Sentence Ordering and Coherence Modeling using Recurrent Neural Networks

Lajanugen Logeswaran[1], **Honglak Lee**[1], **Dragomir Radev**[2]
[1]Department of Computer Science & Engineering, University of Michigan
[2]Department of Computer Science, Yale University
llajan@umich.edu, honglak@eecs.umich.edu, dragomir.radev@yale.edu

Abstract

Modeling the structure of coherent texts is a key NLP problem. The task of coherently organizing a given set of sentences has been commonly used to build and evaluate models that understand such structure. We propose an end-to-end unsupervised deep learning approach based on the set-to-sequence framework to address this problem. Our model strongly outperforms prior methods in the order discrimination task and a novel task of ordering abstracts from scientific articles. Furthermore, our work shows that useful text representations can be obtained by learning to order sentences. Visualizing the learned sentence representations shows that the model captures high-level logical structure in paragraphs. Our representations perform comparably to state-of-the-art pre-training methods on sentence similarity and paraphrase detection tasks.

1 Introduction

Modeling the structure of coherent texts is an important problem in NLP. A well-written text has a particular high-level logical and topical structure. The actual word and sentence choices and their transitions come together to convey the purpose of the text. Our primary goal is to build models that can learn such structure by arranging a given set of sentences to make coherent text.

Multi-document Summarization (MDS) and retrieval-based Question Answering (QA) involve extracting information from multiple documents and organizing it into a coherent summary. Since the relative ordering of sentences from different sources can be unclear, being able to automatically evaluate a particular order is essential. Barzilay and Elhadad (2002) discuss the importance of an ordering component in MDS and show that finding acceptable orderings can enhance user comprehension.

More importantly, by learning to order sentences we can model text coherence. It is difficult to explicitly characterize the properties of text that make it coherent. Ordering models attempt to understand these properties by learning high-level structure that causes sentences to appear in a specific order in human-authored texts. Automatic methods for evaluating human/machine generated text have great importance, with applications in essay scoring (Miltsakaki and Kukich 2004; Burstein, Tetreault, and Andreyev 2010) and text generation (Park and Kim 2015; Kiddon, Zettlemoyer, and Choi 2016). Coherence models aid the better design of these systems.

Exploiting unlabelled corpora to learn semantic representations of data has become an active area of investigation. Self-supervised learning is a typical approach that uses information naturally available as part of the data as supervisory signals (Wang and Gupta 2015; Doersch, Gupta, and Efros 2015). Noroozi and Favaro (2016) attempt to learn visual representations by solving image jigsaw puzzles. Sentence ordering can be considered as a jigsaw puzzle in the language domain and an interesting question is whether we can learn useful textual representations by performing this task.

Our approach to coherence modeling is driven by recent success in capturing semantics using distributed representations and modeling sequences using Recurrent Neural Nets (RNN). RNNs are now the dominant approach to sequence learning and mapping problems. The Sequence-to-sequence (Seq2seq) framework (Sutskever, Vinyals, and Le 2014) and its variants have fueled RNN based approaches to a range of problems such as language modeling, text generation, MT, QA, and many others.

In this work we propose an RNN-based approach to the sentence ordering problem which exploits the set-to-sequence framework of Vinyals, Bengio, and Kudlur (2015). A word-level RNN encoder produces sentence embeddings, and a sentence-level set encoder RNN iteratively attends to these embeddings and constructs a context representation. Initialized with this representation, a sentence-level pointer network selects the sentences sequentially.

The most widely studied task relevant to sentence ordering and coherence modeling is the order discrimination task. Given a document and a permuted version of it, the task involves identifying the more coherent ordering. Our proposed model achieves state-of-the-art performance on two benchmark datasets for this task, outperforming several classical approaches and recent data-driven approaches.

Addressing the more challenging task of ordering a given collection of sentences, we consider the novel and interesting task of ordering sentences from abstracts of scientific articles. Our model strongly outperforms previous work on this task. We visualize the learned sentence representations and show that our model captures high-level discourse structure. We provide visualizations that help understand what information in the sentences the model uses to identify the next sentence.

Finally, we demonstrate that our ordering model learns coherence properties and text representations that are useful for several downstream tasks including summarization, sentence similarity and paraphrase detection. In summary, our key contributions are as follows:

- We propose an end-to-end trainable model based on the set-to-sequence framework to address the problem of coherently ordering a collection of sentences.
- We consider the novel task of understanding structure in abstract paragraphs and demonstrate state-of-the-art results in order discrimination and sentence ordering tasks.
- We show that our model learns sentence representations that perform comparably to recent unsupervised pre-training methods on downstream tasks.

2 Related Work

Coherence modeling & sentence ordering. Coherence modeling and sentence ordering have been approached by closely related techniques. Many approaches propose a measure of coherence and formulate the ordering problem as finding an order with maximal coherence. Recurring themes from prior work include linguistic features, centering theory, local and global coherence.

Local coherence has been modeled by considering properties of local windows of sentences such as sentence similarity and transition structure. Lapata (2003) represent sentences by vectors of linguistic features and learn the transition probabilities between features of adjacent sentences. The Entity-Grid model (Barzilay and Lapata 2008) captures local coherence by modeling patterns of entity distributions. Sentences are represented by the syntactic roles of entities appearing in the document. Features extracted from the entity grid are used to train a ranking SVM. These two methods are motivated from centering theory (Grosz, Weinstein, and Joshi 1995), which states that nouns and entities in coherent discourses exhibit certain patterns.

Global models of coherence typically use HMMs to model document structure. The content model (Barzilay and Lee 2004) represents topics in a particular domain as states in an HMM. State transitions capture possible presentation orderings within the domain. Words of a sentence are modeled using a topic-specific language model. The content model inspired several subsequent work to combine the strengths of local and global models. Elsner, Austerweil, and Charniak (2007) combine the entity grid and the content model using a non-parametric HMM. Soricut and Marcu (2006) use several models as feature functions and define a log-linear model to assign probability to a text. Louis and Nenkova (2012) model the intentional structure in documents using syntax features.

Unlike previous approaches, we do not use any handcrafted features and adopt an embedding-based approach. Local coherence is taken into account by a next-sentence prediction component in our model, and global dependencies are naturally captured by an RNN. We demonstrate that our model can capture both logical and topical structure by several evaluation benchmarks.

Data-driven approaches. Neural approaches have gained attention recently. Li and Hovy (2014) model sentences as embeddings derived from recurrent neural nets and train a feed-forward neural network that takes an input window of sentence embeddings and outputs a probability which represents the coherence of the sentence window. Coherence evaluation is performed by sliding the window over the text and aggregating the score. Li and Jurafsky (2016) study the same model in a larger scale task and also use a sequence-to-sequence approach in which the model is trained to generate the next sentence given the current sentence and vice versa. Nguyen and Joty (2017) learn to model coherence using a convolutional network that operates on the Entity-Grid representation of an input document. These models are limited by their local nature; our experiments show that considering larger contexts is beneficial.

Hierarchical RNNs for document modeling. Word-level and sentence-level RNNs have been used in a hierarchical fashion for modeling documents in prior work. Li, Luong, and Jurafsky (2015) proposed a hierarchical autoencoder for generation and summarization applications. More relevant to our work is a similar model considered by Lin et al. (2015). A sentence-level RNN predicts the bag of words in the next sentence given the previous sentences and a word-level RNN predicts the word sequence conditioned on the sentence RNN hidden state. Our model has a hierarchical structure similar to these models, but takes a discriminative approach.

Combinatorial optimization with RNNs. Vinyals, Bengio, and Kudlur (2015) equip sequence-to-sequence models with the ability to handle input and output sets, and discuss experiments on sorting, language modeling and parsing. This is called the *read*, *process* and *write* (or set-to-sequence) model. The *read* block maps input tokens to a fixed length vector representation. The *process* block is an RNN encoder which, at each time-step, attends to the input token embeddings and computes an attention readout, appending it to the current hidden state. The *write* block is an RNN which produces the target sequence conditioned on the representation produced by the process block. Their goal is to show that input and output orderings can matter in these tasks, which is demonstrated using small scale experiments. Our work exploits this framework to address the challenging problem of modeling logical and hierarchical structure in text. Vinyals, Fortunato, and Jaitly (2015) proposed pointer-networks for combinatorial optimization problems where the output dictionary size depends on the number of input elements. We use a pointer-network as the decoder to sequentially pick the next sentence.

3 Approach

Our proposed model is inspired by the way a human would solve this task. First, the model reads the sentences to capture their meaning and the general context of the paragraph. Given this knowledge, the model tries to pick the sentences one by one sequentially till exhaustion.

Our model is based on the *read*, *process* and *write* framework of Vinyals, Bengio, and Kudlur (2015) briefly discussed in the previous section. We use the encoder-decoder terminology that is more common in the following discussion.

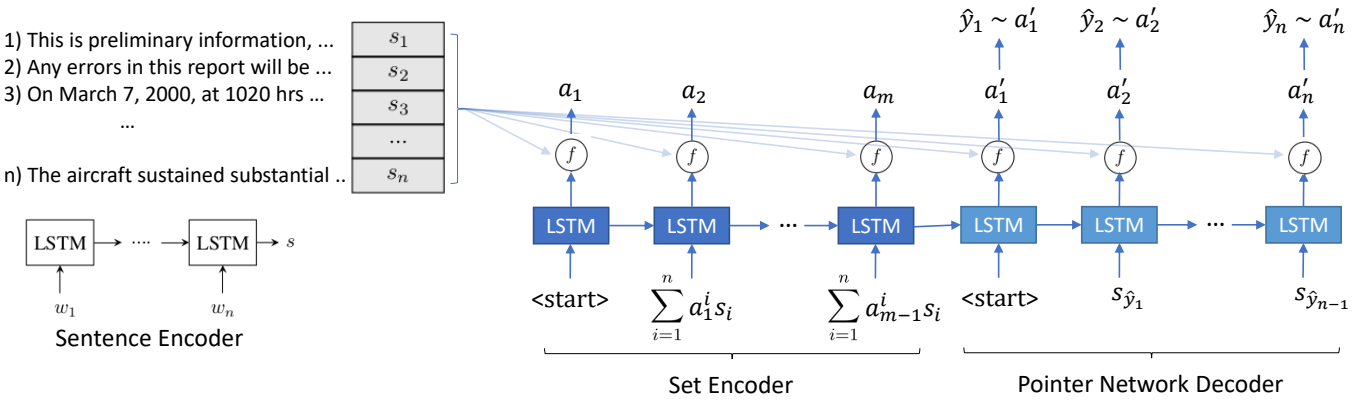


Figure 1: **Model Overview**: The input set of sentences are represented as vectors using a sentence encoder. At each time step of the model, attention weights are computed for the sentence embeddings based on the current hidden state. The encoder uses the attention probabilities to compute the input for the next time-step and the decoder uses them for prediction.

The model is comprised of a sentence encoder RNN, an encoder RNN and a decoder RNN (Fig. 1). The sentence encoder takes as input the words of a sentence s sequentially and computes an embedding representation of the sentence. Henceforth, we use s to refer to a sentence or its embedding interchangeably. The embeddings $\{s_1, s_2, ..., s_n\}$ of a given set of n sentences constitute the sentence memory, available to be accessed by subsequent components.

The encoder LSTM is identical to the originally proposed process block, defined by Eqs 1-4. At each time step the input to the LSTM is computed by taking a weighted sum over the memory elements, the weights being attention probabilities obtained using the previous hidden state as query (Eqs. 1, 2). This is iterated for a fixed number of times called the read cycles. Intuitively, the model identifies a soft input order to read the sentences. As described in Vinyals, Bengio, and Kudlur (2015) the encoder has the desirable property of being invariant to the order in which the sentence embeddings reside in the memory.

$$e^{t,i}_{\text{enc}} = f(s_i, h^t_{\text{enc}}); i \in \{1, ..., n\} \quad (1)$$

$$a^t_{\text{enc}} = \text{Softmax}(e^t_{\text{enc}}) \quad (2)$$

$$s^t_{\text{att}} = \sum_{i=1}^{n} a^{t,i}_{\text{enc}} s_i \quad (3)$$

$$h^{t+1}_{\text{enc}}, c^{t+1}_{\text{enc}} = \text{LSTM}(h^t_{\text{enc}}, c^t_{\text{enc}}, s^t_{\text{att}}) \quad (4)$$

The decoder is a pointer network that takes a similar form with a few differences (Eqs. 5-7). The LSTM takes the embedding of the previous sentence as input instead of the attention readout. At training time the correct order of sentences $(s_{o_1}, s_{o_2}, ..., s_{o_n}) = (x^1, x^2, ..., x^n)$ is known (o represents the correct order) and x^{t-1} is used as the input. At test time the predicted assignment $\hat{x}^{t-1}$ is used instead. The attention computation is identical to that of the encoder, but now $a^{t,i}_{\text{dec}}$ is interpreted as the probability for s_i being the correct sentence choice at position t, conditioned on the previous sentence assignments $p(S_t = s_i | S_1, ..., S_{t-1})$. The initial state of the decoder LSTM is initialized with the final hidden state of the encoder. x^0 is a vector of zeros.

$$h^t_{\text{dec}}, c^t_{\text{dec}} = \text{LSTM}(h^{t-1}_{\text{dec}}, c^{t-1}_{\text{dec}}, x^{t-1}) \quad (5)$$

$$e^{t,i}_{\text{dec}} = f(s_i, h^t_{\text{dec}}); i \in \{1, ..., n\} \quad (6)$$

$$a^t_{\text{dec}} = \text{Softmax}(e^t_{\text{dec}}) \quad (7)$$

Scoring Function. We consider two choices for the scoring function f in Eqs. 1, 6. The first (Eq. 8) is a single hidden layer feed-forward net that takes s, h as inputs (W, b, W', b' are learnable parameters). The structure of f is similar to the window network of Li and Hovy (2014). While they used a local window of sentences to capture context, this scoring function exploits the entire history of sentences encoded in the RNN hidden state to score candidates for the next sentence.

$$f(s, h) = W' \tanh(W[s\,;\,h] + b) + b' \quad (8)$$

We also consider a bilinear scoring function (Eq. 9). Compared to the previous scoring function, this takes a generative approach to regress the next sentence given the current hidden state $(Wh + b)$, enforcing that it be most similar to the correct next sentence. We observed that this scoring function led to better sentence representations (Sec. 4.4).

$$f(s, h) = s^T(Wh + b) \quad (9)$$

Contrastive Sentences. In its vanilla form, we found that the set-to-sequence model tends to rely on certain word clues to perform the ordering task. To encourage holistic sentence understanding, we add a random set of sentences to the sentence memory when the decoder makes classification decisions. This makes the problem more challenging for the decoder since now it has to distinguish between sentences that are relevant and irrelevant to the current context in identifying the correct sentence.

Coherence modeling. We define the coherence score of an arbitrary partial/complete assignment $(s_{p_1}, ..., s_{p_k})$ to the first k sentence positions as

$$\sum_{i=1}^{k} \log p(S_i = s_{p_i} | S_{1,...,i-1} = s_{p_1,...,p_{i-1}}) \quad (10)$$

where $S_1, .., S_k$ are random variables representing the sentence assignment to positions 1 through k. The conditional

Table 1: Mean Accuracy comparison on the Accidents and Earthquakes data for the order discrimination task. The reference models are Entity-Grid (Barzilay and Lapata 2008), HMM (Louis and Nenkova 2012), Graph (Guinaudeau and Strube 2013), Window network (Li and Hovy 2014) and sequence-to-sequence (Li and Jurafsky 2016), respectively.

	Entity-Grid	HMM	Graph	Window (Recurrent)	Window (Recursive)	Seq2seq	Ours
Accidents	0.904	0.842	0.846	0.840	0.864	0.930	**0.944**
Earthquakes	0.872	0.957	0.635	0.951	0.976	0.992	**0.997**

probabilities are derived from the network. This is our measure of comparing the coherence of different renderings of a document. It is also used as a heuristic during decoding.

Training Objective. The model is trained using the maximum likelihood objective

$$\max \sum_{x \in D} \sum_{t=1}^{|x|} \log p(x^t | x^1, ..., x^{t-1}) \qquad (11)$$

where D denotes the training set and each training instance is given by an ordered document of sentences $x = (x^1, ..., x^{|x|})$.

4 Experimental Results

We first consider the order discrmination task that has been widely used in the literature for evaluating coherence models. We then consider the more challenging ordering problem where a coherent order of a given collection of sentences needs to be determined. We then demonstrate that our ordering model learns coherence properties useful for summarization. Finally, we show that our model learns sentence representations that are useful for downstream applications.

For all tasks discussed in this section we train the model with the maximum likelihood objective on the training data relevant to the task. We used the single hidden layer MLP scoring function for the order discrimination and sentence ordering tasks. Models are trained end-to-end. We use pretrained 300 dimensional GloVe word embeddings (Pennington, Socher, and Manning 2014) to initialize word vectors. All LSTMs use a hidden layer size of 1000 and the MLP in Eq. 8 has a hidden layer size of 500. The number of read cycles in the encoder is set to 10. The same model architecture is used across all experiments. We used the Adam optimizer (Kingma and Ba 2014) with batch size 10 and learning rate 5e-4 for learning. The model is regularized using early stopping. Hyperparameters were chosen using the validation set.

4.1 Order Discrimination

The ordering problem is traditionally formulated as a binary classification task: Given a reference paragraph and its permuted version, identify the more coherent one (Barzilay and Lapata 2008).

The datasets widely used for this task in previous work are the Accidents and Earthquakes news reports. In each of these datasets the training and test sets include 100 articles and approximately 20 permutations of each article.

In Table 1 we compare our results with traditional approaches and recent data-driven approaches. The entity grid model provides a strong baseline on the Accidents dataset, only outperformed by our model and Li and Jurafsky (2016). On the Earthquakes data the window approach of Li and Jurafsky (2016) performs strongly. Our approach outperforms prior models on both datasets, achieving near perfect performance on the Earthquakes dataset.

While these datasets have been widely used, they are quite formulaic in nature and are no longer challenging. We hence turn to the more challenging task of ordering a given collection of sentences to make a coherent document.

4.2 Sentence Ordering

In this task we directly address the ordering problem. We do not assume the availability of a set of candidate orderings to choose from and instead find a good ordering from all possible permutations of the sentences.

The difficulty of the ordering problem depends on the nature of the text, as well as the length of paragraphs considered. Evaluation on text from arbitrary text sources makes it difficult to interpret the results, since it may not be clear whether to attribute the observed performance to a deficient model or ambiguity in next sentence choices due to many plausible orderings.

Text summaries are a suitable source of data for this task. They often exhibit a clear flow of ideas and have little redundancy. We specifically look at abstracts of conference papers and research proposals. This data has several favorable properties. Abstracts usually have a particular high-level format - They begin with a brief introduction, a description of the problem and proposed approach and conclude with performance remarks. This would allow us to identify if the model can capture high-level logical structure. Second, abstracts have an average length of about 10, making the ordering task more accessible. This also gives us a significant amount of data to train and test our models.

We use the following sources of abstracts for this task.

- *NIPS Abstracts*. We consider abstracts from NIPS papers in the past 10 years. We parsed 3280 abstracts from paper pdfs and obtained 3259 abstracts after omitting erroneous extracts. The dataset was split into years 2005-2013 for training and 2014, 2015 respectively for validation, testing.
- *ACL Abstracts*. A second source of abstracts are papers from the ACL Anthology Network (AAN) corpus (Radev et al. 2009). We extracted 12,157 abstracts from the text parses using simple keyword matching for the strings 'Abstract' and 'Introduction'. We use all extracts of papers published up to year 2010 for training, year 2011 for validation and years 2012-2013 for testing.

Table 2: Comparison against prior methods on the abstracts data.

	NIPS Abstracts		AAN Abstracts		NSF Abstracts	
	Accuracy	τ	Accuracy	τ	Accuracy	τ
Random	15.59	0	19.36	0	9.46	0
Entity Grid (Barzilay and Lapata 2008)	20.10	0.09	21.82	0.10	-	-
Seq2seq (Uni) (Li and Jurafsky 2016)	27.18	0.27	36.62	0.40	13.68	0.10
Window network (Li and Hovy 2014)	41.76	0.59	50.87	0.65	18.67	0.28
RNN Decoder	48.22	0.67	52.06	0.66	25.79	0.48
Proposed model	**51.55**	**0.72**	**58.06**	**0.73**	**28.33**	**0.51**

- *NSF Abstracts*. We also used the NSF Research Award Abstracts dataset (Lichman 2013). It comprises abstracts from a diverse set of scientific areas in contrast to the previous two sources of data and the abstracts are also lengthier, making this dataset more challenging. Years 1990-1999 were used for training, 2000 for validation and 2001-2003 for testing. We capped the parses of the abstracts to a maximum length of 40 sentences. Unsuccessful parses and parses of excessive length were discarded.

Further details about the data are provided in the supplement.

The following metrics are used to evaluate performance. *Accuracy* measures how often the absolute position of a sentence was correctly predicted. *Kendall's tau (τ)* is computed as $1 - 2 \cdot N/\binom{n}{2}$, where N is the number of pairs in the predicted sequence with incorrect relative order and n is the sequence length. Lapata (2006) discusses that this metric reliably correlates with human judgements.

The following baselines are used for comparison:

- **Entity Grid**. Our first baseline is the Entity Grid model of Barzilay and Lapata (2008). We use the Stanford parser (Klein and Manning 2003) and Brown Coherence Toolkit[1] to derive Entity grid representations. A ranking SVM is trained to score correct orderings higher than incorrect orderings as in the original work. We used 20 permutations per document as training data. Since the entity grid only provides a means of feature extraction we evaluate the model in the ordering setting as follows. We choose 1000 random permutations for each document, one of them being the correct order, and pick the order with maximum coherence. We experimented with transitions of length at most 3 in the entity-grid.
- **Seq2seq**. The second baseline we consider is a sequence-to-sequence model which is trained to predict the next sentence given the current sentence. Li and Jurafsky (2016) consider similar methods and our model is the same as their uni-directional model. These methods were shown to yield sentence embeddings that have competitive performance in several semantic tasks in Kiros et al. (2015).
- **Window Network**. We consider the window approach of Li and Hovy (2014) and Li and Jurafsky (2016) which demonstrated strong performance in the order discrimination task as our third baseline. We adopt the same coherence score interpretation considered by the authors. In both the above models we consider a special embedding vector which is padded at the beginning of a paragraph and learned during training. This vector allows us to identify the initial few sentences during greedy decoding.
- **RNN Decoder**. Another baseline is our proposed model without the encoder. The decoder hidden state is initialized with zeros. We observed that using a special start symbol as for the other baselines helped obtain better performance with this model. However, a start symbol did not help when the model is equipped with an encoder as the hidden state initialization alone was good enough.

We do not place emphasis on the particular search algorithm in this work and thus use beam search with the coherence score heuristic for all models. A beam size of 100 was used. During decoding, sentence candidates that have been already chosen are pruned from the beam. All RNNs use a hidden layer size of 1000. For the window network we used a window size of 3 and a hidden layer size of 2000. We initialize all models with pre-trained GloVe word embeddings.

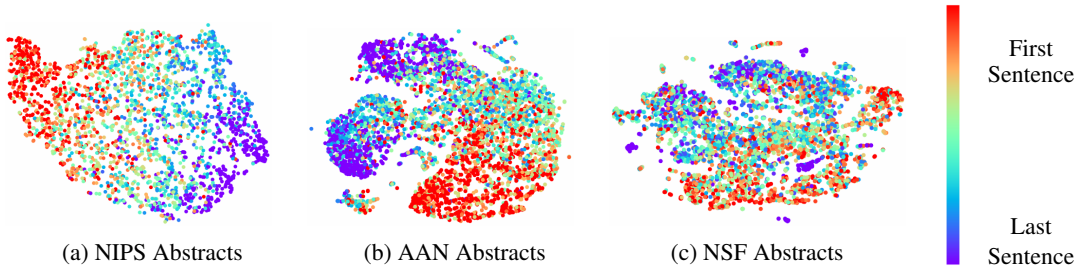
We assess the performance of our model against baseline methods in Table 2. The window network performs strongly compared to the other baselines. Our model does better by a significant margin by exploiting global context, demonstrating that global context is important in this task.

While the Entity-Grid model has been fairly successful for the order discrimination task in the past we observe that it fails to discriminate between a large number of candidates. One reason could be that the feature representation is less sensitive to local changes in sentence order (such as swapping adjacent sentences). The computational expense of obtaining parse trees and constructing grids on a large amount of data prohibited experimenting with this model on the NSF abstracts data.

The Seq2seq model performs worse than the window network. Interestingly, Li and Jurafsky (2016) observe that the Seq2seq model outperforms the window network in an order discrimination task on Wikipedia data. However, the Wikipedia data considered in their work is an order of magnitude larger than the datasets considered here, and that could have potentially helped the generative model. These models are also expensive during inference since they involve computing and sampling from word distributions.

Fig. 2 shows t-SNE embeddings of sentence representations learned by our sentence encoder. These are sentences from test sets, color coded by their positions in the source

[1] bitbucket.org/melsner/browncoherence



(a) NIPS Abstracts (b) AAN Abstracts (c) NSF Abstracts

Figure 2: t-SNE embeddings of representations learned by the model for sentences from the test set. Embeddings are color coded by the position of the sentence in the document it appears.

Table 3: Comparison on extractive summarization between models trained from scratch and models pre-trained with the ordering task.

Model	ROUGE-1	ROUGE-2	ROUGE-L
Summary length = 75b			
From scratch	18.29	47.56	12.79
Pre-train ord.	18.77	50.32	13.25
Summary length = 275b			
From scratch	35.82	10.67	33.69
Pre-train ord.	36.47	10.99	34.27

Table 4: Performance comparison for semantic similarity and paraphrase detection. The first row shows the best performing purely supervised methods. The last section shows our models.

Model	SICK			MSRP	
	r	ρ	MSE	(Acc)	(F1)
Supervised	0.868	0.808	0.253	80.4	86.0
Uni-ST (Kiros et al. 2015)	0.848	0.778	0.287	73.0	81.9
Ordering model	0.807	0.742	0.356	72.3	81.1
+ BoW	0.842	0.775	0.299	74.0	81.9
+ Uni-ST	0.860	0.795	0.270	74.9	82.5

abstract. This shows that our model learns high-level structure in the documents, generalizing well to unseen text. The structure is less apparent in the NSF dataset due to its data diversity and longer documents. While approaches based on the Barzilay and Lee (2004) model explicitly capture topics by discovering clusters in sentences, our neural approach implicitly discovers such structure.

4.3 Sentence Ordering and Summarization

In this section we show that sentence ordering models learn coherence properties useful for summarization. We consider a variation of our model where the model takes a set of sentences from several documents as input and sequentially picks summary sentences until it predicts a special 'stop' symbol. A key distinction between this model and recent work (Cheng and Lapata 2016; Nallapati, Zhou, and Ma 2016) is that the input order of sentences is assumed to be unknown, making it applicable to multi-document summarization.

We train a model from scratch to perform extractive summarization in the above fashion. We then consider a model that is pre-trained on the ordering task and is fine-tuned on the above task. The DailyMail and CNN datasets (Cheng and Lapata 2016) were used for experimentation. We use DailyMail for pre-training purposes and CNN for fine-tuning and evaluation. The labels in DailyMail are not used. We compare ROUGE scores of the two models in Table 3 under standard evaluation settings.

We observe that the model pre-trained with the ordering task scores consistently better than the model trained from scratch. The results can be improved further by using larger news corpora. This shows that sentence ordering is an attractive unsupervised objective for exploiting large unlabelled corpora to improve summarization systems. It further shows that the coherence scores obtained from the ordering model correlates well with summary quality.

4.4 Learned Sentence Representations

One of the original motivations for this work is the question of whether we can learn high-quality sentence representations by learning to model text coherence. To address this question we trained our model on a large number of paragraphs using the BookCorpus dataset (Kiros et al. 2015).

To evaluate the quality of sentence embeddings derived from the model, we use the evaluation pipeline of Kiros et al. (2015) for tasks that involve understanding sentence semantics. These evaluations are performed by training a classifier on top of the embeddings derived from the model (holding the embeddings fixed) so that the performance is indicative of the quality of sentence representations. We present a comparison for the semantic relatedness and paraphrase detection tasks in Table 4. Results for only uni-directional versions of models are discussed here for a fair comparison. Similar to the skip-thought (ST) paper, we train two models - one predicting the correct order in the forward direction and another in the backward direction. The numbers shown for the ordering model were obtained by concatenating the representations from the two models.

Concatenating the above representation with the bag-of-words representation (using the fine-tuned word embeddings) of the sentence further improves performance. This is because the ordering model can choose to pay less attention to specific lexical information and focus on high-level document structure. Hence, the two representations capture complementary semantics. Adding ST features improves performance

Table 5: Visualizing salient words (Abstracts are from the AAN corpus).

In this paper , we propose a new method for semantic class induction . First , we introduce a generative model of sentences , based on dependency trees and which takes into account homonymy . Our model can thus be seen as a generalization of Brown clustering . Second , we describe an efficient algorithm to perform inference and learning in this model . Third , we apply our proposed method on two large datasets (108 tokens , 105 words types) , and demonstrate that classes induced by our algorithm improve performance over Brown clustering on the task of semisupervised supersense tagging and named entity recognition .
Representation learning is a promising technique for discovering features that allow supervised classifiers to generalize from a source domain dataset to arbitrary new domains . We present a novel , formal statement of the representation learning task . We argue that because the task is computationally intractable in general , it is important for a representation learner to be able to incorporate expert knowledge during its search for helpful features . Leveraging the Posterior Regularization framework , we develop an architecture for incorporating biases into representation learning . We investigate three types of biases , and experiments on two domain adaptation tasks show that our biased learners identify significantly better sets of features than unbiased learners , resulting in a relative reduction in error of more than 16% for both tasks , with respect to state-of-the-art representation learning techniques.

further. We observed that the bilinear scoring function and introducing contrastive sentences to the decoder improved the quality of learned representations significantly.

Our model has several key advantages over ST. ST has a word-level reconstruction objective and is trained with large softmax output layers. This limits the vocabulary size and slows down training (they use a vocabulary size of 20k and report two weeks of training). Our model achieves comparable performance and does not have such a word reconstruction component. We train with a vocabulary of 400k words; the above results are based on a training time of two days on a GTX Titan X GPU.

4.5 Word Influence

We attempt to understand what text-level clues the model uses to perform the ordering task. Inspired by Li et al. (2015), we use gradients of prediction decisions with respect to words of the correct sentence as a proxy for the salience of each word. We feed sentences to the decoder in the correct order and at each time step compute the derivative of the score e (Eq. 6) of the correct next sentence $s = (w_1, .., w_n)$ with respect to its word embeddings. The importance of word w_i in correctly predicting s as the next sentence is defined as $\|\frac{\partial e}{\partial w_i}\|$. We assume the hidden states of the decoder to be fixed and only back-propagate gradients through the sentence encoder.

Table 5 shows visualizations of two abstracts. Darker shades correspond to higher gradient norms. In the first example the model appears to be using the word clues 'first', 'second' and 'third'. A similar observation was made by Chen, Qiu, and Huang (2016). In the second example we observe that the model pays attention to phrases such as 'We present', 'We argue', which are typical of abstract texts. It also focuses on the word 'representation' appearing in the first two sentences. These observations link to centering theory which states that entity distributions in coherent discourses exhibit certain patterns. The model implicitly learns these patterns with no syntax annotations or handcrafted features.

5 Conclusion

This work investigated the challenging problem of coherently organizing a set of sentences. Our RNN-based model performs strongly compared to baselines and prior work on sentence ordering and order discrimination tasks. We further demonstrated that it captures high-level document structure and learns useful sentence representations when trained on large amounts of data. Our approach to the ordering problem deviates from most prior work that use handcrafted features. However, exploiting linguistic features for next sentence classification can potentially further improve performance. Entity distribution patterns can provide useful features about named entities that are treated as out-of-vocabulary words. The ordering problem can be further studied at higher-level discourse units such as paragraphs, sections and chapters.

6 Acknowledgments

This material is based in part upon work supported by IBM under contract 4915012629. Any opinions, findings, conclusions or recommendations expressed above are those of the authors and do not necessarily reflect the views of IBM. We thank the UMich/IBM Sapphire team and Junhyuk Oh, Ruben Villegas, Xinchen Yan, Rui Zhang, Kibok Lee and Yuting Zhang for helpful comments and discussions.

References

Barzilay, R., and Elhadad, N. 2002. Inferring strategies for sentence ordering in multidocument news summarization. *Journal of Artificial Intelligence Research* 35–55.

Barzilay, R., and Lapata, M. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics* 34(1):1–34.

Barzilay, R., and Lee, L. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. *arXiv preprint cs/0405039*.

Burstein, J.; Tetreault, J.; and Andreyev, S. 2010. Using entity-based features to model coherence in student essays. In *Human language technologies: The 2010 annual conference of the North American chapter of the Association for Computational Linguistics*, 681–684. Association for Computational Linguistics.

Chen, X.; Qiu, X.; and Huang, X. 2016. Neural sentence ordering. *arXiv preprint arXiv:1607.06952*.

Cheng, J., and Lapata, M. 2016. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*.

Doersch, C.; Gupta, A.; and Efros, A. A. 2015. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, 1422–1430.

Elsner, M.; Austerweil, J. L.; and Charniak, E. 2007. A unified local and global model for discourse coherence. In *HLT-NAACL*, 436–443.

Grosz, B. J.; Weinstein, S.; and Joshi, A. K. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational linguistics* 21(2):203–225.

Guinaudeau, C., and Strube, M. 2013. Graph-based local coherence modeling. In *ACL (1)*, 93–103.

Kiddon, C.; Zettlemoyer, L.; and Choi, Y. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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

Kiros, R.; Zhu, Y.; Salakhutdinov, R. R.; Zemel, R.; Urtasun, R.; Torralba, A.; and Fidler, S. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, 3276–3284.

Klein, D., and Manning, C. D. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, 423–430. Association for Computational Linguistics.

Lapata, M. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, 545–552. Association for Computational Linguistics.

Lapata, M. 2006. Automatic evaluation of information ordering: Kendall's tau. *Computational Linguistics* 32(4):471–484.

Li, J., and Hovy, E. H. 2014. A model of coherence based on distributed sentence representation. In *EMNLP*, 2039–2048.

Li, J., and Jurafsky, D. 2016. Neural net models for open-domain discourse coherence. *arXiv preprint arXiv:1606.01545*.

Li, J.; Chen, X.; Hovy, E.; and Jurafsky, D. 2015. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*.

Li, J.; Luong, M.-T.; and Jurafsky, D. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

Lichman, M. 2013. UCI machine learning repository.

Lin, R.; Liu, S.; Yang, M.; Li, M.; Zhou, M.; and Li, S. 2015. Hierarchical recurrent neural network for document modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 899–907.

Louis, A., and Nenkova, A. 2012. A coherence model based on syntactic patterns. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 1157–1168. Association for Computational Linguistics.

Miltsakaki, E., and Kukich, K. 2004. Evaluation of text coherence for electronic essay scoring systems. *Natural Language Engineering* 10(01):25–55.

Nallapati, R.; Zhou, B.; and Ma, M. 2016. Classify or select: Neural architectures for extractive document summarization. *arXiv preprint arXiv:1611.04244*.

Nguyen, D. T., and Joty, S. 2017. A neural local coherence model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 1320–1330.

Noroozi, M., and Favaro, P. 2016. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, 69–84. Springer.

Park, C. C., and Kim, G. 2015. Expressing an image stream with a sequence of natural sentences. In *Advances in Neural Information Processing Systems*, 73–81.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, 1532–43.

Radev, D. R.; Joseph, M. T.; Gibson, B.; and Muthukrishnan, P. 2009. A Bibliometric and Network Analysis of the field of Computational Linguistics. *Journal of the American Society for Information Science and Technology*.

Soricut, R., and Marcu, D. 2006. Discourse generation using utility-trained coherence models. In *Proceedings of the COLING/ACL on Main conference poster sessions*, 803–810. Association for Computational Linguistics.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

Vinyals, O.; Bengio, S.; and Kudlur, M. 2015. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, 2674–2682.

Wang, X., and Gupta, A. 2015. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, 2794–2802.