SENTENCE ORDERING USING RECURRENT NEURAL NETWORKS

Lajanugen Logeswaran, Honglak Lee & Dragomir Radev
Department of EECS
University of Michigan
Ann Arbor, MI 48109, USA
{llajan,honglak,radev}@umich.edu

ABSTRACT

Modeling the structure of coherent texts is a task of great importance in NLP. The task of organizing a given set of sentences into a coherent order has been commonly used to build and evaluate models that understand such structure. In this work we propose an end-to-end neural approach based on the recently proposed set to sequence mapping framework to address the sentence ordering problem. Our model achieves state-of-the-art performance in the order discrimination task on two datasets widely used in the literature. We also consider a new interesting task of ordering abstracts from conference papers and research proposals and demonstrate strong performance against recent methods. Visualizing the sentence representations learned by the model shows that the model has captured high level logical structure in these paragraphs. The model also learns rich semantic sentence representations by learning to order texts, performing comparably to recent unsupervised representation learning methods in the sentence similarity and paraphrase detection tasks.

1 INTRODUCTION

Modeling the structure of coherent texts is one of the central problems in NLP. A well written piece of text has a particular high level logical and topical structure to it. The actual word and sentence choices as well as their transitions come together to convey the purpose of the text. Our overarching goal is to build models that can learn such structure by learning to arrange a given set of sentences to make coherent text.

The sentence ordering task finds several applications. Multi-document Summarization (MDS) and retrieval based question answering involve extracting information from multiple source documents and organizing the content into a coherent summary. Since the relative ordering about sentences that come from different sources can be unclear, being able to automatically evaluate a particular order and/or finding the optimal order is essential. Barzilay and Elhadad (2002) discuss the importance of an explicit ordering component in MDS systems. Their experiments show that finding an acceptable ordering can enhance user comprehension.

Models that learn to order text fragments can also be used as models of coherence. Automated essay scoring (Miltsakaki and Kukich, 2004; Burstein et al., 2010) is an application that can benefit from such a coherence model. Coherence is one of the key elements on which student essays are evaluated in standardized writing tests such as GRE (ETS). Apart from its importance and applications, our motivation to address this problem also stems from its stimulating nature. It can be considered as a jigsaw puzzle of sorts in the language domain.

Our approach to the problem of modeling coherence is driven by recent successes in 1) capturing semantics using distributed representations and 2) using RNNs for sequence modeling tasks.

Success in unsupervised approaches for learning embeddings for textual entities from large text corpora altered the way NLP problems are studied today. These embeddings have been shown to capture syntactic and semantic information as well as higher level analogical structure. These methods have been adopted to learn vector representations of sentences, paragraphs and entire

documents. Embedding based approaches allow models to be trained end-to-end from scratch with no handcrafting.

Recurrent Neural Networks (RNNs) have become the de facto approach to sequence learning and mapping problems in recent times. The Sequence to sequence mapping framework (Sutskever et al., 2014), as well as several of its variants have fuelled RNN based approaches to a wide variety of problems including language modeling, language generation, machine translation, question answering and many others.

Vinyals et al. (2015a) recently showed that the order in which tokens of the input sequence are fed to seq2seq models has a significant impact on the performance of the model. In particular, for problems such as sorting which involve a source set (as opposed to a sequence), the optimal order to feed the tokens is not clear. They introduce an attention mechanism over the input tokens which allows the model to learn a soft input order. This is called the *read*, *process* and *write* (or set to sequence) framework. The *read* block maps the 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.

In this work we propose an RNN based approach to the sentence ordering problem which exploits the set to sequence framework. A word level RNN encoder produces sentence embeddings. A sentence level set encoder RNN iteratively attends to these embeddings (*process* block above) and constructs a representation of the context. Initialized with this representation, a sentence level pointer network RNN points to the next sentence candidates.

The most widely studied task relevant to sentence ordering and coherence modeling in the literature is the order discrimination task. Given a document and a permuted version of it, the task involves identifying the more coherent ordering of the two. Our proposed model achieves state of the art performance on two benchmark datasets for this task, outperforming several classical approaches and more 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 conference papers and research grants. 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 aid understanding what information in the sentences the model uses to identify the next sentence. We also study the quality of the sentence representations learned by the model by training the model on a large text corpus and show that these embeddings are comparable to recent unsupervised methods in capturing semantics.

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 challenging problem of organizing a given collection of sentences in a coherent order.
- 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 demonstrate that the proposed model is capable of learning semantic representations of sentences that are comparable to recently proposed methods for learning such representations.

2 Related Work

Coherence modeling and sentence ordering The coherence modeling and sentence ordering tasks have been approached by closely related techniques. Most 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 a local window of sentences such as sentence similarity and sentence transition structure. Foltz et al. (1998) represent words using vectors of co-occurent counts and sentences as a mean of these word vectors. Sentence similarity is defined as the cosine distance between sentence vectors and text coherence is modeled as a normalized sum of similarity scores of adjacent sentences. Lapata (2003) represents sentences by vectors of

linguistic features and learn the transition probabilities from one set of features to another in adjacent sentences. A popular model of coherence is the Entity-Grid model Barzilay and Lapata (2008) which captures local coherence by modeling patterns of entity distributions in the discourse. Sentences are represented by the syntactic roles of entities appearing in the document and entity transition frequencies in successive sentences are treated as features that are are used to train a ranking SVM. These two approaches find motivation from ideas in centering theory (Grosz et al., 1995) which state that nouns and entities in coherent discourses exhibit certain patterns.

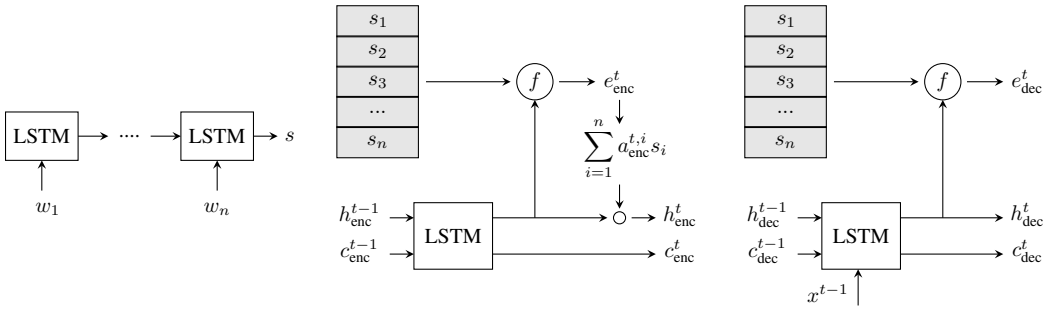
Global models of coherence typically use an HMM to model document structure. The content model proposed by Barzilay and Lee (2004) represents topics in a particular domain as states in an HMM. State transitions capture possible presentation orderings within the domain. Words of a sentence are modeled using a topic-specific language model. The content model has inspired several subsequent work to combine the strengths of local and global models. Elsner et al. (2007) combine the entity model 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 given text. Louis and Nenkova (2012) attempt to capture the intentional structure in documents using syntax as a proxy for the communicative goal of a sentence. Syntax features such as parse tree production rules and constituency tags at a particular tree depth were used.

Unlike previous approaches, we do not employ any handcrafted features and adopt an embedding based approach. Local coherence is taken into account by having a next sentence prediction component in the model and global dependencies are naturally captured by an RNN. We demonstrate that our model is able to capture both logical and topical structure by evaluating its performance on different types of data.

Data-driven approaches Neural approaches have gained attention more recently. Li and Hovy (2014) model sentences as embeddings derived from recurrent/recursive neural nets and train a feedforward 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 consider a sequence to sequence approach where the model is trained to generate the next sentence given the current sentence and vice versa. Chen et al. (2016) also propose a sentence embedding based approach where they model the probability that one sentence should come before another and define coherence based on the likelihood of the relative order of every pair of sentences. We believe these models are limited by the fact that they are local in nature and our experiments show that exploiting larger contexts can be very 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 et al. (2015b) proposed a hierarchical document autoencoder which has potential to be used in generation and summarization applications. More relevant to our work is a similar model (but without an encoder) 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 level RNN hidden state. The model has a structure similar to the content model of Barzilay and Lee (2004) with RNNs playing the roles of the HMM and the bigram language model. Our model has a hierarchical nature in that a sentence level RNN operates over words of a sentence and a document level RNN operates over sentence embeddings.

Combinatorial optimization with RNNs Vinyals et al. (2015a) equip sequence to sequence models with the capability to handle input and output sets, and discuss experiments on sorting, language modeling and parsing. Their goal is to show that input and output orderings can matter in these tasks, which is demonstrated using several small scale experiments. Our work exploits this framework to address the challenging problem of modeling logical and hierarchical structure in text. Vinyals et al. (2015b) proposed pointer-networks, aimed at combinatorial optimization problems where the output dictionary size depends on the number of input elements. We use a pointer-network that points to each of the next sentence candidates as the decoder.



(a) Sentence Encoder (b) Encoder (c) Decoder

Figure 1: **Model Overview**: Illustration of the sentence encoder and single time-step computations in encoder and decoder. s_i's represent sentence embeddings derived from the sentence encoder. Attention weights are computed for the sentences based on their embeddings and the current hidden state. In the encoder an attention readout is concatenated with the LSTM output to form the next hidden state. The decoder uses the attention weights for prediction.

3 APPROACH

Our proposed model is inspired by the way a human would solve this task. First, the model attempts to read the sentences to capture the semantics of the sentences as well as the general context of the paragraph. Given this knowledge, the model attempts to pick the sentences one by one sequentially till exhaustion.

Our model is based on the read, process and write framework proposed by Vinyals et al. (2015a) briefly discussed in section 1. We use the encoder-decoder terminology that is more common in the literature in the following discussion.

The model is comprised of a sentence encoder RNN, an encoder RNN and a decoder RNN (figure 1). An RNN sentence encoder takes as input the words of a sentence s sequentially and computes an embedding representation of the sentence (Figure 1a). Henceforth, we shall 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 is identical to the originally proposed process block and is defined by equations 1-5 (See Figure 1b). Following the regular LSTM hidden state $(h^{t-1}_{\text{enc}}, c^{t-1}_{\text{enc}})$ update, the hidden state is concatenated with an attention readout vector s^t_{att}, and this concatenated vector is treated as the hidden state for the next time step (Equation 5). Attention probabilities are computed by composing the hidden state with embeddings of the candidate sentences through a scoring function f and taking the softmax (Equations 2, 3). This process is iterated for a number of times, called the number of read cycles. As described in Vinyals et al. (2015a) the encoder has the desirable property of being invariant to the order in which the sentence embeddings reside in the memory. The LSTM used here does not take any inputs (input is clamped to zero).

$$\widetilde{h}^t_{\text{enc}}, c^t_{\text{enc}} = \text{LSTM}(h^{t-1}_{\text{enc}}, c^{t-1}_{\text{enc}}) \tag{1}$$

$$e^{t,i}_{\text{enc}} = f(s_i, \widetilde{h}^t_{\text{enc}}); i \in \{1, ..., n\} \tag{2}$$

$$a^t_{\text{enc}} = \text{Softmax}(e^t_{\text{enc}}) \tag{3}$$

$$s^t_{\text{att}} = \sum_{i=1}^{n} a^{t,i}_{\text{enc}} s_i \tag{4}$$

$$h^t_{\text{enc}} = [\widetilde{h}^t_{\text{enc}} \;\; s^t_{\text{att}}] \tag{5}$$

The decoder is a pointer network that takes a similar form with a few differences (equations 6-8, Figure 1c). The LSTM takes the embedding of the previous sentence also as input: 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. This makes concatenating the attention readout to the hidden state somewhat redundant (verified empirically), and hence it is omitted. The attention computation is identical to that of the encoder.

The initial state of the decoder LSTM is initialized with the final hidden state of the encoder as in sequence to sequence models.[1] x^0 is a vector of zeros. Figure 1 illustrates the single time-step computation in the encoder and decoder.

$$h^t_{\text{dec}}, c^t_{\text{dec}} = \text{LSTM}(h^{t-1}_{\text{dec}}, c^{t-1}_{\text{dec}}, x^{t-1}) \tag{6}$$

$$e^{t,i}_{\text{dec}} = f(s_i, h^t_{\text{dec}}); i \in \{1, ..., n\} \tag{7}$$

$$a^t_{\text{dec}} = \text{Softmax}(e^t_{\text{dec}}) \tag{8}$$

The attention probability $a^{t,i}_{\text{dec}}$ is interpreted as the probability for s_i being the correct sentence choice at position t, conditioned on the previous sentence assignments $p(S_t = s_i | S_1, ..., S_{t-1})$.

3.1 Scoring Function

We consider two choices for the scoring functions f in our experiments. The first one is a single hidden layer feed-forward net that takes s, h as inputs and outputs a score $f(s, h) = W' tanh(W[s \;\; h] + b) + b'$ where W, b, W', b' are learnable parameters. This scoring function takes a discriminative approach in classifying the next sentence. Note that the structure of this scoring function is similar to the window network in Li and Hovy (2014). While they used a local window of sentences to capture context, this scoring function exploits the RNN hidden state to score sentence candidates.

We also consider a bilinear scoring function $f(s, h) = s^T(Wh + b)$. Compared to the previous scoring function, this takes a generative approach of trying to regress the next sentence given the current hidden state $(Wh + b)$ and enforcing that it be most similar to the correct next sentence. We observed that this scoring function led to learning better sentence representations (section 4.4).

3.2 Training Objective

The model is trained with the maximum likelihood objective:

$$\max \sum_{x \in D} \sum_{t=1}^{|x|} \log p(x^t | x^1, ..., x^{t-1}) \tag{9}$$

where D denotes the training set and each training instance is given by an ordered document of sentences $x = (x^1, ..., x^{|x|})$. We also considered an alternative structured margin loss which imposes less penalty for assigning high scores to sentence candidates that are close to the correct sentence in the source document instead of uniformly penalizing all incorrect sentence candidates. However, the softmax output with cross entropy loss consistently performed better.

3.3 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

$$\sum_{i=1}^{k} \log p(S_i = s_{p_i} | S_1 = s_{p_1}, ..., S_{i-1} = s_{p_{i-1}}) \tag{10}$$

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

4 Experimental Results

4.1 Model Training

For both the tasks discussed in the next sections we train the model with the same objective (equation 9) on the training data relevant to the task. Models are trained end-to-end.

Model parameters. We use pre-trained 300 dimensional GloVe word embeddings (Pennington et al., 2014). All LSTMs use a hidden layer size of 1000 and the MLP in section 3.1 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.

[1] A subtle difference is that the final hidden state of the encoder h^N_{enc} has more dimensions than h^0_{dec} and only the first part of the vector is copied (The attention readout is ignored for this time step).

Table 1: Statistics of data used in our experiments. For the first two datasets, the test set size* is the number of permutation pairs used for order discrimination experiments.

Dataset	Length Statistics				Data Split			Types
	Min	Mode	Mean	Max	Train	Val	Test	
Accidents	6	11	11.6	19	100	-	1986*	5,140
Earthquakes	3	7	10.4	31	100	-	1956*	3,775
NIPS abstracts	2	7	6	15	2448	409	402	18,696
AAN abstracts	1	4	5	20	8569	962	2626	40,288
NSF abstracts	2	7	8.9	40	96070	10185	21580	373,909

Preprocessing. The nltk sentence tokenizer was used for word tokenization. The GloVe vocabulary was used as the reference vocabulary. Any word not in the vocabulary is checked for a case insensitive match. If a token is hyphenated, we check if the constituent words are in the vocabulary. In the AAN abstracts data (section 4.3.1), some words tend to have a hyphen in the middle because of word hyphenation across lines in the original document. Hence we also check if stripping hyphens produces a vocabulary word. If all checks fail, and a token appears in the training set above a certain frequency, it is added to the vocabulary.
Learning . We used a batch size of 10 and the Adam optimizer (Kingma and Ba, 2014) with a base learning rate of 5e-4 for all experiments. Early stopping is used for regularization.

4.2 Order Discrimination

Finding the optimal ordering is a difficult problem when a large number of sentences are required to be rearranged or when there is inherent ambiguity in the ordering of the sentences. For this reason, the ordering problem is commonly formulated as the following binary classification task. Given a reference paragraph and a permuted version of it, the more coherently organized one needs to be identified (Barzilay and Lapata, 2008).

4.2.1 Data

We consider data from two different domains that have been widely used for this task in previous work since Barzilay and Lee (2004); Barzilay and Lapata (2008). The ACCIDENTS data (aka AIRPLANE data) is a set of aviation accident reports from the National Transportation Safety Board's database. The EARTHQUAKES data comprises newspaper articles from the North American News Text Corpus. In each of the above datasets the training and test sets include 100 articles as well as approximately 20 permutations of each article. Further statistics about the data are shown in table 1.

4.2.2 Results

Table 2 compares the performance of our model against prior approaches. We compare results against traditional approaches in the literature as well as some recent data-driven approaches (See section 2 for more details). The entity grid model provides a strong baseline on the ACCIDENTS dataset, only outperformed by our model and Li and Jurafsky (2016) . On the EARTHQUAKE data the window approach of Li and Hovy (2014) and Li and Jurafsky (2016) perform strongly. Our approach outperforms prior models on both datasets, achieving near perfect performance on the Earthquakes dataset.

While these datasets have been widely used in the literature, 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.

4.3 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 attempt to find a good ordering from all possible permutations of the sentences.

Table 2: Mean Accuracy comparison on the Accidents and Earthquakes data for the order discrimination task. Reference results obtained from the respective publications.

Methods	ACCIDENTS	EARTHQUAKES
Barzilay and Lapata (2008)	0.904	0.872
Louis and Nenkova (2012)	0.842	0.957
Guinaudeau and Strube (2013)	0.846	0.635
Li and Hovy (2014) - Recurrent	0.840	0.951
Li and Hovy (2014) - Recursive	0.864	0.976
Li and Jurafsky (2016)	0.930	0.992
Ours	**0.944**	**0.997**

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 minimal redundancy. We specifically look at abstracts of conference papers and NSF research proposals. This data has several favorable properties. Abstracts usually have a particular high level format - They start out with a brief introduction, a description of the problem addressed and proposed approach and conclude with performance remarks. This would allow us to identify if the model is capable of capturing high level logical structure. Second, abstracts have an average length of about 10, making the ordering task more accessible. Furthermore, this also gives us a significant amount of data to train and test our models.

4.3.1 DATA

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 years 2014, 2015 respectively for validation and testing[2].

ACL Abstracts. A second source of abstracts we consider are papers from the ACL Anthology Network (AAN) corpus (Radev et al., 2009) of ACL papers. At the time of retrieval, the corpus had publications up to year 2013. We extracted abstracts from the text parses using simple keyword matching for the strings 'Abstract' and 'Introduction'. Our extraction is successful for 12,157 articles. Most of the failures occur for older papers due to improper formatting and OCR issues. We use all extracts of papers published up to year 2010 for training, year 2011 for validation and years 2012-2013 for testing. We additionally merge words hyphenated at the edges of paragraph boundaries.

NSF Abstracts. We also evaluate our model on the NSF Research Award Abstracts dataset (Lichman, 2013). This dataset 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 datasets are provided in table 1.

4.3.2 METRICS

We use the following metrics to evaluate performance on this task. **Accuracy** measures how often the absolute position of a sentence was correctly predicted. Being a too stringent measure, it penalizes correctly predicted subsequences that are shifted. Another metric widely used in the literature is **Kendall's tau** (τ), computed as $1 - 2 \times (\text{number of inversions}) / \binom{n}{2}$, where the number of inversions is the number of pairs in the predicted sequence with incorrect relative order and n is the length of the sequence. Lapata (2006) discusses that this metric reliably correlates with human judgements.

[2]Experimentation with a random split yielded similar performance. We adopt this split so that future work can easily perform comparisons with our results.

Table 3: 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	37.51	0.50	48.40	0.60	18.97	0.29
Proposed model	**51.55**	**0.72**	**58.06**	**0.73**	**28.33**	**0.51**

4.3.3 BASELINES

Entity Grid. Our first baseline is the Entity Grid model of Barzilay and Lapata (2008). We use the Stanford parser Klein and Manning (2003) to get constituency trees for all sentences in our datasets. We derive entity grid representations for the parsed sentences using the Brown Coherence Toolkit.[3] 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 representation 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.

Sequence to sequence. 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 same as the uni-directional model in their work. 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 the above work. 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 we consider 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.

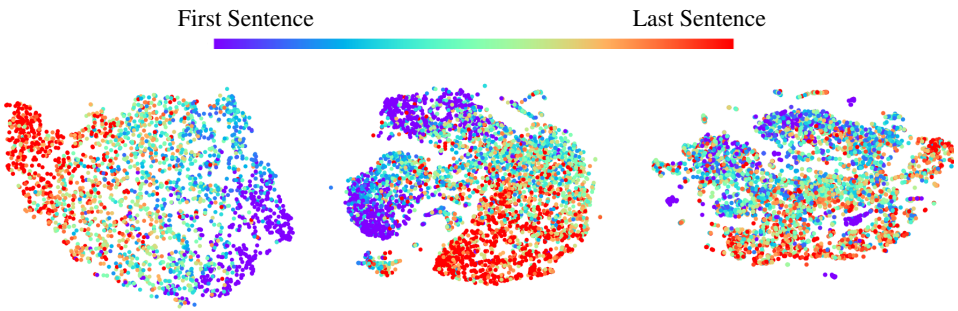
We do not place emphasis on the particular search algorithm in this work and thus use beam search using the coherence score heuristic for all models. A beam size of 100 was used. During decoding, sentence candidates that have been already chosen are pruned from the beam. All RNNs use a hidden layer size 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.

4.3.4 RESULTS

We assess the performance of our model against baseline methods in table 3. The window network performs strongly compared to the other baselines, outperforming the decoder only version of our model in the NIPS and ACL abstracts data. These results show that local context alone conveys significant information about the next sentence. Our model does better by a significant margin by exploiting global context, which shows how global context is important to be successful 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 fairly less sensitive to local changes in sentence order (such as

[3] https://bitbucket.org/melsner/browncoherence/overview



(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. The embeddings are color coded by the position of the sentence in the document it appears.

swapping adjacent sentences). We did not use coreference resolution for computing the entity-grids due to the computational overhead. This could potentially improve results by a few percentage points. The computational expense of obtaining parse trees and constructing grids on a large amount of data prohibited us from experimenting with this model on the NSF abstracts data.

The sequence to sequence model falls short of the window network in performance. 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 has an order of magnitude more data that 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.

We observe that our model performs poorly when not equipped with the encoder. However, when the sentence encoder is initialized with a good set of parameters (such as pre-trained skip-thought parameters (Kiros et al., 2015)) this compensates for the lack of an encoder and leads to better performance. This shows that the encoder helps learn good representations and contributes significantly to performance. Another observation we made was that a randomly initialized sentence encoder with smaller hidden layer size performed better than one initialized with pre-trained skip-thought parameters in all the experiments.

In Figure 3 we attempt to visualize the sentence representations learned by the sentence encoder in our model. The figure shows 2-dimensional t-SNE embeddings of test set sentences from each of the datasets color coded by their positions in the source abstract. This shows that the model learns high-level structure in the documents, generalizing well to unseen documents. The structure is less apparent in the NSF data which we presume is because of the data diversity and longer documents. While approaches based on the content model of Barzilay and Lee (2004) attempt to explicitly capture topics by discovering clusters in sentences, we observe that the neural approach implicitly discovers such structure.

4.4 Learned Sentence Representations

One of the original motivations for this work is the question 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 dataset of paragraphs. We chose the BookCorpus dataset (Kiros et al., 2015) for this purpose. We trained the model with two key changes from the models trained on the abstracts data - 1) In addition to the sentences in the paragraph being considered, we added more contrastive sentences from other paragraphs as well. 2) We use the bilinear scoring function. These techniques helped obtain better representations when training on large amounts of data.

To evaluate the quality of the 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 so that the performance is indicative of the quality of sentence representations. We consider the semantic relatedness and paraphrase detection tasks. Our results are presented in tables 4a, 4b. Results for only uni-directional versions of different models are discussed here for a reasonable comparison.

Table 4: Performance comparison for the semantic similarity (SICK dataset) and paraphrase detection (MSR paraphrase corpus) tasks. In each table the first section shows some best performing supervised methods in the literature. The second section shows models relevant to the skip-thought model. The third section shows our models.

(a) Sentence similarity

Method	r	ρ	MSE
Purely supervised methods			
DT-RNN (Tai et al., 2015)	0.792	0.732	0.382
LSTM (Tai et al., 2015)	0.853	0.791	0.283
DT-LSTM (Tai et al., 2015)	0.868	0.808	0.253
Classifier trained on sentence embeddings			
skip-bow (Kiros et al., 2015)	0.782	0.724	0.398
uni-skip (Kiros et al., 2015)	0.848	0.778	0.287
Ordering model	0.807	0.742	0.356
+ BoW	0.842	0.775	0.299
+ uni-skip	0.860	0.795	0.270

(b) Paraphrase detection

Method	Acc	F1
Purely supervised methods		
Socher et al. (2011)	76.8	83.6
Madnani et al. (2012)	77.4	84.1
Ji and Eisenstein (2013)	80.4	86.0
Classifier trained on sentence embeddings		
uni-skip-bow (Kiros et al., 2015)	67.8	80.3
uni-skip (Kiros et al., 2015)	73.0	81.9
Ordering model	72.3	81.0
+ BoW	74.0	81.9
+ uni-skip	74.9	82.5

Skip-thought vectors are learned by predicting both the previous and next sentences given the current sentence. Following suit, we train two models - one predicting the correct order in the forward direction and another in the backward direction. Note that the sentence level RNN is still uni-directional in both cases. The numbers shown for the ordering model were obtained by concatenating the representations obtained 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[4]. We believe the reason to be that the ordering model can choose to pay less attention to specific lexical information and instead focus on the high level document structure. Hence the two representations can be seen as capturing complementary semantics. Adding the skip-thought embedding features as well improves performance further.

Our model has several key advantages over the skip-thought model. The skip-thought model has a word-level reconstruction objective and requires training with large softmax output layers. This limits the size of the vocabulary and makes training very time consuming (they use a vocabulary size of 20k and report 2 weeks of training). Our model achieves comparable performance and does not have such a word reconstruction component. We are able to train with a large vocabulary of 400k words and the above results were obtained with a training time of 2 days.

A conceptual issue surrounding word-level reconstruction is that it forces the model to predict both the meaning and syntax of the target sentence. This makes learning difficult since there are numerous ways of expressing the same idea in syntax. In our model we instead let the model discover features from a sentence which are both predictive (of the next sentence) and predictable (from the previous sentences) and interpret these set of features as a meaning representation. We believe this is an important distinction and hope to study these models further in the context of learning syntax independent semantic representations of sentences.

5 Conclusion

In this work we considered the challenging problem of coherently organizing a given set of sentences. Our RNN based model performs strongly compared to baseline methods as well as prior work on sentence ordering and order discrimination tasks. We further demonstrated that the model 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 on the task. 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.

[4]We used the same hyperparameters that were used for the abstracts data to train our model. The skip-bow and uni-skip embeddings have dimensionality 640, 2400 respectively. Representations from the ordering model have dimensionality 2000, and adding BoW features gives 2600 dimensional embeddings.

REFERENCES

Automated scoring of writing quality. URL https://www.ets.org/research/topics/as_nlp/writing_quality. Accessed: 2016-11-1.

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

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

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

J. Burstein, J. Tetreault, and S. Andreyev. 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*, pages 681–684. Association for Computational Linguistics, 2010.

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

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

P. W. Foltz, W. Kintsch, and T. K. Landauer. The measurement of textual coherence with latent semantic analysis. *Discourse processes*, 25(2-3):285–307, 1998.

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

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

Y. Ji and J. Eisenstein. Discriminative improvements to distributional sentence similarity. In *EMNLP*, pages 891–896, 2013.

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

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

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

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

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

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

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

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

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

M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

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

A. Louis and A. Nenkova. 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*, pages 1157–1168. Association for Computational Linguistics, 2012.

N. Madnani, J. Tetreault, and M. Chodorow. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 182–190. Association for Computational Linguistics, 2012.

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

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

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

R. Socher, E. H. Huang, J. Pennin, C. D. Manning, and A. Y. Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809, 2011.

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

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

K. S. Tai, R. Socher, and C. D. Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

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

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

Table 5: Visualizing salient words.

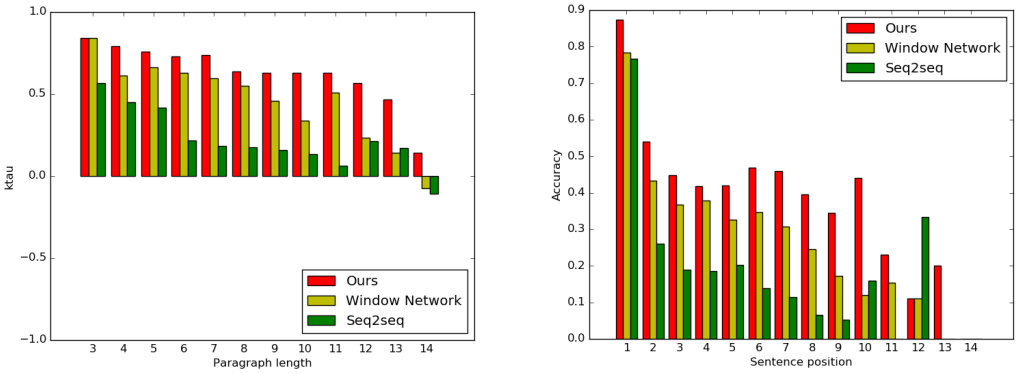
In this paper , we propose a new method for semantic class induction . First , we introduce a generative model of sentences , based on dependency trees and which takes into account homonymy . Our model can thus be seen as a generalization of Brown clustering . Second , we describe an efficient algorithm 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 existing state-of-the-art representation learning techniques.
We present an approach for detecting salient (important) dates in texts in order to automatically build event timelines from a search query (e.g . the name of an event or person , etc .) . This work was carried out on a corpus of newswire texts in English provided by the Agence France Presse (AFP). In order to extract salient dates that warrant inclusion in an event timeline , we first recognize and normalize temporal expressions in texts and then use a machine-learning approach to extract salient dates that relate to a particular topic . We focused only on extracting the dates and not the events to which they are related . The paper aims to come up with a system that examines the degree of semantic equivalence between two sentences .
At the core of the paper is the attempt to grade the similarity of two sentences by finding the maximal weighted bipartite match between the tokens of the two sentences . The tokens include single words , or multiwords in case of Named Entitites , adjectivally and numerically modified words . Two token similarity measures are used for the task - WordNet based similarity , and a statistical word similarity measure which overcomes the shortcomings of WordNet based similarity . As part of three systems created for the task , we explore a simple bag of words tokenization scheme , a more careful tokenization scheme which captures named entities , times , dates , monetary entities etc. , and finally try to capture context around tokens using grammatical dependencies .

A WORD INFLUENCE

We attempt to understand what text level clues the model captures to perform the ordering task. Some techniques for visualizing neural network models in the context of text applications are discussed in Li et al. (2015a). Drawing inspiration from this work, we use gradients of prediction decisions with respect to the words of the correct sentence as a proxy for the salience of each word.

For each time step during decoding we do the following. Assume the sentence assignments for all previous time steps have been correct. let h be the current hidden state in this setting and $s = (w_1, ..., w_n)$ be the correct next sentence candidate, the w_i being its words. The score for this sentence is defined as $e = f(s, h)$ (See equation 7). The importance of word w_i in predicting s as the correct next sentence is interpreted as $\|\frac{\partial e}{\partial w_i}\|$. We assume h to be fixed and only backpropagate gradients through the sentence encoder.

Table 5 shows visualizations of a few selected abstracts. Words expressed in darker shades correspond to higher gradient norms. In the first example the model seems to be using the word clues 'first', 'second' and 'third'. A similar observation was made by Chen et al. (2016) in their experiments. In the second example we observe that the model has paid attention to phrases such as 'We present', 'We argue' which are typical of abstract texts. The model has also focused on the word 'representation'



(a) τ scores of order predictions on paragraphs of a given length.

(b) Accuracy of predicting the correct sentence at a given position.

Figure 3: Performance with respect to paragraph length and sentence position - NIPS abstracts test data.

appearing in the first two sentences. Similarly in the third example, the words 'salient' and 'dates' have been attended to. In the last example, the words 'token', 'tokens', 'tokenization' have received attention. We believe that these observations link to ideas from centering theory which state that entity distributions in coherent discourses adhere to certain patterns. The model has implicitly learned learned these patterns with no syntax annotations or handcrafted features.

B PERFORMANCE ANALYSIS

Figure 3a shows the average τ for the models on the NIPS abstracts test set for a given paragraph length. The performance of local approaches dies down fairly quickly as we can expect and face difficulties handling lengthy paragraphs. Our model attempts to maintain consistent performance with increasing paragraph size with a more gradual decline in performance.

Figure 3b compares the average prediction accuracy for a given sentence position in the test set. It is interesting to observe that all models fair well in predicting the first sentence. The greedy decoding procedure also contributes to the decline in performance as we move right. Our model remains more robust compared to the other two methods.

Another trend to be observed is that as the context size increases (2 for next sentence generation, 3 for window network, complete sentential history for our model) the performance decline is more gradual.

C MODEL DETAILS

The LSTM update in equation 1 of the paper

$$h_t, c_t = LSTM(h_{t-1}, c_{t-1}) \tag{11}$$

is as follows.

$$i_t = \sigma(W_i h_{t-1} + b_i) \tag{12}$$
$$f_t = \sigma(W_f h_{t-1} + b_f) \tag{13}$$
$$o_t = \sigma(W_o h_{t-1} + b_o) \tag{14}$$
$$\hat{c}_t = \tanh(W_c h_{t-1} + b_c) \tag{15}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t \tag{16}$$
$$h_t = o_t \odot \tanh(c_t) \tag{17}$$

where $W_{\{i,f,o,c\}}, b_{\{i,f,o,c\}}$ are learnable parameters.

The LSTM update in equation 6

$$h_t, c_t = LSTM(h_{t-1}, c_{t-1}, x_{t-1}) \tag{18}$$

is given by the following.

$$i_t = \sigma(W_{hi}h_{t-1} + W_{xi}x_{t-1} + b_i) \tag{19}$$
$$f_t = \sigma(W_{hf}h_{t-1} + W_{xf}x_{t-1} + b_f) \tag{20}$$
$$o_t = \sigma(W_{ho}h_{t-1} + W_{xo}x_{t-1} + b_o) \tag{21}$$
$$\hat{c}_t = \text{tanh}(W_{hc}h_{t-1} + W_{xc}x_{t-1} + b_c) \tag{22}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t \tag{23}$$
$$h_t = o_t \odot \text{tanh}(c_t) \tag{24}$$

where $W_{\{hi,hf,ho,hc\}}, W_{\{xi,xf,xo,xc\}}, b_{\{i,f,o,c\}}$ are learnable parameters.