1986. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis.
Smith, Alvy Ray. 1978. Color gamut transform pairs. *ACM Siggraph Computer Graphics*, 12(3):12–19.
Sonoda, Sho and Noboru Murata. 2017. Neural network with unbounded activation functions is universal approximator. *Applied and Computational Harmonic Analysis*, 43(2):233 – 268.
Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.
Van De Weijer, Joost, Cordelia Schmid, Jakob Verbeek, and Diane Larlus. 2009. Learning color names for real-world applications. *IEEE Transactions on Image Processing*, 18(7):1512–1523.
White, Lyndon. 2016. Encoding angle data for neural networks. Cross Validated Stack Exchange.
White, Lyndon, Roberto Togneri, Wei Liu, and Mohammed Bennamoun. 2015. How well sentence embeddings capture meaning. In *Proceedings of the 20th Australasian Document Computing Symposium*, ADCS '15, pages 9:1–9:8, ACM.
White, Lyndon, Roberto Togneri, Wei Liu, and Mohammed Bennamoun. 2018. Novelperspective. *Pending submission*.
Winn, Olivia and Smaranda Muresan. 2018. 'lighter' can still be dark: Modeling comparative color descriptions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 790–795, Association for Computational Linguistics.

1. On the Conditional Independence of Color Channels given a Color Name

As discussed in the main text, we conducted a superficial investigation into the truth of our assumption that given a color name, the distributions of the hue, value and saturation are statistically independent.

We note that this investigation is, by no means, conclusive though it is suggestive. The investigation focusses around the use of Spearman's rank correlation. This correlation measures the monotonicity of the relationship between the random variables. A key limitation is that the relationship may exist but be non-monotonic. This is almost certainly true for any relationship involving channels, such as hue, which wrap around. In the case of such relationships Spearman's correlation will underestimate the true strength of the relationship. Thus, this test is of limited use in proving the conditional independence. However, it is a quick test to perform and does suggest that the conditional independence assumption may not be so incorrect as one might assume.

For the Monroe Color Dataset training data given by $V \subset \mathbb{R}^3 \times T$, where $\mathbb{R}^3$ is the value in the color-space under consideration, and T is the natural language space. The subset of the training data for the description $t \in T$ is given by $V_{|t} = \{(\tilde{v}_i, t_i) \in V \mid t_i = t\}$. Further let $T_V = \{t_i \mid (\tilde{v}, t_i) \in V$ be the set of color names used in the training set. Let $V_{\alpha|t}$ be the α channel component of $V_{|t}$, i.e. $V_{\alpha|t} = \{v_\alpha \mid ((v_1, v_2, v_3), t) \in V_{|t}\}$.

The set of absolute Spearman's rank correlations between channels a and b for each color name is given by $S_{ab} = \{|\rho(V_{a|t}, V_{b|t})| \mid t \in T_V\}$.

Color-Space	$Q3(S_{12})$	$Q3(S_{13})$	$Q3(S_{23})$	max
HSV	0.1861	0.1867	0.1628	0.1867
HSL	0.1655	0.2147	0.3113	0.3113
YCbCr	0.4005	0.4393	0.3377	0.4393
YIQ	0.4088	0.4975	0.4064	0.4975
LCHab	0.5258	0.411	0.3688	0.5258
DIN99d	0.5442	0.4426	0.4803	0.5442
DIN99	0.5449	0.4931	0.5235	0.5449
DIN99o	0.5608	0.4082	0.5211	0.5608
RGB	0.603	0.4472	0.5656	0.603
Luv	0.5598	0.6112	0.4379	0.6112
LCHuv	0.6124	0.4072	0.3416	0.6124
HSI	0.2446	0.2391	0.6302	0.6302
CIELab	0.573	0.4597	0.639	0.639
xyY	0.723	0.5024	0.4165	0.723
LMS	0.968	0.7458	0.779	0.968
XYZ	0.9726	0.8167	0.7844	0.9726

Table 7
The third quartile for the pairwise Spearman's correlation of the color channels given the color name.

We consider the third quartile of that correlation as the indicative statistic in Table 7. That is to say for 75% of all color names, for the given color-space, the correlation is less than this value.

Of the 16 color-spaces considered, it can be seen that the HSV exhibits the strongest signs of conditional independence – under this (mildly flawed) metric. More properly put, it exhibits the weakest signs of non-independence. This includes being significantly less correlated than other spaces featuring circular channels such as HSL and HSI.

Our overall work makes the conditional independence assumption, much like n-gram language models making Markov assumption. The success of the main work indicates that the assumption does not cause substantial issues.

2. KDE based smoothing of Training Data

It can be seen that smoothing has very little effect on the performance of any of the neural network based distribution estimation models. All 3 term based models (SOWE, CNN,

RNN) all perform very similarly whether or not the training data is smoothed. This is seen consistently in all the distribution estimation tasks. Contrast Tables 8 to 10 to the tables for the unsmoothed results Tables 1, 3 and 5.

If however, smoothing is not applied to the Operational Upper Bound, it works far worse. In Tables 8 to 10 the Direct result refers to using the training histograms almost directly, without any smoothing or term-based input processing. This is the same as the Operation Upper Bound, minus the KDE. It works very poorly (by comparison). This is because the bins values are largely independent: a very high probability in one bin does not affect the probability of the adjacent bin – which by chance of sampling my be lower than the trust distribution would have it.

This is particularly notable in the case of the direct, full training set result on the unseen combinations task reported in Table 10. As these were some of the rarest terms in the training set, several did not coincide with any bins for observations in testing set. This is because without smoothing it results in estimating the probability based on bins unfilled by any observation. We do cap that empty bin probability at $\epsilon_{64} \approxeq 2 \times 10^{-16}$ to prevent undefined perplexity. (We found capping the lower probability for bins like this to be far more effective than add-on smoothing).

Conversely, on this dataset the neural network models do quiet well, with or without smoothing. As the network can effectively learn the smoothness, not just from the observations of one color but from all of the observations. It learns that increasing the value of one bin should increase adjacent ones. As such it does not need the smoothing applied to the training data.

Method	$\frac{PP}{256^3}$
Direct	0.164
Operational Upper Bound	0.071
SOWE-smoothed	0.075
CNN-smoothed	0.079
RNN-smoothed	0.088

Table 8
The results for the **full distribution estimation task** using smoothed training data. Lower perplexity (PP) is better. This corresponds to the main results Table 1.

Method	$\frac{PP}{256^3}$
Direct	0.244
Operational Upper Bound	0.053
SOWE-smoothed	0.055
CNN-smoothed	0.058
RNN-smoothed	0.122

Table 9
The results for the **order distribution estimation task** using smoothed training data. Lower perplexity (PP) is better. This is a subset of the full test set containing only tests where the order of the words matters. This corresponds to the main results Table 3.

Method	Full Training Set $\frac{PP}{256^3}$	Restricted Training Set $\frac{PP}{256^3}$
Direct	175.883	–
Operational Upper Bound	0.050	–
SOWE-smoothed	0.050	0.056
CNN-smoothed	0.053	0.063
RNN-smoothed	0.112	0.183

Table 10
The results for the **unseen combinations distribution estimation task** using smoothed training data. Lower perplexity (PP) is better. This uses the extrapolation subset of the full test set. In the extrapolating results certain rare word combinations were removed from the training and development sets. In the non-extrapolating results the full training and development stet was used. This corresponds to the main results Table 5.

Finding Word Sense Embeddings Of Known Meaning

Lyndon White, Roberto Togneri, Wei Liu, and Mohammed Bennamoun

The University of Western Australia,
35 Stirling Highway, Crawley, Western Australia
lyndon.white@research.uwa.edu.au, roberto.togneri@uwa.edu.au,
wei.liu@uwa.edu.au, mohammed.bennamoun@uwa.edu.au

Abstract. Word sense embeddings are vector representations of polysemous words – words with multiple meanings. These induced sense embeddings, however, do not necessarily correspond to any dictionary senses of the word. To overcome this, we propose a method to find new sense embeddings with known meaning. We term this method refitting, as the new embedding is fitted to model the meaning of a target word in an example sentence. The new lexically refitted embeddings are learnt using the probabilities of the existing induced sense embeddings, as well as their vector values. Our contributions are threefold: (1) The refitting method to find the new sense embeddings; (2) a novel smoothing technique, for use with the refitting method; and (3) a new similarity measure for words in context, defined by using the refitted sense embeddings. We show how our techniques improve the performance of the Adaptive Skip-Gram sense embeddings for word similarly evaluation; and how they allow the embeddings to be used for lexical word sense disambiguation.

1 Introduction

Popular word embedding vectors, such as Word2Vec, represent a word's semantic meaning and its syntactic role as a point in a vector space [1, 2]. As each word is only given one embedding, such methods are restricted to the representation of only a single combined sense, or meaning, of the word. *Word sense embeddings* generalise word embeddings to handle polysemous and homonymous words. Often these sense embeddings are learnt through unsupervised Word Sense Induction (WSI) [3–6]. The induced sense embeddings are unlikely to directly coincide with any set of human defined meaning at all, i.e. they will not match lexical senses such as those defined in a lexical dictionary, e.g. WordNet [7]. These induced senses may be more specific, more broad, or include the meanings of jargon not in common use.

One may argue that WSI systems can capture better word senses than human lexicographers do manually. However, this does not mean that induced senses can replace standard lexical senses. It is important to appreciate the vast wealth of existing knowledge defined around lexical senses. Methods to link induced senses to lexical senses allow us to take advantage of both worlds.

We propose *a refitting method* to generate a sense embedding vector that matches with a labelled lexical sense. Given an example sentence with the labelled lexical sense of a particular word, the refitting method algorithmically combines the induced sense embeddings of the target word such that the likelihood of the example sentence is maximised. We find that in doing so, the sense of the word in that sentence is captured. With the refitting, the induced sense embeddings are now able to be used in more general situations where standard senses, or user defined senses are desired.

Refitting word sense vectors to match a lexicographical sense inventory, such as WordNet or a translator's dictionary, is possible if the sense inventory features at least one example of the target sense's use. Our method allows this to be done very rapidly, and from only the single example of use this has with possible applications in low-resource languages.

Refitting can also be used to fit to a user provided example, giving a specific sense vector for that use. This has strong applications in information retrieval. The user can provide an example of a use of the word they are interested in. For example, searching for documents about "`banks`" as in "the river banks were very muddy". By generating an embedding for that specific sense, and by comparing with the generated embeddings in the indexed documents, we can not only pick up on suitable uses of other-words for example "`beach`" and "`shore`", but also exclude different usages, for example of a financial bank. The method we propose, using our refitted embeddings, has lower time complexity than AvgSimC [3], the current standard method for evaluating the similarity of words in context. This is detailed in Section 5.1.

We noted during refitting, that a single induced sense would often dominate the refitted representation. It is rare in natural language for the meaning to be so unequivocal. Generally, a significant overlap exists between the meaning of different lexical senses, and there is often a high level of disagreement when humans are asked to annotate a corpus [8]. We would expect that during refitting there would likewise be contention over the most likely induced sense. Towards this end, we develop a smoothing method, which we call *geometric smoothing* that de-emphasises the sharp decisions made by the (unsmoothed) refitting method. We found that this significantly improves the results. This suggests that the sharpness of sense decisions is an issue with the language model, which smoothing can correct. The geometric smoothing method is presented in Section 3.2.

We demonstrate the refitting method on sense embedding vectors induced using Adaptive Skip-Grams (AdaGram) [6], as well as our own simple greedy word sense embeddings. The method is applicable to any skip-gram-like language model that can take a sense vector as its input, and can output the probability of a word appearing in that sense's context.

The rest of the paper is organised as follows: Section 2 briefly discusses two areas of related works. Section 3 presents our refitting method, as well as our proposed geometric smoothing method. Section 4 describes the WSI embedding models used in the evaluations. Section 5 defines the RefittedSim measure for word similarity in context, and presents its results. Section 6 shows how the refitted sense vectors can be used for lexical WSD. Finally, the paper concludes in Section 7.

2 Related Works

2.1 Directly Learning Lexical Sense Embeddings

In this area of research, the induction of word sense embeddings is treated as a supervised, or semi-supervised task, that requires sense labelled corpora for training.

Iacobacci et al. [9] use a Continuous Bag of Word language model [1], using word senses as the labels rather than words. This is a direct application of word embedding techniques. To overcome the lack of a large sense labelled corpus, Iacobacci et al. use a 3rd party WSD tool, BabelFly [10], to add sense annotations to a previously unlabelled corpus.

Chen et al. [11] use a supervised approach to train sense vectors, with an unsupervised WSD labelling step. They partially disambiguate their training corpus, using word sense vectors based on WordNet; and use these labels to train their embeddings. This relabelled data is then used as training data, for finding sense embeddings using skip-grams.

Our refitting method learns a new sense embedding as a weighted sum of existing induced sense embeddings of the target word. Refitting is a one-shot learning solution, as compared to the approaches used in the works discussed above. A notable advantage is the time taken to add a new sense. Adding a new sense is practically instantaneous, and replacing the entire sense inventory, of several hundred thousand senses, is only a matter of a few hours. Whereas for the existing approaches this would require repeating the training process, which will often take several days. Refitting is a process done to word sense embeddings, rather than a method for finding sense embeddings from a large corpus.

2.2 Mapping induced senses to lexical senses

By defining a stochastic map between the induced and lexical senses, Agirre et al. [12], propose a general method for allowing WSI systems to be used for WSD. Their work was used in SemEval-2007 Task 02 [13] to evaluate all entries. Agirre et al. use a mapping corpus to find the probability of a lexical sense, given the induced sense according to the WSI system. This is more general than the approach we propose here, which only works for sense embedding based WSI. By exploiting the particular properties of sense embedding based WSI systems we propose a system that can better facilitate the use of this subset of WSI systems for WSD.

3 Proposed Refitting Framework

The key contribution of this work is to provide a way to synthesise a word sense embedding given only a single example sentence and a set of pretrained sense embedding vectors. We termed this *refitting* the sense vectors. By refitting the unsupervised vectors we define a new vector, that lines up with the specific meaning of the word from the example sentence.

This can be looked at as a one-shot learning problem, analogous to regression. The training of the induced sense, and of the language model, can be considered

an unsupervised pre-training step. The new word sense embedding should give a high value for the likelihood of the example sentence, according to the language model. It should also generalise to give a high likelihood of other contexts where this word sense occurs.

We initially attempted to directly optimise the sense vector to predict the example. We applied the L-BFGS [14] optimisation algorithm with the sense vector being the parameter being optimised over, and the objective being to maximise the probability of the example sentence according to the language model. This was found to generalise poorly, due to over-fitting, and to be very slow. Rather than a direct approach, we instead take inspiration from the locally linear relationship between meaning and vector position that has been demonstrated for word embeddings [1, 15, 16].

To refit the induced sense embeddings to a particular meaning of a word, we express that a new embedding as as a weighted combination of the induced sense vectors. The weight is determined by the probability of each induced sense given the context.

Given a collection of induced (unlabelled) embeddings $\mathbf{u} = u_1,...,u_{n_u}$, and an example sentence $\mathbf{c} = w_1,...,w_{n_c}$ we define a function $l(\mathbf{u}\,|\,\mathbf{c})$ which determines the refitted sense vector, from the unsupervised vectors and the context as:

$$l(\mathbf{u}\,|\,\mathbf{c}) = \sum_{\forall u_i \in \mathbf{u}} u_i P(u_i\,|\,\mathbf{c}) \tag{1}$$

Bayes' Theorem can be used to estimate the posterior predictive distribution $P(u_i\,|\,\mathbf{c})$.

Bengio et al. [17] describe a similar method to Equation (1) for finding (single sense) word embeddings for words not found in their vocabulary. The formula they give is as per Equation (1), but summing over the entire vocabulary of words (rather than just $\mathbf{u}$).

3.1 A General WSD method

Using the language model and application of Bayes' theorem, we define a general word sense disambiguation method that can be used for refitting (Equation (1)), and for lexical word sense disambiguation (see Section 6). This is a standard approach of using Bayes' theorem [5, 6]. We present it here for completeness.

The context is used to determine which sense is the most suitable for this use of the *target word* (the word being disambiguated). Let $\mathbf{s} = (s_1,...,s_n)$, be the collection of senses for the target word[1].

Let $\mathbf{c} = (w_1,...,w_{n_c})$ be a sequence of words making up the context of the target word. For example for the target word *kid*, the context could be $\mathbf{c} =$ (*wow the wool from the, is, so, soft, and, fluffy*), where *kid* is the central word taken from between *the* and *fluffy*.

[1] As this part of our method is used with both the unsupervised senses and the lexical senses, referred to as $\mathbf{u}$ and $\mathbf{l}$ respectively in other parts of the paper, here we use a general sense $\mathbf{s}$ to avoid confusion.

For any particular sense, s_i, the multiple sense skip-gram language model can be used to find the probability of a word w_j occurring in the context: $P(w_j \mid s_i)$. By assuming the conditional independence of each word w_j in the context, given the sense embedding s_i, the probability of the context can be calculated:

$$P(\mathbf{c} \mid s_i) = \prod_{\forall w_j \in \mathbf{c}} P(w_j \mid s_i) \tag{2}$$

The correctness of the conditional independence assumption depends on the quality of the representation – the ideal sense representation would fully capture all information about the contexts it can appear in – thus the other contexts elements would not present any additional information, and so $P(w_a \mid w_b, s_i) = P(w_a \mid s_i)$. Given this, we have an estimate of $P(\mathbf{c} \mid s_i)$ which can be used to find $P(s_i \mid \mathbf{c})$. However, a false assumption of independence contributes towards overly sharp estimates of the posterior distribution [18], which we seek to address in Section 3.2 with geometric smoothing.

Bayes' Theorem is applied to this context likelihood function $P(\mathbf{c} \mid s_i)$ and a prior for the sense $P(s_i)$ to allow the posterior probability to be found:

$$P(s_i \mid \mathbf{c}) = \frac{P(\mathbf{c} \mid s_i) P(s_i)}{\sum_{s_j \in \mathbf{s}} P(\mathbf{c} \mid s_j) P(s_j)} \tag{3}$$

This is the probability of the sense given the context.

3.2 Geometric Smoothing for General WSD

During refitting, we note that often one induced sense would be calculated as having much higher probability of occurring than the others (according to Equation (3)). This level of certainty is not expected to occur in natural languages, ambiguity is almost always possible. To resolve such dominance problems, we propose a new *geometric smoothing* method. This is suitable for smoothing posterior probability estimates derived from products of conditionally independent likelihoods. It smooths the resulting distribution, by shifting all probabilities to be closer to the uniform distribution.

We hypothesize that the sharpness of probability estimates from Equation (3) is a result of data sparsity, and of a false independence assumption in Equation (2). This is well known to occur for n-gram language models [18]. Word-embeddings language models largely overcome the data sparsity problem due to weight sharing effects [17]. We suggest that the problem remains for word sense embeddings, where there are many more classes. Thus the training data must be split further between each sense than it was when split for each word. The power law distribution of word use [19] is compounded by word senses within those used also following the a power law distribution [20]. Rare senses are liable to over-fit to the few contexts they do occur in, and so give disproportionately high likelihoods to contexts that those are similar to. We propose to handle these issues through additional smoothing.

We consider replacing the unnormalised posterior with its n_c-th root, where n_c is the length of the context. We replace the likelihood of Equation (2) with

$P_S(\mathbf{c} \mid s_i) = \prod_{\forall w_j \in \mathbf{c}} \sqrt[n_c]{P(w_j \mid s_i)}$. Similarly, we replace the prior with: $P_S(s_i) = \sqrt[n_c]{P(w_j \mid s_i)}$ When this is substituted into Equation (3), it becomes a smoothed version of $P(s_i \mid \mathbf{c})$.

$$P_S(s_i \mid \mathbf{c}) = \frac{\sqrt[n_c]{P(\mathbf{c} \mid s_i) P(s_i)}}{\sum_{s_j \in \mathbf{s}} \sqrt[n_c]{P(\mathbf{c} \mid s_j) P(s_j)}} \tag{4}$$

The motivation for taking the n_c-th root comes from considering the case of the uniform prior. In this case $P_S(\mathbf{c} \mid s_i)$ is the geometric mean of the individual word probabilities $P_S(w_j \mid s_i)$. Consider, if one has two context sentences, $\mathbf{c} = \{w_1,...,w_{n_c}\}$ and $\mathbf{c}' = \{w'_1,...,w'_{n_{c'}}\}$, such that $n'_c > n'_c$ then using Equation (2) to calculate $P(\mathbf{c} \mid s_i)$ and $P(\mathbf{c}' \mid s_i)$ will result in incomparable results as additional number of probability terms will dominate – often significantly more than the relative values of the probabilities themselves. The number of words that can occur in the context of any given sense is very large – a large portion of the vocabulary. We would expect, averaging across all words, that each addition word in the context would decrease the probability by a factor of $\frac{1}{V}$, where V is the vocabulary size. The expected probabilities for $P(\mathbf{c} \mid s_i)$ is $\frac{1}{V^{n_c}}$ and for $P(\mathbf{c}' \mid s_i)$ is $\frac{1}{V^{n_{c'}}}$. As $n_{c'} > n_c$, thus we expect $P(\mathbf{c}' \mid s_i) \ll P(\mathbf{c} \mid s_i)$. Taking the n_c-th and $n_{c'}$-th roots of $P(\mathbf{c} \mid s_i)$ and $P(\mathbf{c} \mid s_i)$ normalises these probabilities so that they have the same expected value; thus making a context-length independent comparison possible. When this normalisation is applied to Equation (3), we get the smoothing effect.

4 Experimental Sense Embedding Models

We trained two sense embedding models, AdaGram [6] and our own Greedy Sense Embedding method. During training we use the Wikipedia dataset as used by Huang et al. [4]. However, we do not perform the extensive preprocessing used in that work.

Most of our evaluations are carried out on Adaptive SkipGrams (AdaGram) [6]. AdaGram is a non-parametric Bayesian extension of Skip-gram. It learns a number of different word senses, as are required to properly model the language.

We use the implementation[2] provided by the authors with minor adjustments for Julia [21] v0.5 compatibility.

The AdaGram model was configured to have up to 30 senses per word, where each sense is represented by a 100 dimension vector. The sense threshold was set to 10^{-10} to encourage many senses. Only words with at least 20 occurrences are kept, this gives a total vocabulary size of 497,537 words.

To confirm that our techniques are not merely a quirk of the AdaGram method or its implementation, we implemented a new simple baseline word sense embedding method. This method starts with a fixed number of randomly initialised embeddings, then greedily assigns each training case to the sense which predicts it with the highest probability (using Equation (3)). The task remains the same: using skip-grams with hierarchical softmax to predict the context words for the input

[2] `https://github.com/sbos/AdaGram.jl`

word sense. This is similar to [22], however it is using collocation probability, rather than distance in vector-space as the sense assignment measure. Our implementation is based on a heavily modified version of Word2Vec.jl[3].

This method is intrinsically worse than AdaGram. Nothing in the model encourages diversification and specialisation of the embeddings. Manual inspection reveals that a variety of senses are captured, though with significant repetition of common senses, and with rare senses being missed. Regardless of its low quality, it is a fully independent method from AdaGram, and so is suitable for our use in checking the generalisation of the refitting techniques.

The vocabulary used is smaller than for the AdaGram model. Words with at least 20,000 occurrences are allocated 20 senses. Words with at least 250 occurrences are restricted to a single sense. The remaining rare words are discarded. This results in a vocabulary size of 88,262, with 2,796 words having multiple senses. We always use a uniform prior, as the model does not facilitate easy calculation of the prior.

5 Similarity of Words in Context

Estimating word similarity with context is the task of determining how similar words are, when presented with the context they occur in. The goal of this task is to match human judgements of word similarity. For each of the target words and contexts; we use refitting on the target word to create a word sense embedding specialised for the meaning in the context provided. Then the similarity of the refitted vectors can be measured using cosine distance (or similar). By measuring similarity this way, we are defining a new similarity measure.

Reisinger and Mooney [3] define a number of measures for word similarity suitable for use with sense embeddings. The most successful was AvgSimC, which has become the gold standard method for use on similarity tasks. It has been used with great success in many works [4, 11, 5].

AvgSimC is defined using distance metric d (normally cosine distance) as:

$$\text{AvgSimC}((\mathbf{u},\mathbf{c}),(\mathbf{u}',\mathbf{c}')) = \frac{1}{n \times n'} \sum_{u_i \in \mathbf{u}} \sum_{u'_j \in \mathbf{u}'} P(u_i \mid \mathbf{c}) P(u'_j \mid \mathbf{c}') d(u_i, u'_j) \qquad (5)$$

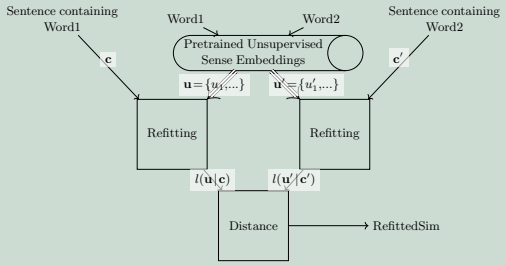
for contexts $\mathbf{c}$ and $\mathbf{c}'$, the contexts of the two words to be compared, and for $\mathbf{u} = \{u_1,...,u_n\}$ and $\mathbf{u}' = \{u'_1,...,u\prime_{n'}\}$ the respective sets of induced senses of the two words.

5.1 A New Similarity Measure: RefittedSim

We define a new similarity measure, RefittedSim, as the distance between the refitted sense embeddings. As shown in Figure 1 the example contexts are used to refit the induced sense embeddings of each word. This is a direct application of Equation (1).

[3] `https://github.com/tanmaykm/Word2Vec.jl/`

Fig. 1: Block diagram for RefittedSim similarity measure



Using the same definitions as in Equation (5), RefittedSim is defined as:

$$\text{RefittedSim}((\mathbf{u},\mathbf{c}),(\mathbf{u}',\mathbf{c}')) = d(l(\mathbf{u}\,|\,\mathbf{c}),l(\mathbf{u}'\,|\,\mathbf{c}') = d\Big(\textstyle\sum_{u_i\in\mathbf{u}} u_i P(u_i\,|\,\mathbf{c}),\sum_{u'_j\in\mathbf{u}'} u_i P(u'_j\,|\,\mathbf{c}')\Big) \tag{6}$$

AvgSimC is a probability weighted average of pairwise computed distances for each sense vector. Whereas RefittedSim is a single distance measured between the two refitted vectors – which are the probability weighted averages of the original unsupervised sense vectors.

There is a notable difference in time complexity between AvgSimC and RefittedSim. AvgSimC has time complexity $O(n\|\mathbf{c}\|+n'\|\mathbf{c}'\|+n\times n')$, while RefittedSim has $O(n\|\mathbf{c}\|+n'\|\mathbf{c}'\|)$. The product of the number of senses of each word $n\times n'$, may be small for dictionary senses, but it is often large for induced senses. Dictionaries tend to define only a few senses per word – the average[4] number of senses per word in WordNet is less than three [7]. For induced senses, however, it is often desirable to train many more senses, to get better results using the more fine-grained information. Reisinger and Mooney [3] found optimal results in several evaluations near 50 senses. In such cases the $O(n\times n')$ is significant, avoiding it with RefittedSim makes the similarity measure more useful for information retrieval.

5.2 Experimental Setup

We evaluate our refitting method using Stanford's Contextual Word Similarities (SCWS) dataset [4]. During evaluation, each context paragraph is limited to 5 words to either side of the target word, as in the training.

5.3 Results

Table 1a shows the results of our evaluations on the SCWS similarity task. A significant improvement can be seen by applying our techniques.

[4] It should be noted, though, that the number of meanings is not normally distributed [23].

Table 1: Spearman rank correlation $\rho \times 100$ when evaluated on the SCWS task.

(a) For varying hyper-parameters.

Method	Geometric Smoothing	Use Prior	AvgSimC	RefittedSim
AdaGram	T	T	**53.8**	64.8
AdaGram	T	F	36.1	**65.0**
AdaGram	F	T	43.8	47.8
AdaGram	F	F	20.7	24.1
Greedy	T	F	23.6	49.7
Greedy	F	F	22.2	40.7

(b) Compared to other methods . RefittedSim-S is with smoothing, and RefittedSim-SU is with uniform prior

Paper	Embedding	Similarity	$\rho \times 100$
This paper	AdaGram	AvgSimC	43.8
This paper	AdaGram	RefittedSim-S	64.8
This paper	AdaGram	RefittedSim-SU	65.0
[4]	Huang et al.	AvgSimC	65.7
[4]	Pruned tf-idf	AvgSimC	60.5
[11]	Chen et al.	AvgSimC	68.9
[5]	Tian et al.	AvgSimC	65.4
[5]	Tian et al.	MaxSim	**65.6**
[9]	SenseEmbed	Min Tanimoto	58.9
[9]	SenseEmbed	Weighted Tanimoto	62.4

The RefittedSim method consistently outperforms AvgSimC across all configurations. Similarly geometric smoothing consistently improves performance both for AvgSimC and for RefittedSim. The improvement is significantly more for RefittedSim than for AvgSimC results. In general using the unsupervised sense prior estimate from the AdaGram model, improves performance – particularly for AvgSimC. The exception to this is with RefittedSim with smoothing, where it makes very little difference. Unsurprisingly, given its low quality, the Greedy embeddings are always outperformed by AdaGram. It is not clear if these improvements will transfer to clustering based methods due to the differences in how the sense probability is estimated, compared to the language model based method evaluated on in Table 1a.

Table 1b compares our results with those reported in the literature using other methods. These results are not directly comparable, as each method uses a different training corpus, with different preprocessing steps, which can have significant effects on performance. It can been seen that by applying our techniques we bring the results of our AdaGram model from very poor ($\rho \times 100 = 43.8$) when using normal AvgSimC without smoothing, up to being competitive with other models, when using RefittedSim with smoothing. The method of Chen et al. [11], has a significant lead on the other results presented. This can be attributed to its very effective semi-supervised fine-tuning method. This suggests a possible avenue for future development in using refitted sense vectors to relabel a corpus, and then performing fine-tuning similar to that done by Chen et al.

6 Word Sense Disambiguation

6.1 Refitting for Word Sense Disambiguation

Once refitting has been used to create sense vectors for lexical word senses, an obvious used of them is to perform word sense disambiguation. In this section we refer to the lexical word sense disambiguation problem, i.e. to take a word and find its dictionary sense; whereas the methods discussed in Equations (3)

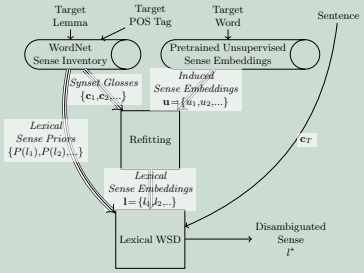


Fig. 2: Block diagram for performing WSD using refitting.

and (4) consider the more general problem, as applicable to disambiguating lexical or induced word senses depending on the inputs. Our overall process shown in Figure 2 uses both: first disambiguating the induced senses as part of refitting, then using the refitted sense vectors to find the most likely dictionary sense.

First, refitting is used to transform the induced sense vectors into lexical sense vectors. We use the targeted word's lemma (i.e. base form), and part of speech (POS) tag to retrieve all possible definitions of the word (Glosses) from WordNet; there is one gloss per sense. These glosses are used as the example sentence to perform refitting (see Section 3). We find embeddings, $\mathbf{l}=\{l_1,...,l_{n_l}\}$ for each of the lexical word senses using Equation (1). These lexical word senses are still supported by the language model, which means one can apply the general WSD method to determine the posterior probability of a word sense, given an observed context.

When given a sentence $\mathbf{c}_T$, containing a target word to be disambiguated, the probability of each lexical word sense $P(l_i \,|\, \mathbf{c}_T)$, can be found using Equation (3) (or the smoothed version Equation (4)), over the lexically refitted sense embeddings. Then, selecting the correct sense is simply selecting the most likely sense:

$$l^\star(\mathbf{l},\mathbf{c}_T) = \underset{\forall l_i \in \mathbf{l}}{\operatorname{argmax}} P(l_i|\mathbf{c}_T) = \underset{\forall l_i \in \mathbf{l}}{\operatorname{argmax}} \frac{P(\mathbf{c}_T \,|\, l_i) P(l_i)}{\sum_{\forall l_j \in \mathbf{l}} P(\mathbf{c}_T \,|\, l_j) P(l_j)} \tag{7}$$

6.2 Lexical Sense Prior

WordNet includes frequency counts for each word sense based on Semcor [24]. These form a prior for $P(l_i)$. The comparatively small size of Semcor means that many word senses do not occur at all. We apply add-one smoothing to remove any zero counts. This is in addition to using our proposed geometric smoothing as an optional part of the general WSD. Geometric smoothing serves a different (but related) purpose, of decreasing the sharpness of the likelihood function – not of removing impossibilities from the prior.

6.3 Experimental Setup

The WSD performance is evaluated on the SemEval 2007 Task 7.

We use the weighted mapping method of Agirre et al. [12], (see Section 2.2) as a baseline alternative method for using WSI senses for WSD. We use Semcor as the mapping corpus, to derive the mapping weights.

The second baseline we use is the Most Frequent Sense (MFS). This method always disambiguates any word as having its most common meaning. Due to the power law distribution of word senses, this is a very effective heuristic [20]. We also consider the results when using a backoff to MSF when a method is unable to determine the word sense the method can report the MFS instead of returning no result (a non-attempt).

6.4 Word Sense Disambiguation Results

Method	Attempted	Precision	Recall	F1
Refitted-S AdaGram	99.91%	0.799	0.799	**0.799**
Refitted AdaGram	99.91%	0.774	0.773	0.774
Refitted-S Greedy	79.95%	0.797	0.637	0.708
Refitted-S Greedy ∗	100.00%	0.793	0.793	0.793
Refitted Greedy	79.95%	0.725	0.580	0.645
Refitted Greedy ∗	100.00%	0.793	0.793	0.793
Mapped AdaGram	84.31%	0.776	0.654	0.710
Mapped AdaGram ∗	100.00%	0.736	0.736	0.736
MFS baseline	100.00%	0.789	0.789	0.789

Table 2: Results on SemEval 2007 Task 7 – course-all-words disambiguation. The -*S* marks results using geometric smoothing. The ∗ marks results with MSF backoff.

The results of employing our method for WSD , are shown in Table 2. Our results using smoothed refitting, both with AdaGram and Greed Embeddings with backoff, outperform the MSF baseline [25] – noted as a surprisingly hard baseline to beat [11].

The mapping method [12] was not up to the task of mapping unsupervised senses to supervised senses, on this large scale task. The Refitting method works better. Though refitting is only usable for language-model embedding WSI, the mapping method is suitable for all WSI systems.

While not directly comparable due to the difference in training data, we note that our Refitted results, are similar in performance, as measured by F1 score, to the results reported by Chen et al. [11]. AdaGram with smoothing, and Greedy embeddings with backoff have close to the same result as reported for L2R with backoff – with the AdaGram slightly better and the Greedy embeddings slightly worse. They are exceeded by the best method reported in that paper: S2C method with backoff. Comparison to non-embedding based methods is not discussed here for brevity. Historically state of the art systems have functioned very differently; normally by approaching the WSD task by more direct means.

Our results are not strong enough for Refitted AdaGram to be used as a WSD method on its own, but do demonstrate that the senses found by refitting are capturing the information from lexical senses. It is now evident that the refitted

sense embeddings are able to perform WSD, which was not possible with the unsupervised senses.

7 Conclusion

A new method is proposed for taking unsupervised word embeddings, and adapting them to align to particular given lexical senses, or user provided usage examples. This refitting method thus allows us to find word sense embeddings with known meaning. This method can be seen as a one-shot learning task, where only a single labelled example of each class is available for training. We show how our method can be used to create embeddings to evaluate the similarity of words, given their contexts.

This allows us to propose a new similarity measuring method, RefittedSim. The performance of RefittedSim on AdaGram is comparable to the results reported by the researchers of other sense embeddings techniques using AvgSimC, but its time complexity is significantly lower. We also demonstrate how similar refitting principles can be used to create a set of vectors that are aligned to the meanings in a sense inventory, such as WordNet.

We show how this can be used for word sense disambiguation. On this difficult task, it performs marginally better than the hard to beat MFS baseline, and significantly better than a general mapping method used for working with WSI senses on lexical WSD tasks. As part of our method for refitting, we present a geometric smoothing to overcome the issues of overly dominant senses probability estimates. We show that this significantly improves the performance. Our refitting method provides effective bridging between the vector space representation of meaning, and the traditional discrete lexical representation. More generally it allows a sense embedding to be created to model the meaning of a word in any given sentence. Significant applications of sense embeddings in tasks such as more accurate information retrieval thus become possible.

References

1. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv:1301.3781 (2013)
2. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014). (2014) 1532–1543
3. Reisinger, J., Mooney, R.J.: Multi-prototype vector-space models of word meaning. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Association for Computational Linguistics (2010) 109–117
4. Huang, E.H., Socher, R., Manning, C.D., Ng, A.Y.: 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, Association for Computational Linguistics (2012) 873–882
5. Tian, F., Dai, H., Bian, J., Gao, B., Zhang, R., Chen, E., Liu, T.Y.: A probabilistic model for learning multi-prototype word embeddings. In: COLING. (2014) 151–160

6. Bartunov, S., Kondrashkin, D., Osokin, A., Vetrov, D.P.: Breaking sticks and ambiguities with adaptive skip-gram. CoRR **abs/1502.07257** (2015)
7. Miller, G.A.: Wordnet: a lexical database for english. Communications of the ACM **38** (1995) 39–41
8. Véronis, J.: A study of polysemy judgements and inter-annotator agreement. In: Programme and advanced papers of the Senseval workshop. (1998) 2–4
9. Iacobacci, I., Pilehvar, M.T., Navigli, R.: Sensembed: learning sense embeddings for word and relational similarity. In: Proceedings of ACL. (2015) 95–105
10. Moro, A., Raganato, A., Navigli, R.: Entity Linking meets Word Sense Disambiguation: a Unified Approach. Transactions of the Association for Computational Linguistics (TACL) **2** (2014) 231–244
11. Chen, X., Liu, Z., Sun, M.: A unified model for word sense representation and disambiguation. In: EMNLP, Citeseer (2014) 1025–1035
12. Agirre, E., Martínez, D., De Lacalle, O.L., Soroa, A.: Evaluating and optimizing the parameters of an unsupervised graph-based wsd algorithm. In: Proceedings of the first workshop on graph based methods for natural language processing, Association for Computational Linguistics (2006) 89–96
13. Agirre, E., Soroa, A.: Semeval-2007 task 02: Evaluating word sense induction and discrimination systems. In: Proceedings of the 4th International Workshop on Semantic Evaluations. SemEval '07, Stroudsburg, PA, USA, Association for Computational Linguistics (2007) 7–12
14. Nocedal, J.: Updating quasi-newton matrices with limited storage. Mathematics of computation **35** (1980) 773–782
15. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems. (2013) 3111–3119
16. Mikolov, T., Yih, W.t., Zweig, G.: Linguistic regularities in continuous space word representations. In: HLT-NAACL. (2013) 746–751
17. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. The Journal of Machine Learning Research (2003) 137–186
18. Rosenfeld, R.: Two decades of statistical language modeling: Where do we go from here? Proceedings of the IEEE **88** (2000) 1270–1278
19. Zipf, G.: Human behavior and the principle of least effort: an introduction to human ecology. Addison-Wesley Press (1949)
20. Kilgarriff, A. In: How Dominant Is the Commonest Sense of a Word? Springer Berlin Heidelberg, Berlin, Heidelberg (2004) 103–111
21. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: A fresh approach to numerical computing. (2014)
22. Neelakantan, A., Shankar, J., Passos, A., McCallum, A.: Efficient non-parametric estimation of multiple embeddings per word in vector space. arXiv preprint arXiv:1504.06654 (2015)
23. Zipf, G.K.: The meaning-frequency relationship of words. The Journal of general psychology **33** (1945) 251–256
24. Tengi, R.I.: Design and implementation of the WordNet lexical database and searching software. In: WordNet: an electronic lexical database, The MIT Press, Cambridge, Massachusetts. (1998) 105
25. Navigli, R., Litkowski, K.C., Hargraves, O.: Semeval-2007 task 07: Coarse-grained english all-words task. In: Proceedings of the 4th International Workshop on Semantic Evaluations. SemEval '07, Stroudsburg, PA, USA, Association for Computational Linguistics (2007) 30–35

NovelPerspective

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

Abstract

We present NovelPerspective: a proof-of-concept tool to allow consumers to subset their digital literature, based on point of view (POV) character. Many novels have multiple main characters each with their own storyline running in parallel. A well-known example is George R. R. Martin's novel: "A Game of Thrones", and others from that series. Our tool detects the main character that each section is from the POV of, and allows the user to generate a new ebook with only those sections. This gives consumers new options in how they consume their media; allowing them to pursue the storylines sequentially, or skip chapters about characters they find boring. We present two heuristic-based baselines, and two machine learning based methods for the detection of the main character.

1 Introduction

Often each section of a novel is written from the perspective of a different main character. The characters each take turns in the spot-light, with their own parallel storylines being unfolded by the author. As readers, we have often desired to read just one storyline at a time, particularly when reading the book a second-time. In this paper, we present a tool, NovelPerspective, to give the consumer this choice.

Our tool allows the consumer to select which characters of the book they are interested in, and to generate a new ebook file containing just the sections from that character's point of view (POV). The critical part of this system is the detection of the POV character. This is not an insurmountable task, building upon the well established field of named entity recognition. However to our knowledge there is no software to do this. Such a tool would have been useless, in decades past when booked were distributed only on paper. But today, the surge in popularity of ebooks has opened a new niche for consumer narrative processing. Methods are being created to extract social relationships between characters (Elson et al., 2010; Wohlgenannt et al., 2016); to align scenes in movies with those from books (Zhu et al., 2015); and to otherwise augment the literature consumption experience. Tools such as the one presented here, give the reader new freedoms in controlling how they consume their media.

Having a large cast of characters, in particular POV characters, is a hallmark of the epic fantasy genre. Well known examples include: George R.R. Martin's "A Song of Ice and Fire", Robert Jordan's "Wheel of Time", Brandon Sanderson's "Cosmere" universe, and Steven Erikson's "Malazan Book of the Fallen", amongst thousands of others. Generally, these books are written in *limited* third-person POV; that is to say the reader has little or no more knowledge of the situation described than the main character does.

We focus here on novels written in the *limited* third-person POV. In these stories, the main character is, for our purposes, the POV character. Limited third-person POV is written in the third-person, that is to say the character is referred to by name, but with the observations limited to being from the perspective of that character. This is in-contrast to the *omniscient* third-person POV, where events are described by an external narrator. Limited third-person POV is extremely popular in modern fiction. It preserves the advantages of first-person, in allowing the reader to observe inside the head of the character, while also allowing the flexibility to the perspective of another character (Booth, 1961). This allows for multiple concurrent storylines around different characters.

Our tool helps users un-entwine such storylines, giving the option to process them sequentially.

The utility of dividing a book in this way varies with the book in question. Some books will cease to make sense when the core storyline crosses over different characters. Other novels, particularly in epic fantasy genre, have parallel storylines which only rarely intersect. While we are unable to find a formal study on this, anecdotally many readers speak of:

- "Skipping the chapters about the boring characters."
- "Only reading the *real* main character's sections."
- "Reading ahead, past the side-stories, to get on with the *main* plot."

Particularly if they have read the story before, and thus do not risk confusion. Such opinions are a matter of the consumer's personal taste. The NovelPerspective tool gives the reader the option to customise the book in this way, according to their personal preference.

We note that sub-setting the novel once does not prevent the reader from going back and reading the intervening chapters if it ceases to make sense, or from sub-setting again to get the chapters for another character whose path intersects with the storyline they are currently reading. We can personally attest for some books reading the chapters one character at a time is indeed possible, and pleasant: the first author of this paper read George R.R. Martin's "A Song of Ice and Fire" series in exactly this fashion.

The primary difficulty in segmenting ebooks this way is attributing each section to its POV character. That is to say detecting who is the point of view character. Very few books indicate this clearly, and the reader is expected to infer it during reading. This is easy for most humans, but automating it is a challenge. To solve this, the core of our tool is its character classification system. We investigated several options which the main text of this paper will discuss.

2 Character Classification Systems

The full NovelPerspective pipeline is shown in Figure 1. The core character classification step (step 3), is detailed in Figure 2. In this step the raw text is first enriched with parts of speech, and named entity tags. From this, features are extracted for each named entity. These feature vectors are used to score the entities for the most-likely POV character. The highest scoring character is returned by the system. The different systems presented modify the **Feature Extraction** and **Character Scoring** steps. A broadly similar idea, for detecting the focus location of news articles, was presented by (Imani et al., 2017).

2.1 Baseline systems

To the best of our knowledge no systems have been developed for this task before. As such, we have developed two deterministic baseline character classifiers. These are both potentially useful to the end-user in our deployed system (Section 5), and used to gauge the performance of the more complicated systems in the evaluations presented in Section 4.

It should be noted that the baseline systems, while not using machine learning for the character classification steps, do make extensive use of machine learning-based systems during the pre-processing stages.

2.1.1 "First Mentioned" Entity

An obvious way to determine the main character of the section is to select the first named entity. We use this to define the "First Mentioned" baseline In this system, the **Feature Extraction** step is simply retrieving the position of the first use of each name; and the **Character Scoring** step assigns each a score such that earlier is higher. This works for many examples: *"One dark and stormy night, Bill heard a knock at the door."*; however it fails for many others *" 'Is that Tom?' called out Bill, after hearing a knock."*. Sometimes a section may go several paragraphs describing events before it even mentions the character who is perceiving them. This is a varying element of style.

2.1.2 "Most Mentioned" Entity

A more robust method to determine the main character, is to use the occurrence counts. We call this the "Most Mentioned" baseline. The **Feature Extraction** step is to count how often the name is used. The **Character Scoring** step assigns each a score based what proportional of all names used were for this entity. This works well for many books. The more important a character is, the more often their name occurs. However, it is fooled, for example, by book chapters that

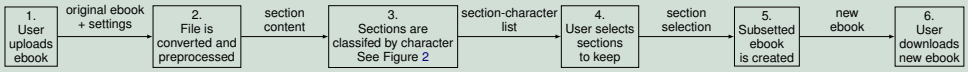


Figure 1: The full NovelPerspective pipeline. Note that step 5 uses the original ebook to subset.

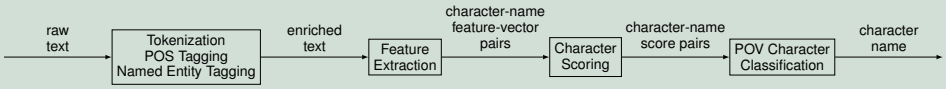


Figure 2: The general structure of the character classification systems. This repeated for each section of the book during step 3 of the full pipeline shown in Figure 1.

are about the POV character's relationship with a secondary character. In such cases the secondary character may be mentioned more often.

2.2 Machine learning systems

One can see the determination of the main character as a multi-class classification problem. From the set of all named entities in the section, classify that section as to which one is the main character. Unlike typical multi-class classification problems the set of possible classes varies per section being classified. Further, even the total set of possible named characters, i.e. classes, varies from book to book. An information extraction approach is required which can handle these varying classes. As such, a machine learning model for this task can not incorporate direct knowledge of the classes (i.e. character names).

We reconsider the problem as a series of binary predictions. The task is to predict if a given named entity is the point of view character. For each possible character (i.e. each named-entity that occurs), a feature vector is extracted (see Section 2.2.1). This feature vector is the input to a binary classifier, which determines the probability that it represents the main character. The **Character Scoring** step is thus the running of the binary classifier: the score is the output probability normalised over all the named entities.

2.2.1 Feature Extraction for ML

We investigated two feature sets as inputs for our machine learning-based solution. They correspond to different **Feature Extraction** steps in Figure 2. A hand-engineered feature set, that we call the "Classical" feature set; and a more modern "Word Embedding" feature set. Both feature sets give information about how the each named entity token was used in the text.

The "Classical" feature set uses features that are well established in NLP related tasks. The features can be described as *positional features*, like in the First Mentioned baseline; *occurrence count features*, like in the *Most Mentioned* baseline and *adjacent POS counts*, to give usage context. The *positional features* are the index (in the token counts) of the first and last occurrence of the named entity. The *occurrence count features* are simply the number of occurrences of the named entity, supplemented with its rank on that count compared to the others. The *adjacent POS counts* are the occurrence counts of each of the 46 POS tags on the word prior to the named entity, and on the word after. We theorised that this POS information would be informative, as it seemed reasonable that the POV character would be described as doing more things, so co-occurring with more verbs. This gives 100 base features. To allow for text length invariance we also provide each of the base features expressed as a portion of its maximum possible value (e.g. for a given POS tag occurring before a named entity, the potion of times this tag occurred). This gives a total of 200 features.

The "Word Embedding" feature set uses FastText word vectors (Bojanowski et al., 2017). We use the pretrained 300 dimensional embeddings trained on English Wikipedia [1]. We concatenate the 300 dimensional word embedding for the word immediately prior to, and immediately after each occurrence of a named entity; and take the element-wise mean of this concatenated vector over all occurrences of the entity. Such averages of word embeddings have been shown to be a useful feature in many tasks (White et al., 2015; Mikolov et al., 2013). This has a total of 600 features.

[1] https://fasttext.cc/docs/en/pretrained-vectors.html

2.2.2 Classifier

The binary classifier, that predicts if a named entity is the main character, is the key part of the **Character Scoring** step for the machine learning systems. From each text in the training dataset we generated a training example for every named entity that occurred. All but one of these was a negative example. We then trained it as per normal for a binary classifier. The score for a character is the classifier's predicted probability of its feature vector being for the main character.

Our approach of using a binary classifier to rate each possible class, may seem similar to the one-vs-rest approach for multi-class classification. However, there is an important difference. Our system only uses a single binary classifier; not one classifier per class, as the classes in our case vary with every item to be classified. The fundamental problem is information extraction, and the classifier is a tool for the scoring which is the correct information to report.

With the classical feature set we use logistic regression, with the features being preprocessed with 0-1 scaling. During preliminary testing we found that many classifiers had similar high degree of success, and so chose the simplest. With the word embedding feature set we used a radial bias support vector machine, with standardisation during preprocessing, as has been commonly used with word embeddings on other tasks.

3 Experimental Setup

3.1 Datasets

We make use of three series of books selected from our own personal collections. The first four books of George R. R. Martin's "A Song of Ice and Fire" series (hereafter referred to as ASOIAF); The two books of Leigh Bardugo's "Six of Crows" duology (hereafter referred to as SOC); and the first 9 volumes of Robert Jordan's "Wheel of Time" series (hereafter referred to as WOT). In Section 4 we consider the use of each as a training and testing dataset. In the online demonstration (Section 5), we deploy models trained on the combined total of all the datasets.

To use a book for the training and evaluation of our system, we require a ground truth for each section's POV character. ASOIAF and SOC provide ground truth for the main character in the chapter names. Every chapter only uses the POV of that named character. WOT's ground truth comes from an index created by readers.[2] We do not have any datasets with labelled sub-chapter sections, though the tool does support such works.

Dataset	Chapters	POV Characters
ASOIAF	256	15
SOC	91	9
WOT	432	52
combined	779	76

Table 1: The number of chapters and point of view characters for each dataset.

The total counts of chapters and characters in the datasets, after preprocessing, is shown in Table 1. Preprocessing consisted of discarding chapters for which the POV character was not identified (e.g. prologues); and of removing the character names from the chapter titles as required.

3.2 Evaluation Details

In the evaluation, the systems are given the body text and asked to predict the character names. During evaluation, we sum the scores of the characters alternative aliases/nick-names used in the books. For example merging `Ned` into `Eddard` in ASOIAF. This roughly corresponds to the case that a normal user can enter multiple aliases into our application when selecting sections to keep. We do not use these aliases during training, though that is an option that could be investigated in a future work.

3.3 Implementation

The implementation is available on GitHub. [3] Scikit-Learn (Pedregosa et al., 2011) is used for the machine learning and evaluations, and NLTK (Bird and Loper, 2004) is used for textual preprocessing. The text is tokenised, and tagged with POS and named entities using NLTK's default methods. Specifically, these are the Punkt sentence tokenizer, the regex-based improved Tree-Bank word tokenizer, greedy averaged perceptron POS tagger, and the max-entropy binary named entity chunker. The use of a binary, rather than a multi-class, named entity chunker is significant. Fantasy novels often use "exotic" names for characters, we found that this often resulted in charac-

[2] http://wot.wikia.com/wiki/List_of_Point_of_View_Characters

[3] https://github.com/oxinabox/NovelPerspective/

Test Set	Method	Train Set	Acc
ASOIAF	First Mentioned	—	0.250
ASOIAF	Most Mentioned	—	0.914
ASOIAF	ML Classical Features	SOC	0.953
ASOIAF	ML Classical Features	WOT	**0.984**
ASOIAF	ML Classical Features	WOT+SOC	0.977
ASOIAF	ML Word Emb. Features	SOC	0.863
ASOIAF	ML Word Emb. Features	WOT	0.977
ASOIAF	ML Word Emb. Features	WOT+SOC	0.973
SOC	First Mentioned	—	0.429
SOC	Most Mentioned	—	0.791
SOC	ML Classical Features	WOT	0.923
SOC	ML Classical Features	ASOIAF	0.923
SOC	ML Classical Features	WOT+ASOIAF	0.934
SOC	ML Word Emb. Features	WOT	0.934
SOC	ML Word Emb. Features	ASOIAF	**0.945**
SOC	ML Word Emb. Features	WOT+ASOIAF	**0.945**
WOT	First Mentioned	—	0.044
WOT	Most Mentioned	—	0.660
WOT	ML Classical Features	SOC	0.701
WOT	ML Classical Features	ASOIAF	**0.745**
WOT	ML Classical Features	ASOIAF+SOC	0.736
WOT	ML Word Emb. Features	SOC	0.551
WOT	ML Word Emb. Features	ASOIAF	0.699
WOT	ML Word Emb. Features	ASOIAF+SOC	0.681

Table 2: The results of the character classifier systems. The best results are **bolded**.

ter named entities being misclassified as organisations or places. Note that this is particularly disadvantageous to the First Mentioned baseline, as any kind of named entity will steal the place. Nevertheless, it is required to ensure that all character names are a possibility to be selected.

4 Results and Discussion

Our evaluation results are shown in Table 2 for all methods. This includes the two baseline methods, and the machine learning methods with the different feature sets. We evaluate the machine learning methods using each dataset as a test set, and using each of the other two and their combination as the training set.

The First Mentioned baseline is very weak. The Most Mentioned baseline is much stronger. In almost all cases machine learning methods outperform both baselines. The results of the machine learning method on the ASOIAF and SOC are very strong. The results for WOT are weaker, though they are still accurate enough to be useful when combined with manual checking.

It is surprising that using the combination of two training sets does not always out-perform each on their own. For many methods training on just one dataset resulted in better results. We believe

Test Set	Method	Train Set	Acc
ASOIAF	ML Classical Features	ASOIAF	0.980
ASOIAF	ML Word Emb. Features	ASOIAF	0.988
SOC	ML Classical Features	SOC	0.945
SOC	ML Word Emb. Features	SOC	0.956
WOT	ML Classical Features	WOT	0.785
WOT	ML Word Emb. Features	WOT	0.794

Table 3: The training set accuracy of the machine learning character classifier systems.

that the difference between the top result for a method and the result using the combined training sets is too small to be meaningful. It can, perhaps, be attributed to a coincidental small similarity in writing style of one of the training books to the testing book. To maximise the generalisability of the NovelPerspective prototype (see Section 5), we deploy models trained on all three datasets combined.

Almost all the machine learning models resulted in similarly high accuracy. The exception to this is word embedding features based model trained on SOC, which for both ASOIAF and WOT test sets performed much worse. We attribute the poor performance of these models to the small amount of training data. SOC has only 91 chapters to generate its training cases from, and the word embedding feature set has 600 dimensions. It is thus very easily to over-fit which causes these poor results.

Table 3 shows the training set accuracy of each machine learning model. This is a rough upper bound for the possible performance of these models on each test set, as imposed by the classifier and the feature set. The WOT bound is much lower than the other two texts. This likely relates to WOT being written in a style that closer to the line between third-person *omniscient*, than the more clear third-person *limited* POV of the other texts. We believe longer range features are required to improve the results for WOT. However, as this achieves such high accuracy for the other texts, further features would not improve accuracy significantly, without additional more difficult training data (and would likely cause overfitting).

The results do not show a clear advantage to either machine learning feature set. Both the classical features and the word embeddings work well. It seems that the classical feature are more robust; both with smaller training sets (like SOC),

and with more difficult test sets (like WOT). With more training data, it would be reasonable to more features (e.g. additional adjacent words), by doing this it may be that word embedding may prove better.

5 Demonstration System

We have deployed an online prototype to https://white.ucc.asn.au/tools/np. A video of its use can be found at https://youtu.be/iu41pUF4wTY. This web-app, made using the CherryPy framework,[4] allows the user to apply any of the model discussed to their own novels.

The web-app functions as shown in Figure 1. The user uploads an ebook, and selects one of the character classification systems that we have discussed above. They are then presented with a page displaying a list of sections, with the predicted main character(/s) paired with an excerpt from the beginning of the section. The user can adjust to show the top-k most-likely characters on this screen, to allow for additional recall.

The user can select sections to retain. They can use a regular expression to match the character names(/s) they are interested in. The sections with matching predicted character names will be selected. As none of the models is perfect, some mistakes are likely. The user can manually correct the selection before downloading the book.

6 Conclusion

We have presented a tool to allow consumers to restructure their ebooks around the characters they find most interesting. The system must discover the named entities that are present in each section of the book, and then classify each section as to which character's point of view the section is narrated from. For named entity detection we make use of standard tools. However, the classification is non-trivial. In this design we implemented several systems. Simply selecting the most commonly named character proved successful as a baseline approach. To improve upon this, we developed several machine learning based approaches which perform very well. While none of the classifiers are perfect, they achieve high enough accuracy to be useful.

A future version of our application will allow the users to submit corrections, giving us more training data. However, storing this information poses copyright issues that are yet to be resolved.

[4] http://cherrypy.org/

References

Bird, S. and Loper, E. (2004). Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Booth, W. C. (1961). *The rhetoric of fiction*. University of Chicago Press.

Elson, D. K., Dames, N., and McKeown, K. R. (2010). Extracting social networks from literary fiction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 138–147, Stroudsburg, PA, USA. Association for Computational Linguistics.

Imani, M. B., Chandra, S., Ma, S., Khan, L., and Thuraisingham, B. (2017). Focus location extraction from political news reports with bias correction. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1956–1964.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

White, L., Togneri, R., Liu, W., and Bennamoun, M. (2015). How well sentence embeddings capture meaning. In *Proceedings of the 20th Australasian Document Computing Symposium*, ADCS '15, pages 9:1–9:8. ACM.

Wohlgenannt, G., Chernyak, E., and Ilvovsky, D. (2016). Extracting social networks from literary text with word embedding tools. In *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH)*, pages 18–25.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

NovelPerspective: Supplementary Materials

Anonymous ACL submission

Shown here are screen-shots of NovelPerspective web-app. They are taken from the online prototype; which is available at https://white.ucc.asn.au/tools/np. In this example we have used Brandon Sanderson's "The Way of Kings". A video showing this same content can be found at https://youtu.be/iu41pUF4wTY.

They show the screens the used goes through The user uploads their book and sets the configuration options as shown in Figure 1. Then the preprocessing is done as shown in Figure 2. The user interface for selecting which sections to keep (based on the POV characters) is shown in Figures 3 to 5.

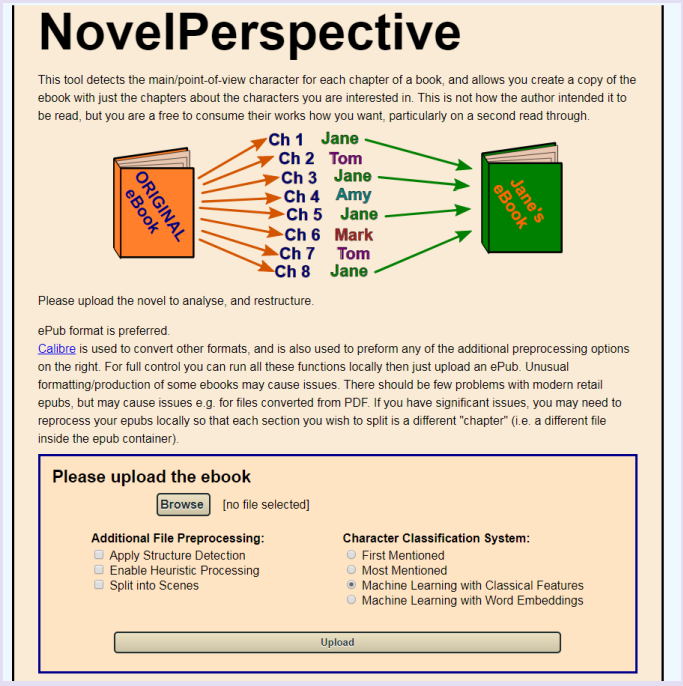


Figure 1: The start page for the NovelPerspectiveApp. Here the users uploads their book, and selects the POV character detection system, and any preprocessing options.

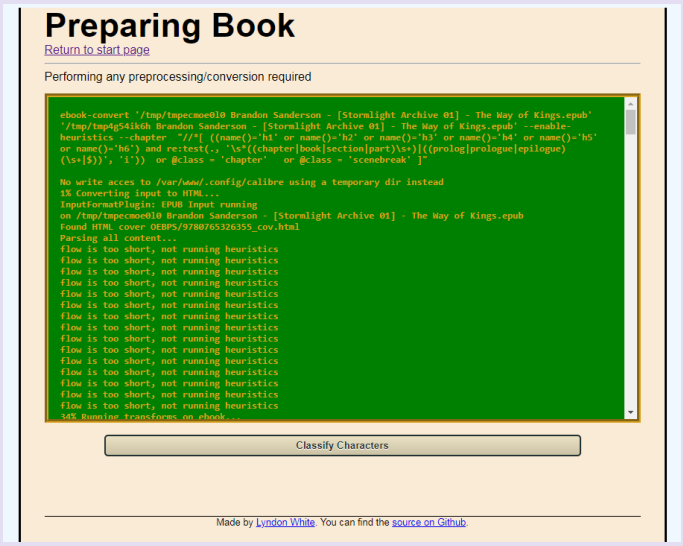


Figure 2: The preparation page. Here the Calibre ebook-conversion tool [2] is uses to convert the input into an epub, and to perform any requested preprocessing. The output of that tool is streamed to the user.

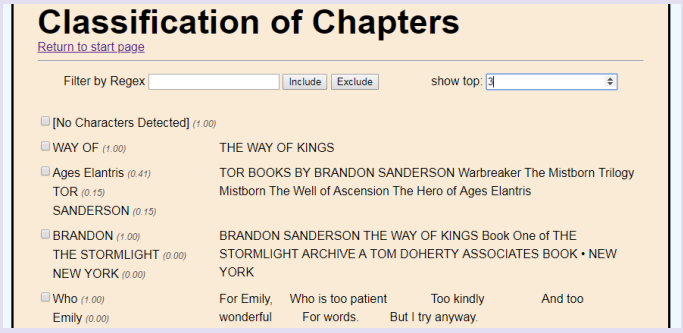


Figure 3: The section selection controls. The first few sections are preamble such as the title page and the dedication. Shown at the top are the show-top, and the regex selection controls. **On the right:** The user can use the show-top control to change the number of possible main characters to display. This is by default one for just the most likely POV character. For demonstration here we have set it to show the top three most likely. **On the left:** The user can use regex patterns for character names to include or exclude the selection of sections. They are matched against the currently displayed names. By alternating the manipulation of show-top, and the regex selection controls, the user can achieve a fine degree of control over their selection.

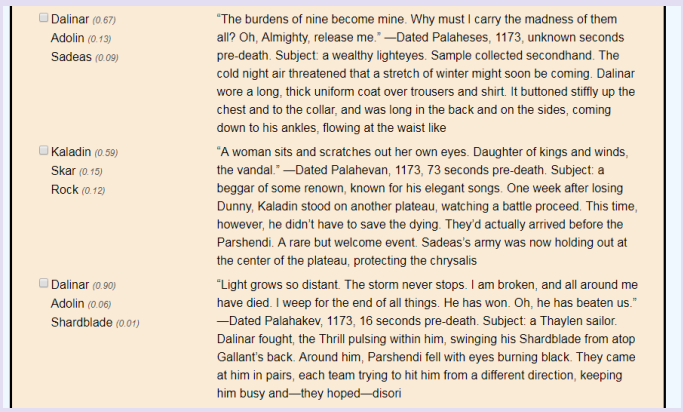


Figure 4: The main selection selection interface showing the sections that can be selected. This screenshot is taken from the same page as Figure 3, but from the middle of the page. Each row corresponds to a particular section of the book. On the left is shown the names of the characters who are most likely to be the POV character. The numbers of characters shown is determined by the show-top control. In smaller writing next to each is the characters score as determining by the classification method. On the right is shown the start of the section

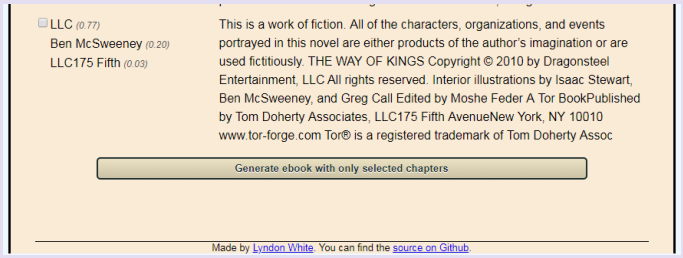


Figure 5: Generating a book from the main section selection interface. This screen-shot is taken from the same page as Figure 3, but from the end of the page. Once the user has made their selections, they press the Generate ebook button shown to create and download an ebook with only their selected content.

Generating Bags of Words from the Sums of their Word Embeddings

Lyndon White, Roberto Togneri, Wei Liu, and Mohammed Bennamoun

The University of Western Australia
35 Stirling Highway, Crawley, Western Australia
lyndon.white@research.uwa.edu.au, roberto.togneri@uwa.edu.au,
wei.liu@uwa.edu.au, mohammed.bennamoun@uwa.edu.au

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 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. Dinu and Baroni (2014) 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 (Farhadi et al. 2010; Socher et al. 2014). 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 – 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 (Kågebäck et al. 2014; Yogatama et al. 2015), 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, note that it takes as input only a word embedding vocabulary ($\mathcal{V}$) and the vector ($\tilde{s}$) to generate the BOW ($\tilde{c}$).

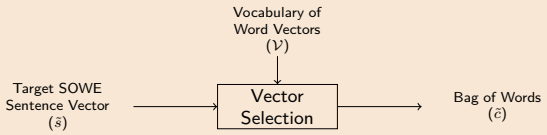


Fig. 1: The process for the regenerating BOW from SOWE sentence embeddings.

The rest of the paper is organized into the following sections. Section 2 introduces the area, discussing in general sentence models, and prior work on generation. Section 3 explains the problem in detail and our algorithm for solving it. Section 4 described the settings used for evaluation. Section 5 discusses the results of this evaluation. The paper presents its conclusions in Section 6, including a discussion of future work.

2 Background

The current state of the art for full sentence generation from sentence embeddings are the works of Iyyer et al. 2014 and Bowman et al. 2015. Both these advance beyond the earlier work of Dinu and Baroni 2014 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 (Yin and Schütze 2015). Two of the more notable are the SkipGrams of Mikolov et al. (2013a,b) and the Global Vector word representations (GloVe) of Pennington et al. (2014). 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 Le and Mikolov (2014) and Socher (2014). 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.

White et al. (2015) 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.

Ritter et al. (2015) 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.

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.

Definition 1. *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.

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 (Karp 1972). 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]

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.

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.

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,

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

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
    t̃ ⟵ Σ_{x_i ∈ bag_c} x_i
    while true do
        x̃* ⟵ argmin_{x_j ∈ V} d(s̃, t̃ + x̃_j)        /* exhaustive search of V */
        if d(s̃, t̃ + x̃*) < d(s̃, t̃) then
            t̃ ⟵ t̃ + x̃*  bag_c ⟵ bag_c ∪ {x̃*}
        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.

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.

4 Experimental Setup and Evaluations

4.1 Word Embeddings

GloVe representations of words (Pennington et al. 2014) 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 Mikolov et al. (2013a) suggested similar performance.

4.2 Corpora

The evaluation was performed on the Brown Corpus (Francis and Kucera 1979) and on a subset of the Books Corpus (Zhu et al. 2015). The Brown Corpus was sourced with samples from a 500 fictional and non-fictional works from 1961. The

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

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 (Nation 2006). These corpora remain sufficiently large and complex to quantitatively evaluate the algorithm.

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 (Bezanson et al. 2014), 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.

5 Results and Discussion

Table 1 shows examples of the output. Eight sentences which were used for demonstration of sentence generation in Bowman et al. (2015) and Iyyer et al.

[3] The same is true for any number of repetitions of the word *buffalo* – each of which forms a valid sentence as noted in Tymoczko et al. (1995)

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

Table 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 Iyyer et al. (2014), Examples f-h from Bowman et al. (2015). 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 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

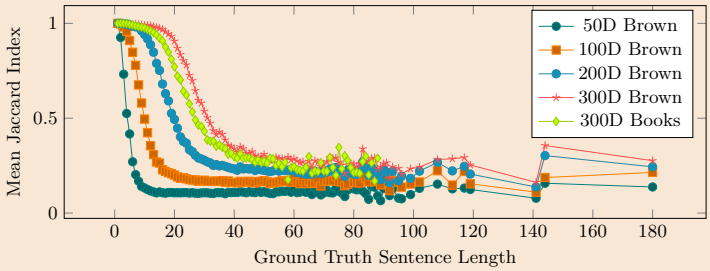


Fig. 2: The mean Jaccard index achieved during the word selection step, shown against the ground truth length of the sentence. Note that the vast majority of sentences are in the far left end of the plot. The diminishing samples are also the cause of the roughness, as the sentence length increases.

(2014) 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, 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 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 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 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 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.

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).

References

Bezanson, Jeff et al. (2014). "Julia: A Fresh Approach to Numerical Computing". In: arXiv: `1411.1607 [cs.MS]`.

Bowman, Samuel R et al. (2015). "Generating Sentences from a Continuous Space". In: *arXiv preprint arXiv:1511.06349*.

Dinu, Georgiana and Marco Baroni (2014). "How to make words with vectors: Phrase generation in distributional semantics". In: *Proceedings of ACL*, pp. 624–633.

Farhadi, Ali et al. (2010). "Every picture tells a story: Generating sentences from images". In: *Computer Vision–ECCV 2010*. Springer, pp. 15–29.

Francis, W Nelson and Henry Kucera (1979). "Brown corpus manual". In: *Brown University*.

Iyyer, Mohit, Jordan Boyd-Graber, and Hal Daumé III (2014). "Generating Sentences from Semantic Vector Space Representations". In: *NIPS Workshop on Learning Semantics*.

Kågebäck, Mikael et al. (2014). "Extractive summarization using continuous vector space models". In: *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)@ EACL*, pp. 31–39.

Karp, Richard M (1972). *Reducibility among combinatorial problems*. Springer.

Le, Quoc and Tomas Mikolov (2014). "Distributed Representations of Sentences and Documents". In: *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1188–1196.

Mikolov, Tomas et al. (2013a). "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781*.

Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig (2013b). "Linguistic Regularities in Continuous Space Word Representations." In: *HLT-NAACL*, pp. 746–751.

Nation, I (2006). "How large a vocabulary is needed for reading and listening?" In: *Canadian Modern Language Review* 63.1, pp. 59–82.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pp. 1532–1543.

Ritter, Samuel et al. (2015). "Leveraging Preposition Ambiguity to Assess Compositional Distributional Models of Semantics". In: *The Fourth Joint Conference on Lexical and Computational Semantics*.

Socher, Richard (2014). "Recursive Deep Learning for Natural Language Processing and Computer Vision". PhD thesis. Stanford University.

Socher, Richard et al. (2014). "Grounded compositional semantics for finding and describing images with sentences". In: *Transactions of the Association for Computational Linguistics* 2, pp. 207–218.

Tymoczko, T., J. Henle, and J.M. Henle (1995). *Sweet Reason: A Field Guide to Modern Logic*. Textbooks in Mathematical Sciences. Key College. ISBN: 9780387989303.

White, Lyndon et al. (2015). "How Well Sentence Embeddings Capture Meaning". In: *Proceedings of the 20th Australasian Document Computing Symposium*. ADCS '15. Parramatta, NSW, Australia: ACM, 9:1–9:8. ISBN: 978-1-4503-4040-3. DOI: 10.1145/2838931.2838932.

Yin, Wenpeng and Hinrich Schütze (2015). "Learning Word Meta-Embeddings by Using Ensembles of Embedding Sets". In: eprint: 1508.04257.

Yogatama, Dani, Fei Liu, and Noah A Smith (2015). "Extractive Summarization by Maximizing Semantic Volume". In: *Conference on Empirical Methods in Natural Language Processing*.

Zhu, Yukun et al. (2015). "Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books". In: *arXiv preprint arXiv:1506.06724*.

THE VECTOR SELECTION PROBLEM

LYNDON WHITE, WEI LIU

Definition 1. 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 vector $\tilde{s}$, $\tilde{s} \in \mathbb{R}^n$
- any metric d

  by

$$\underset{\{\forall \tilde{c} \in \mathbb{N}_0^V\}}{\operatorname{argmin}} \; d(\tilde{s}, \sum_{j=1}^{j=V} \tilde{x}_j c_j)$$

- V is size of the vocabulary $\mathcal{V}$. (~1300 for ATIS2, ~50,000 for Brown, ~10,000 for daily English)
- $\tilde{x}_j$ is the vector embedding for the jth word in the vocabulary $\tilde{x}_j \in \mathcal{V}$
  - We can express the embedding vocabulary $\mathcal{V} = \{\tilde{x}_j \,|\, \forall j \in \mathbb{N} \wedge 1 \leq j \leq V\} \subset \mathbb{R}^n$
  - If we treat $\mathcal{V}$ as a matrix with vectors $\tilde{x}_j$ for rows,(and treat length 1 vectors as scalars) we get the compact notation

$$\underset{\{\forall \tilde{c} \in \mathbb{N}_0^J\}}{\operatorname{argmin}} \; d(\tilde{s}, \sum_{j=1}^{j=V} \tilde{x}_j c_j) = \underset{\{\forall \tilde{c} \in \mathbb{N}_0^J\}}{\operatorname{argmin}} \; d(\tilde{s}, \mathcal{V}\,\tilde{c}^T)$$

- c_j is the count of how many times the jth word in the vocabulary occurs. $\tilde{c} \in \mathbb{N}_0^V$, so $c_j \in \mathbb{N}_0$
- n is the dimensional of the word vectors, $n = 300$ in current trials.

0.1. **An Analogy for the problem.** (which may or may not help)

Imaging you are in a tile shop which as a variety of rectangular (2D) tiles. They have many copies of each tile (an unlimited number in-fact), but only a finite number of different sizes. This tiles have connectors on them, like jigsaw pieces, such that you can attach a North/South side to another North/South side even on a tile of different size, and similar for the East/West sides. But you can't attach a North/South side to a East/West side. i.e. You can not rotated the tiles.

You have 2 lengths given as your target when choosing tiles: a North/South length, and a East/West length.

Your task is to select a collection of tiles from the store, such that when connected on the north/south and east/west the total length in those directions is as close as possible to those to targets.

A formula is given for how your solution will be judged. It takes the form of some distance metric. E.g it might be the your distance from the target east/west length with your connected tiles, plus your distance from the target north/south length. Or maybe accuracy on north/south is twice as important as east/west. Or north/south difference squared etc. Your task it to minimize that score.

For example, if the store had 3 types of tiles. 4.1×4.1,1.5×5.0 and 100×1, and your targets were 13.1 and 20.0, and the scoring was Manhattan.

you might choose one 4.1×4.1 tiles and three 1.5×5.0 tiles, giving you length totals 8.6 and 19.1 and a score of 5.4

Had you chosen to take an extra 4.1×4.1 tile though given totals of 12.7 and 23.2 giving a better score of 4.5

Now generalize it from 2D tiles to hyperblocks of some arbitrary dimensionality.

0.2. **Reduction from Subset sum.** The subset sum problem is well known to be NP-complete. First shown in by Karp under the name "Knapsack"[1] which has since come to be used for the more general problem.

It can be defined with the question: for a given set $\mathcal{S} \subset \mathbb{Z}$, does there exists $\mathcal{L} \subseteq \mathcal{S}$ such that $\sum_{l_i \in \mathcal{L}} l_i = 0$?

We reduce from subset sum to the Vector Selection Problem by showing any general solution to the Vector Selection Problem could be used to solve subset sum with only linear time additional work.

Claim 2. Any method which can solve the Vector Selection Problem will allow Subset sum to be completed with only linear time additional operations

- Let $\mathcal{S} = \{w_1, w_2, ..., w_m\}$
- Let $\Omega = 2m \left(\max_{i \in [1,m]} |w_i| + 1\right)$ and thus larger than the largest possible sum of elements of $\mathcal{S}$.

1

  - Finding this is a linear time operation, the only such operation in this method.
- Let $\omega = \frac{1}{2m}$ and thus smaller than any element of $\mathcal{S}$, except if $0 \in \mathcal{S}$ (in which case the solution is trivial)
- then we can define an embedding vocabulary $\mathcal{V}_s$ from based on $\mathcal{S}$ by

$$\mathcal{V}_s = \{[[w_i,\, 1]\,;\, \hat{e}_i] \,:\, w_i \in \mathcal{S}\}$$

  - By imposing some arbitrary total ordering on $\mathcal{S}$.
  - where ; is the concatenation operator,
  - and $\hat{e}_i$ is the elementary basis unit vector for dimension i. ie a vector with all zeros, except at index i, where it is 1.
  - i.e. we take the image of $\mathcal{S}$ into $\mathcal{V}_S$, by the function:

$$w_i \mapsto \begin{bmatrix} w_i \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \hat{e}_{i+2}$$

  - In doing so we map each integer w_i in $\mathcal{S}$ to a point in $\mathbb{R}^{m+1}$ where
    * the first index is the integer, w_i,
    * the second a term is used to force a solution that is nonempty to be better than an empty solution – all other things being equal;
    * the remaining m terms are used to force a solution which uses the same element more than once to be worse than one which uses it once or zero times.
  - Note that $\mathcal{V}_S \subset \mathbb{R}^{m+2}$
- we define the target vector by $\tilde{s_s} = \left[[0, m]\,;\, 0.5 \sum_{j=1}^{j=m} \hat{e}_j\right] = \begin{pmatrix} 0 \\ m \\ 0.5 \\ \vdots \\ 0.5 \end{pmatrix}$
- we define the distance metric being given by a weighed Manhattan distance (i.e. weighted L1 Norm).

$$d_s\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}\right) = |x_1 - y_1| + \omega\,|x_2 - y_2| + \Omega \sum_{j=3}^{j=n} |x_j - y_j|$$

  - We will prove that d_s is a metric below.
- The procedure for using these once defined to solve subset sum is:
  - the Vector Selection Problem for $(\mathcal{V}_s, \tilde{s}_s, d_s)$ is solved getting back $\tilde{c}^\star$
  - if $\tilde{c}^\star = \mathbf{0}$, or $\sum_{j=1}^{j=m} \tilde{x}_{j,1} c_j^\star \neq 0$ then no such solution exists, otherwise:
  - such a subset $\mathcal{L} \subset \mathcal{S}$ does exist, and is given by $\mathcal{L} = \{w_i \in \mathcal{S} \,:\, c_i^\star \geq 1\}$.
  - Note that it does not matter if $c_i^\star > 1$ as for such cases clipping it to multiplicity 1 is just as optimal. Which we will prove below.

Proof. d_s is a metric

The d_s is a special case of

$$d\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}\right) = \sum_{1 \leq j \leq n} \omega_i\, d'\,(x_j, y_j)$$

for d' a metric defined on scalars and $\forall i$ where $\omega_i, x_i, w_i \in \mathbb{R}$ and $\omega_i > 0$

Below it is shown that that this is always a metric, and thus d_s is a metric, by showing it meets the 3 requirements: of the coincidence axiom, being symmetric, and of the triangle inequality. The following properties hold for all $\tilde{a}, \tilde{b}, \tilde{c} \in \mathbb{R}^n$.

If d follows the coincidence axiom: $d(x, y) = 0 \iff x = y$. This is shown by:
if $\tilde{a} = \tilde{b}$ then $\forall j \in [1, n]$ $d'(a_j, b_j) = 0$ thus $d(\tilde{a}, \tilde{b}) = 0$.

and if $d(\tilde{a},\tilde{b})=0$ then as all $\forall j \in [1,n]\ w_j > 0$, therefore $d'(a_j,b_j)=0$.

If d is symmetric then $d(x,y)=d(y,x)$. Shown by:

$$d(\tilde{a},\tilde{b}) = \sum_{j=1}^{j=n} \omega_i\, d'\left(a_j,b_j\right) = \sum_{j=1}^{j=n} \omega_i\, d'\left(b_j,a_j\right) = d(\tilde{b},\tilde{a})$$

Thus d is symmetric

if d follows the triangle inequality then $d(x,z) \leq d(x,y)+d(y,z)$.
making use of $d'\left(a_j,b_j\right)+d'\left(b_j,c_j\right) \geq d'(a_j,c_j)$, It is shown:

$$\begin{aligned}
d(\tilde{a},\tilde{b})+d(\tilde{b},\tilde{c}) &= \sum_{1\leq j\leq n} \omega_i\, d'\left(a_j,b_j\right) + \sum_{1\leq j\leq n} \omega_i\, d'\left(b_j,c_j\right) \\
d(\tilde{a},\tilde{b})+d(\tilde{b},\tilde{c}) &= \sum_{1\leq j\leq n} \omega_i \left(d'\left(a_j,b_j\right)+ d'\left(b_j,c_j\right)\right) \\
d(\tilde{a},\tilde{b})+d(\tilde{b},\tilde{c}) &\leq \sum_{1\leq j\leq n} \omega_i \left(d'(a_j,c_j)\right) \\
d(\tilde{a},\tilde{b})+d(\tilde{b},\tilde{c}) &\leq d(\tilde{a},\tilde{b})
\end{aligned}$$

Thus d is a metric, and so d_s is a metric. □

Proof. Show for $c_j \notin \{0,1\}$ a better or at least equally good solution can be found for $c_j^{alt} \in \{0,1\}$

For a proof by contraction, we assume the existence of some optimal solution the Vector Selection Problem, c^* where for at least one index i,$c_i^* \geq 2$.

Consider also some alternative count vector (which by our assumption, can not be more optimal)

$$c' = c^* - \hat{e}_i$$

That is, c' is the same as c^*except that there is one less count for c_i^*

recalling $\tilde{s}_s = \begin{pmatrix} 0 \\ m \\ 0.5 \\ \vdots \\ 0.5 \end{pmatrix}$

we define the optimal sum of vectors by $\tilde{t}^*$

$$\tilde{t}^* = \sum_{j=1}^{j=V} \tilde{x}_j c_j^* = \begin{bmatrix} \left(\sum_{j=1}^{j=m} w_j c_j^*\right) \\ \left(\sum_{j=1}^{j=m} w_j c_j^*\right) \\ c_1^* \\ \vdots \\ c_n^* \end{bmatrix}$$

we define the alternative sum of vectors by $\tilde{t}'$

$$\tilde{t}' = \sum_{j=1}^{j=V} \tilde{x}_j c'_j = \left(\sum_{j=1}^{j=V} \tilde{x}_j c^*_j \right) - \tilde{x}_i = \begin{bmatrix} \left(\sum_{j=1}^{j=m} w_j c^*_j\right) - w_i \\ \left(\sum_{j=1}^{j=m} w_j c^*_j\right) - 1 \\ c^*_1 \\ \vdots \\ c^*_{i-1} \\ c^*_i - 1 \\ c^*_{i+1} \\ \vdots \\ c^*_n \end{bmatrix}$$

Note: we know that $c^*_i > 0.5$ as $c^*_i \geq 2$. Similarly we know $c'_i \geq 0.5$ for the as $c'_i = c^*_i - 1$

$$d_s(\tilde{s}_s, \tilde{t}^*) = \begin{array}{rc} & \left(\sum_{j=1}^{j=m} w_j c^*_j\right) \\ + & \omega \left| \left(\sum_{j=1}^{j=m} c^*_j\right) - m \right| \\ + & \Omega \left| c^*_1 - 0.5 \right| \\ & \vdots \\ + & \Omega \left| c^*_{i-1} - 0.5 \right| \\ + & \Omega (c^*_i - 0.5) \\ + & \Omega \left| c^*_{i+1} - 0.5 \right| \\ & \vdots \\ + & \Omega \left| c^*_n - 0.5 \right| \end{array}$$

and

$$d_s(\tilde{s}_s, \tilde{t}') = \begin{array}{rc} & \left(\sum_{j=1}^{j=m} w_j c^*_j\right) - w_i \\ + & \omega \left| \left(\sum_{j=1}^{j=m} c^*_j\right) - 1 - m \right| \\ + & \Omega \left| c^*_1 - 0.5 \right| \\ & \vdots \\ + & \Omega \left| c^*_{i-1} - 0.5 \right| \\ + & \Omega (c^*_i - 0.5 - 1) \\ + & \Omega \left| c^*_{i+1} - 0.5 \right| \\ & \vdots \\ + & \Omega \left| c^*_n - 0.5 \right| \end{array}$$

Since $\tilde{t}^*$ from the more optimal solution: $d_s(\tilde{s}_s, \tilde{t}') - d_s(\tilde{s}_s, \tilde{t}^*) \geq 0$

$$\begin{array}{rc} & \left(\sum_{j=1}^{j=m} w_j c^*_j\right) - w_i \\ + & \omega \left| \left(\sum_{j=1}^{j=m} c^*_j\right) - 1 - m \right| \\ + & \Omega \left| c^*_1 - 0.5 \right| \\ & \vdots \\ + & \Omega \left| c^*_{i-1} - 0.5 \right| \\ + & \Omega (c^*_{i+1} - 0.5 - 1) \\ + & \Omega \left| c^*_{i+1} - 0.5 \right| \\ & \vdots \\ + & \Omega \left| c^*_n - 0.5 \right| \end{array} - \begin{array}{rc} & \left(\sum_{j=1}^{j=m} w_j c^*_j\right) \\ + & \omega \left| \left(\sum_{j=1}^{j=m} c^*_j\right) - m \right| \\ + & \Omega \left| c^*_1 - 0.5 \right| \\ & \vdots \\ + & \Omega \left| c^*_{i-1} - 0.5 \right| \\ + & \Omega (c^*_{i+1} - 0.5) \\ + & \Omega \left| c^*_{i+1} - 0.5 \right| \\ & \vdots \\ + & \Omega \left| c^*_n - 0.5 \right| \end{array} \geq 0$$

And after canceling terms:

$$-w_i + \omega \left(\left| \left(\sum_{j=1}^{j=m} c^*_j \right) - 1 - m \right| - \left| \left(\sum_{j=1}^{j=m} c^*_j \right) - m \right| \right) - \Omega \geq 0$$

let $K = \left(\sum_{j=1}^{j=m} c_j^*\right) - m$

$$-w_i + \omega\left(|K-1| - |K|\right) - \Omega \geq 0$$

The largest value $|K-1| - |K|$ can take is 1. (The other cases are 0, and -1, both of which result in the contradiction of the sum of 2 and 3 negative values respectively being greater than or equal to zero)

$$-w_i + \omega - \Omega \geq -w_i + \omega\left(|K-1| - |K|\right) - \Omega \geq 0$$

$$w_i + \Omega \leq \omega$$

Substituting in the values from the definitions:
$\Omega = 2m\left(\max_{j\in[1,m]} |w_j| + 1\right)$ and $\omega = \frac{1}{2m}$

$$w_i + 2m\left(\max_{j\in[1,m]} |w_j|\right) + 2m \leq \frac{1}{2m}$$

$$2mw_i + 4m^2\left(\max_{j\in[1,m]} |w_j|\right) + 4m^2 \leq 1$$

Assume w_i takes the most negative value possible: $w_i = -\left(\max_{j\in[1,m]} |w_j|\right)$ giving:

$$\left(4m^2 - 2m\right)\left(\max_{j\in[1,m]} |w_j|\right) + 4m^2 \leq 2mw_i + 4m^2\left(\max_{j\in[1,m]} |w_j|\right) + 4m^2 \leq 1$$

As $m \geq 1$, consider it taking that the smallest value it can take (so $m = 1$)
$2\left(\max_{j\in[1,m]} |w_j|\right) + 4 \leq \left(4m^2 - 2m\right)\left(\max_{j\in[1,m]} |w_j|\right) + 4m^2 \leq 1$
requiring, $\max_{j\in[1,m]} |w_j| \leq -\frac{3}{2}$
Which is impossible as the absolute value of an integer is always non-negative.
Thus a contradiction.
Thus c' is at least as optimal as c^*.
We may apply this proof to all claimed optimal solutions with a $c_i > 1$ to show that an equally optimal (or more so), solution has that c_i at 1 lower.
Thus if some solution with any count $c_i > 1$ is found, it can be transformed into a solution that is equally (or more so) optimal, by clipping all counts c_i at one. □

A finer proof could be developed showing strict inequality and that c' yields a strictly better solution that c^*

Proof. Proof of Correctness
let $\tilde{c}'$ be the solution to the Vector Selection Problem on $(\mathcal{V}_s, \tilde{s}_s, d_s)$
As it was shown above that for any $c_i' > 1$ an equally optimal solution can be created by clipping $c_i'$to 1.
We will thus assume $c_i' \in \{0, 1\}$.
let $L' = \{w_i \in \mathcal{S} : c_i' = 1\}$

Case 1. Subset Sum Exists, but the Vector Selection Problem based method says it does not
We assume for a proof by contradiction that the the Vector Selection Problem based method states that no such subset sub exists,
however it is incorrect and such a subset does and is given by $L^* \subseteq \mathcal{S}$.

Then $\mathcal{L}^*$defines a indicator vector $c^* \in \{0,1\}^m$, given by $c_j^* = \begin{cases} 1 & w_j \in \mathcal{L}^* \\ 0 & w_j \notin \mathcal{L}^* \end{cases}$, where w_j is the jth element of $\mathcal{S}$.
so $\sum_{j=1}^{j=m} w_j c_j^* = 0$.
Note also as $\mathcal{L}^* \neq \emptyset$ (by definition of subset sum) $\exists i \in [1, m]$ such that $c_i^* = 1$.
we define $\tilde{t}^*$ to be the sum of the vectors which correspond to c_j^* by

$$\tilde{t}^* = \sum_{j=1}^{j=V} \tilde{x}_j c_j^* = \begin{bmatrix} \sum_{j=1}^{j=m} w_j c_j^* \\ \sum_{j=1}^{j=m} c_j^* \\ c_1^* \\ \vdots \\ c_m^* \end{bmatrix} = \begin{bmatrix} 0 \\ \sum_{j=1}^{j=m} c_j^* \\ c_1^* \\ \vdots \\ c_m^* \end{bmatrix}$$

and so

$$d_s(\tilde{s}, \tilde{t}^*) = d_s\left(\begin{bmatrix} 0 \\ m \\ 0.5 \\ \vdots \\ 0.5 \\ \vdots \\ 0.5 \end{bmatrix}, \begin{bmatrix} 0 \\ \sum_{j=1}^{j=m} c_j^* \\ c_1^* \\ \vdots \\ c_m^* \end{bmatrix}\right) = \begin{matrix} 0 \\ + \quad \omega \left| \left(\sum_{j=1}^{j=m} c_j^* \right) - m \right| \\ + \quad \Omega \left| c_1^* - 0.5 \right| \\ \vdots \\ + \quad \Omega \left| c_n^* - 0.5 \right| \end{matrix} = \omega \left| \left(\sum_{j=1}^{j=m} c_j^* \right) - m \right| + 0.5m\Omega$$

since $0 < \sum_{j=1}^{j=m} c_j^* \leq m$ as it is sum of m variables $0 \leq c_j^* \leq 1$ and not all $c_j^* = 0$, we can that to simplify to

$$d_s(\tilde{s}, \tilde{t}^*) = \omega \left(m - \sum_{j=1}^{j=m} c_j^* \right) + 0.5m\Omega$$

Now then consider the cases when the method (incorrectly) reports no such subset exists:

Case i. $\tilde{c}' = [0, ..., 0]$ (the zero vector)

We define the total sum of vectors given by $\tilde{t}'$

$$\tilde{t}' = \sum_{j=1}^{j=V} \tilde{x}_j c_j' = \begin{bmatrix} \sum_{j=1}^{j=m} w_j c_j' \\ \sum_{j=1}^{j=m} c_j' \\ c_1' \\ \vdots \\ c_m' \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

So find

$$d_s(\tilde{s}, \tilde{t}') = Multidimentionald_s\left(\begin{bmatrix} 0 \\ m \\ 0.5 \\ \vdots \\ 0.5 \\ \vdots \\ 0.5 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}\right) = \omega m + 0.5m\Omega$$

thus $d_s(\tilde{s}, \tilde{t}^*) < d_s(\tilde{s}, \tilde{t}')$ and so $c' = [0, ..., 0]$ could not have been the solution returned for solving the Vector Selection Problem as it is not the correct selection for the argmax. Thus a contradiction.

Case ii. $\tilde{c}' \neq \mathbf{0}$ thus $\sum_{j=1}^{j=m} \tilde{x}_{j,1} c_j' = k$ for $k \neq 0$

So thus the method reports that there is no nonempty subset which sums to zero; (the closest it can get is summing to k.

We redefine $\tilde{t}'$ for this case to be

$$\tilde{t}' = \sum_{j=1}^{j=V} \tilde{x}_j c_j' = \begin{bmatrix} \sum_{j=1}^{j=m} w_j c_j' \\ \sum_{j=1}^{j=m} c_j' \\ c_1' \\ \vdots \\ c_m' \end{bmatrix} = \begin{bmatrix} k \\ \sum_{j=1}^{j=m} c_j' \\ c_1' \\ \vdots \\ c_m' \end{bmatrix}$$

and so

$$d_s(\tilde{s},\tilde{t'}) = d_s(\begin{bmatrix} 0 \\ m \\ 0.5 \\ \vdots \\ 0.5 \\ \vdots \\ 0.5 \end{bmatrix}, \begin{bmatrix} k \\ \sum_{j=1}^{j=m} c'_j \\ c'_1 \\ \vdots \\ c'_m \end{bmatrix}) = \begin{matrix} \\ + \\ + \\ \\ + \end{matrix} \begin{matrix} |k| \\ \omega \left| \left(\sum_{j=1}^{j=m} c'_j \right) - m \right| \\ \Omega \left| c'_1 - 0.5 \right| \\ \vdots \\ \Omega \left| c'_n - 0.5 \right| \end{matrix} = |k| + \omega \left| \left(\sum_{j=1}^{j=m} c'_j \right) - m \right| + 0.5m\Omega$$

As $c'$was selected over c^* then
by our
recalling:

$$d_s(\tilde{s},\tilde{t}^*) = \omega \left| \left(\sum_{j=1}^{j=m} c^*_j \right) - m \right| + 0.5m\Omega$$

as $\tilde{t'}$ is the sum of vectors giving min value for the distance to the target point $\tilde{s_s}$ thus

$$d_s(\tilde{s_s},\tilde{t'}) \leq d_s(\tilde{s_s},\tilde{t}^*)$$

i.e.

$$|k| + \omega \left| \left(\sum_{j=1}^{j=m} c'_j \right) - m \right| + 0.5m\Omega \leq \omega \left| \left(\sum_{j=1}^{j=m} c^*_j \right) - m \right| + 0.5m\Omega$$

let $C' = \left(\sum_{j=1}^{j=m} c'_j \right)$ and $C^* = \left(\sum_{j=1}^{j=m} c^*_j \right)$
as $0 \leq C' \leq m$ and $0 \leq C^* \leq m$ as both are sums of indicator variables (0,1)
$\left| \left(\sum_{j=1}^{j=m} c^*_j \right) - m \right| = |C^* - m| = m - C^*$ and similarly for C'
substituting in:
$|k| + \omega (m - C') + 0.5m\Omega \leq \omega (m - C^*) + 0.5m\Omega$
i.e $|k| + \omega C' \leq \omega C^*$
i.e $|k| \leq \omega (C^* - C')$
thus $(C^* > C')$ it can not be equal as otherwise $k = 0$ which would be a contradiction.
let $(C^* - C') = C_d$, $C_d \in \mathbb{N}$
$0 < C_d$ as otherwise $|k| = 0$ (which would be the contradiction)
$C_d \leq m$ as the largest case is $C^* = m$ and $C' = 0$
Substitute

$$|k| \leq \omega C_d$$

substitute from the definition of $\omega = \frac{1}{2m}$

$$|k| \leq \frac{1}{2m} C_d$$

consider the largest value C_d can take: $C_d = m$

$$|k| \leq \frac{m}{2m}$$

$$|k| \leq \frac{1}{2}$$

As k is an integer this would mean $k = 0$
But this is a contradiction as $|k| > 0$.
Therefore it is not possible for the the Vector Selection Problem based method to say there is no solution if there is a solution.
i.e. if a solution exists, the the Vector Selection Problem based method will find it.

Case 2. Case: Subset sum does not exists, but the Vector Selection Problem based method says it does
Then:

$$\sum_{j=1}^{j=m} \tilde{x}_{j,1} c'_j = 0$$

.

We know that $\tilde{c}' \neq \mathbf{0}$ as other wise the the Vector Selection Problem based method would have said no solution exists.

Thus $\mathcal{L}' \neq \emptyset$

further we know by definition of $\tilde{x_j}$ that $\tilde{x}_{j,1} = w_j$ for $w_j \in \mathcal{S}$

thus we have $\sum_{j=1}^{j=m} w_j c'_j = 0$

thus in fact the sum of the elements of $\mathcal{L}'$ is zero.

And so a subset sum does exist.

This is a contradiction, thus the the Vector Selection Problem based method will never say there is a solution unless one exists.

Thus the method described in Claim 2 is a correct method to solve subset sum.

□

0.2.1. *Subset Sum Reduction Concluding note:* Thus it has been shown that if a general solution to the vector selection problem can be found a solution to subset sum could be found which would take at most a linear amount of additional time. Thus were a polynomial time solution for the Vector Selection Problem found, it would show that $P = NP$. However, the proof above is only for the general case, which is defined over $(\mathcal{V}, \tilde{s}, d)$ for finite subsets of $\mathbb{R}^n$, $\mathcal{V}$; and any $\tilde{s} \in \mathbb{R}^n$, using any metric d. Thus the hardness result is only for the general case. Like for many problems from the knapsack family, there certainly exists special cases for which faster solutions are possible. For example $\mathcal{V} \subset \mathbb{R}^1_+$, $\tilde{s} = [0]$ and $d = (x, y) \mapsto |x - y|$, a linear time solution exists, found by finding the index of the smallest member of $\mathcal{V}$. The general problem however is not expected to have an exact solution in polynomial time.

References

1. Richard M Karp, *Reducibility among combinatorial problems*, Springer, 1972.

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

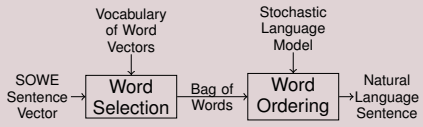


Figure 1. The Sel. BOW+Ord. process for the regenerating sentences from SOWE-type sentence vectors.

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.

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.

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.

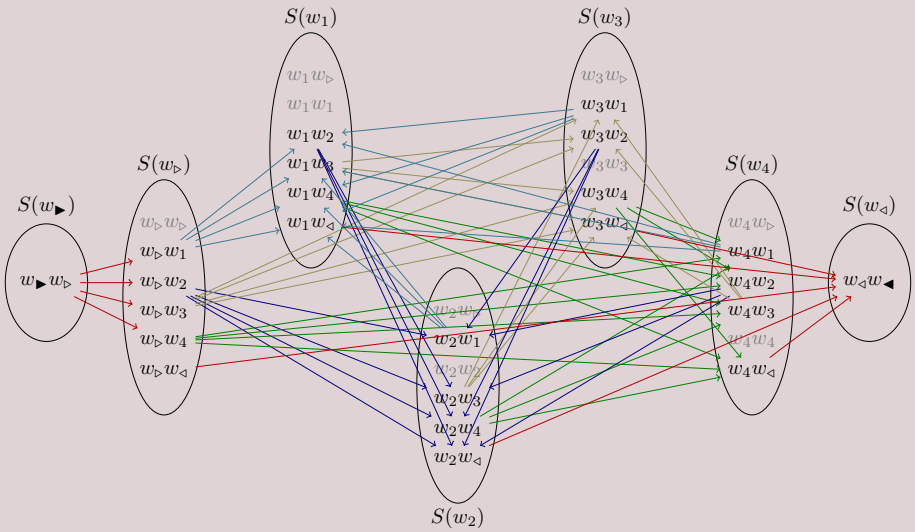


Figure 2. A graph showing the legal transitions between states, when the word-ordering problem is expressed similar to a GA-TSP. Each edge $\langle w_a, w_b\rangle \to \langle w_c, w_d\rangle$ has cost $-\log(P(w_c|w_a w_b)$. The nodes are grouped into districts (words). Nodes for invalid states are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

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

The table of transitions to be optimized is:

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

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.

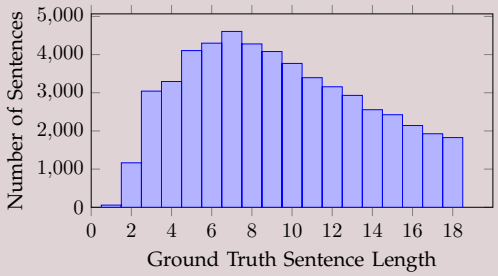


Figure 3. The distribution of the evaluation corpus after preprocessing.

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]

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

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

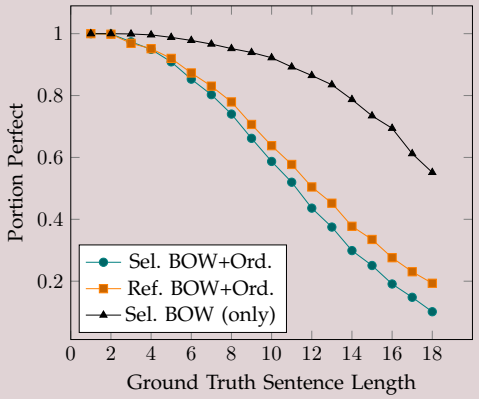


Figure 4. The portion of sentences reconstructed perfectly by the Sel. BOW+Ord. process. Shown also is the results on ordering only (Ref. BOW+Ord.), which orders the reference BOWS; and the portion of BOWs perfect from the word selection step only (Sel. BOW (only)) i.e. the input to the ordering step.

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

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", *International Conference on Learning Representations (ICLR) Workshop*, 2016.

[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.

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

Lyndon White, Roberto Togneri, Wei Liu and **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

These supplementary materials show additional examples of the performance of our method against the works of Iyyer, Boyd-Graber, and Daumé III [1] and Bowman, Vilnis, Vinyals, *et al.* [2], as of our well as on sentences with ambiguous order. Bare in mind, exact reproduction is not the goal of either prior work; nor truly is it a goal of out work. Our goal being the regeneration of sentences while preserving meaning – exact reproduction does of course meet that goal. The examples that follow should highlight the differences in the performance of the methods.

Tables 1 to 3 show quantitative examples; including comparison to the existing works. In these tables ✗ and ✓ are used to show correctness of the output in the selection (Sel.) and in the ordering (Ord.) steps.

The sentences shown in Table 1, are difficult. The table features long complex sentences containing many proper nouns. These examples are sourced from Iyyer, Boyd-Graber, and Daumé III [1]. The output from their DT-RAE method is also shown for contrast. Only 3C is completed perfectly by our method. Of the remainder the MIP word ordering problem has no solutions, except in 3D, where it is wrong, but does produce an ordered sentence. In the others the language model constraints does not return any feasible ($P(\tau) > 0$) ordering solutions. This failure may be attributed in a large part to the proper nouns. Proper nouns are very sparse in any training corpus for language modelling. The Kneser-Ney smoothed trigrams back-off only down to bigrams, so if the words of the bigrams from the training corpus never appear adjacently in the training corpus, ordering fails. This is largely the case for very rare words. The other significant factor is the sentence length.

The sentences in Table 2, are short and use common words – they are easy to resynthesis. These examples come from Bowman, Vilnis, Vinyals, *et al.* [2]. The output of their VAE based approach can be compared to that from our approach. Of the three there were two exact match's, and one failure.

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.

The examples shown in Table 3 highlight sentences where the order is ambiguous – where there are multiple reasonable solutions to the word ordering problem. In both cases the word selection performs perfectly, but the ordering is varied. In 5A, the Ref. BOW+Ord. sentence and the overall Sel. BOW+Ord. sentence in word order but not in word content. This is because under the trigram language model both sentences have exactly identical probabilities, so it comes to which solution is found first, which varies on the state of the MIP solver. In 5B the word order is switched – "from Paris to London" vs "to London from Paris", which has the same meaning. But, it could also have switched the place names. In cases like this where two orderings are reasonable, the ordering method is certain to fail consistently for one of the orderings. Though it is possible to output the second (and third etc.) most probable ordering, which does ameliorate the failure somewhat. This is the key limitation which prevents this method from direct practical applications.

		Sel.	Ord.
4A Reference	we looked out at the setting sun .		
Ref. BOW+Ord.	we looked out at the setting sun .	–	✓
Sel. BOW+Ord.	we looked out at the setting sun .	✓	✓
VAE Mean	they were laughing at the same time .		
VAE Sample1	ill see you in the early morning .		
VAE Sample2	i looked up at the blue sky .		
VAE Sample3	it was down on the dance floor .		
4B Reference	i went to the kitchen .	**Sel.**	**Ord.**
Ref. BOW+Ord.	i went to the kitchen .	–	✓
Sel. BOW+Ord.	i went to the kitchen .	✓	✓
VAE Mean	i went to the kitchen .		
VAE Sample1	i went to my apartment .		
VAE Sample2	i looked around the room .		
VAE Sample3	i turned back to the table .		
4C Reference	how are you doing ?	**Sel.**	**Ord.**
Ref. BOW+Ord.	how are you doing ?	–	✓
Sel. BOW+Ord.	how 're do well ?	✗	✗
VAE Mean	what are you doing ?		
VAE Sample1	are you sure ?		
VAE Sample2	what are you doing, ?		
VAE Sample3	what are you doing ?		

Table 2
A comparison of the output of the Two Step process proposed in this paper, to the example sentences generated by the VAE method of Bowman, Vilnis, Vinyals, *et al.* [2].

		Sel.	Ord.
5A Reference	it was the worst of times , it was the best of times .		
Ref. BOW+Ord.	it was the worst of times , it was the best of times .	–	✓
Sel. BOW+Ord.	it was the best of times , it was the worst of times .	✓	✗
5B Reference	please give me directions from Paris to London .	**Sel.**	**Ord.**
Ref. BOW+Ord.	please give me directions to London from Paris .	–	✗
Sel. BOW+Ord.	please give me directions to London from Paris .	✓	✗

Table 3
A pair of example sentences, where the correct order is particularly ambiguous.

References

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

[2] 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.

[3] 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.

		Sel.	Ord.
3A Reference	name this 1922 novel about leopold bloom written by james joyce .		
Ref. BOW+Ord.	written by name this . novel about 1922 bloom leopold james joyce	–	✖
Sel. BOW+Ord.	written novel by name james about leopold this bloom 1922 joyce .	✓	✖
DT-RAE Ref.	name this 1906 novel about gottlieb_fecknoe inspired by james_joyce		
DT-RAE Para.	what is this william golding novel by its written writer		
3B Reference	ralph waldo emerson dismissed this poet as the jingle man and james russell lowell called him three-fifths genius and two-fifths sheer fudge .	**Sel.**	**Ord.**
Ref. BOW+Ord.	sheer this as james two-fifths emerson fudge lowell poet genius waldo called russell the and ralph and him . dismissed jingle three-fifths man	–	✖
Sel. BOW+Ord.	him " james great as emerson genius ralph the lowell and sheer waldo three-fifths man fudge dismissed jingle russell two-fifths and gwalchmai 2009 vice-versa _____ prominent called 21.25 explained	✗	✖
DT-RAE Ref.	henry_david_thoreau rejected this author like the tsar boat and imbalance created known good writing and his own death		
DT-RAE Para.	henry_david_thoreau rejected him through their stories to go money well inspired stories to write as her writing		
3C Reference	this is the basis of a comedy of manners first performed in 1892 .	**Sel.**	**Ord.**
Ref. BOW+Ord.	this is the basis of a comedy of manners first performed in 1892 .	–	✓
Sel. BOW+Ord.	this is the basis of a comedy of manners first performed in 1892 .	✓	✓
DT-RAE Ref.	another is the subject of this trilogy of romance most performed in 1874		
DT-RAE Para.	subject of drama from him about romance		
3D Reference	in a third novel a sailor abandons the patna and meets marlow who in another novel meets kurtz in the congo .	**Sel.**	**Ord.**
Ref. BOW+Ord.	kurtz and another meets sailor meets the marlow who abandons a third novel in a novel in the congo in patna .	–	✗
Sel. BOW+Ord.	kurtz and another meets sailor meets the marlow who abandons a third novel in a novel in the congo in patna .	✓	✗
DT-RAE Ref.	during the short book the lady seduces the family and meets cousin he in a novel dies sister from the mr.		
DT-RAE Para.	during book of its author young lady seduces the family to marry old suicide while i marries himself in marriage		
3E Reference	thus she leaves her husband and child for aleksei vronsky but all ends sadly when she leaps in front of a train .	**Sel.**	**Ord.**
Ref. BOW+Ord.	train front of child vronsky but and for leaps thus sadly all her she she in when aleksei husband ends a . leaves	–	✖
Sel. BOW+Ord.	she her all when child for leaves front but and train ends husband aleksei leaps of vronsky in a sadly micro-history thus , she the	✗	✖
DT-RAE Ref.	however she leaves her sister and daughter from former fiancé and she ends unfortunately when narrator drives into life of a house		
DT-RAE Para.	leaves the sister of man in this novel		

Table 1

A comparison our method, to the example sentences generated by the DT-RAE method of Iyyer, Boyd-Graber, and Daumé III [1]. Ref. BOW+Ord. shows the word ordering step on the reference BOW. the Sel. and Ord. columns indicate if the output had the correct words selected, and ordered respectively. With ✓ indicating correct and ✗ indicating incorrect. ✖ indicates not only that ordering was not correct, but that the MIP problem had no feasible solutions at all. DT-RAE Ref. shows the result of the method of Iyyer, Boyd-Graber, and Daumé III [1], when the dependency tree of the output is provided to the generating process, whereas in DT-RAE Para. an arbitrary dependency tree is provided to the generating process. Note that the reference used as input to Sel. BOW+Ord. and Ref. BOW+Ord. sentence was varied slightly from that used in Iyyer, Boyd-Graber, and Daumé III [1] and White, Togneri, Liu, *et al.* [3], in that terminating punctuation was not removed, and nor were multiword entity references grouped into single tokens.