LEARNING TEXT SEGMENTATION USING DEEP LSTM

Noam Mor, Omri Koshorek, Adir Cohen & Michael Rotman
The Blavatnik School of Computer Science
Tel-Aviv University
Israel
{noammor,omrikosh,adircohen,rotmanmi}@post.tau.ac.il

ABSTRACT

We train an LSTM-based model to predict structure in Wikipedia articles. This results in a model that is capable of segmenting any English text, is not constrained to a limited number of topics, and has much better runtime characteristics than previous methods. Finally, we introduce a new dataset which is much more extensive than current ones, and compare our method with previous methods in terms of segmentation quality and runtime on existing datasets.

1 INTRODUCTION

Text segmentation is the task of dividing a text into segments by textual topic, such that sentences within a segment discuss the same topic, and cutoff points indicate a change of topic. It is often used on long unstructured text, such as the output of automatic speech recognition or long, unstructured text taken from the Internet. Additionally, it might be used in other NLP tasks where knowing the text structure can help improve performance, such as in text summarization.

Prior work tried to either come up with a heuristic to identify whether two sentences discuss the same topic - as in Choi (2000), Glavaš et al. (2016) - or to model topics explicitly and use methods such as LDA to assign a topic from a pre-defined list to a paragraph or a sentence, as in Chen et al. (2009).

We approach the problem from a different direction. Instead of engineering a similarity measure between sentences ourselves, or modeling a textual topic explicitly, we build a deep neural-network model and train it to predict topic changes in naturally-occurring articles taken from Wikipedia.

Our model achieves a result on our test set of 77,000 articles which is comparable to that of other methods achieve on a small amount of natural texts when specifically trained on a pre-defined topic list. Our method results in a single model that can be thought of as a generic model for text segmentation of any English text.

This paper is structured as follows. In section 2 we present related work. Section 3 describes the specifics of our model, and section 4 the training and benchmark data. Section 5 contains segmentation quality evaluations, as well as comparisons with other methods. Section 6 concludes and presents ideas for future work.

2 RELATED WORK

Bayesian approaches are among the well preforming supervised methods for text segmentation. Most of them - such as Chen et al. (2009), (Riedl & Biemann, 2012) - use LDA (or variants thereof) as a generative probabilistic model for text: a document is a sequence of topics, which are sampled from a topics distribution, and each topic imposes a distribution over the vocabulary. Riedl & Biemann (2012) performs the best among this family of methods. It defines a coherence score between pairs of sentences, which measures the distance between two sentences' topic probability vector, and segments by finding drops in coherence scores between pairs of adjacent sentences.

Another noteworthy approach to text segmentation is GraphSeg - Glavaš et al. (2016) - an unsupervised method which builds a *semantic relatedness graph*, and analyzes it to obtain a well-behaved

segmentation. GraphSeg performs competitively on synthetic datasets and outperforms Bayesian approaches on a real world dataset. GraphSeg works by representing the distance between two words as the cosine distance between their embedding vectors. The distance between two sentences is defined to be the cost of the best bipartite matching between the words of the two sentences. A graph is then built where nodes are sentences, and an edge between two sentences signifies that the sentences are semantically similar. The segmentation is then determined by finding maximal cliques of adjacent sentences, and completing the segmentation based on those cliques using a heuristic. GraphSeg also has two hyperparameters that must be tuned on a per dataset basis.

3 Our Approach

3.1 The Model

Our method is heavily based on the LSTM architecture - Gers et al. (1999). Given a document of length k sentences, we compute a sentence embedding vector for each sentence by concatenating the last hidden states of 2-layer bidirectional LSTM run on the sentences' word embeddings. These sentence embeddings then serve as input to another 2-layer bi-LSTM. This gives us k d-dimensional vectors. We then multiply each of the vectors by a $d \times 2$ matrix, trim the last result, and apply softmax to obtain $k - 1$ cutoff probabilities between each two sentences. See Appendix 1 for a diagram.

3.1.1 Model variants

We have tested several variants of the described model:

- Base model: Uses a bidirectional LSTM for both modules, and computes the sentence embeddings as the concatenation of the last hidden state vectors.
- Max sentence embedding: Changes the final sentence embedding to be the result of a maximum operation on the last bi-LSTM output in the sentence embedding module.
- Attention: This model employs an attention mechanism as described in Bahdanau et al. (2014) to calculate the sentence embedding from the LSTM hidden states.
- External sentence embedding: Instead of learning the sentence embedding ,we plugged a pre-trained 4096-dimensional sentence embedding model from Conneau et al. (2017).
- Uni-directional LSTM: Identical to base model, with each bi-LSTM layers replaced with two stacked LSTM layers.

During testing, the base model and the max sentence embedding models outperformed the rest. It is worth noting that the single-directional LSTM model performed very badly compared to all the others.

3.2 Training

Our model predicts for each sentence whether it ends a segment. For each document, we train $k - 1$ binary classification problems by minimizing the negative log-likelihood of the parameters θ. Let y is the predicted segmentation probabilities and $\hat{y}$ the ground truth vector; then our training objective is:

$$\min_{\theta} \sum_{i=1}^{k-1} -\hat{y}_i \cdot \log y_i.$$

Training is done by stochastic gradient descent in an end-to-end manner.

3.2.1 Analysis

The LSTM layers which take as input the sentence embeddings give the model the property of being global, in the sense that any segmentation decision taken by the model may depend on information found in other locations in the document.

The sentence embedding model is trained specifically for the task of text segmentation. As such, one could presume that it would output an embedding vector that contains information about the topics discussed within the sentence, as well as information about formulations that often start or end a segment. Whereas other works are concerned with the exact methods used to detect a change of topic, we let our deep neural network decide ways to solve this problem by itself.

3.3 Inference

During test time, the model takes a sequence of word embeddings divided into sentences, and returns a vector p of cutoff probabilities between sentences. We employ greedy decoding - that is to say, we segment where p_i is greater than some threshold τ. The threshold τ is thus a parameter that needs to be tuned according to the desired average segment length - for example, if the desired segment length is 10, τ should be the value at the 90th percentile in p. In practice, we set $\tau = 0.3$ in all of our experiments, as we have found it to produce reasonable segmentation results.

3.4 Technical Details

For word embeddings, we use the GoogleNews word2vec pre-trained model - Google (2013). The sentence embedding model is a 2-layer bidirectional LSTM with 256 units each, which makes for a 1024-dimensional sentence embedding vector in the base model. The top-level LSTM is a 2-layer bidirectional LSTM with 256 units each.

The system is written in the PyTorch[1] framework and trained on a single nVidia Titan X GPU for 36 hours.

4 Data

For this work we have created a new dataset, which we name Wiki-819k. It is a collection of 819,000 English Wikipedia articles[2], and their segmentation, as it appears in their table of contents. As preprocessing, we filtered out articles with less than three segments, and removed the first segment from all articles, as it usually is a summary text that discusses many topics. We view this data as very fitting for the text segmentation task, since it is natural, open-domain, and it has a single, well-defined segmentation. Moreover, neural network models often benefit from a wealth of training data, and our dataset can easily be expanded up to two million articles with very little effort.

4.1 Existing Datasets

Existing text segmentation datasets are much smaller in size and variety than Wiki-819k. The most common dataset for evaluating performance on the text segmentation task was created by Choi in Choi (2000). It is a synthetic dataset, where each document is a concatenation of chunks of texts from 10 different documents. Glavaš et al. (2016) created a dataset of their own, which consists of 6 political manifestos from the Manifesto project [3]. The texts were manually segmented by first classifying each sentence to one of 7 pre-defined topics, followed by joining adjacent sentences classified to the same topic into segments. Chen et al. (2009) also use English Wikipedia documents to evaluate text segmentation. They define two datasets, one with 100 articles about major cities and one with 118 articles about chemical elements. During evaluation, they rely on the intuition that articles about cities will have similar topics in a similar order.

5 Evaluation

We compare our method to other methods in terms of segmentation quality and runtime. We measure our runtime when running on a mid-range laptop CPU, and when running on a nVidia Titan X GPU.

[1] Facebook (2017)

[2] Wikipedia Dump (2017)

[3] https://manifestoproject.wzb.eu

Table 1: Datasets details

	Wiki-819k	Choi	Manifesto
Documents	819000	700	6
Real-world	✓	✗	✓
Large Variety of Topics	✓	✗	✗

Table 2: P_k Results (percent)

	Wiki-819k	Wiki-103	Choi	Manifesto	Wiki-Cities	Wiki-Elements
Chen et al. (2009)[4]					22.2	**22.1**
GraphSeg		48.68[5]	**5.6-7.2**	**28.09**	40.09[6]	
Ours (LSTM State)	24.9	26.6	18.8	47.77	20.5	32.3
Ours (Max Output)	**24.27**	**23.8**	20.2	48.43	**19.5**	31.5
Random baseline	51.30		47.74	40.6		

5.1 Segmentation Quality Metric

We evaluated our performance using the P_k metric defined in Beeferman et al. (1999). P_k is the probability that two randomly-chosen sentences k sentences apart, will be incorrectly detected as either belonging to the same segment or belonging to different segments. Following Glavaš et al. (2016), we set k to half the average segment size in the ground-truth segmentation.

5.2 Comparisons

It is difficult to compare our open-ended segmentation method with other methods, which may require a pre-defined list of topics, that all articles would have the same topics, or have hyperparameters that must be tuned per single example. Different methods pre-process data differently, and datasets may have idiosyncrasies - for example, GraphSeg measure their performance on Manifesto using the P_k measurement, but the Manifesto datasets defines that "sentences" are often actually parts of a sentence. Since our system has only ever seen complete sentences, our system is not competitive on this benchmark. We attempt a comparison nonetheless in Table 1.

For comparisons, we define a small dataset of 103 Wikipedia articles. We evaluate our own model on Choi, Manifesto, Wiki-103 and the test set in Wiki-819k (some 77,000 documents), and the two small wikipedia datasets introduced Chen et al. (2009) .

We also evaluate the two best-performing variants of our method: the basic setting, where the sentence embedding is a concatenation of the last states of bi-LSTM layers, and the maximum sentence embedding model. In addition, we evaluate the performance of a random baseline model, which starts a new segment with probability 0.5 every k sentences, where k is half of average segment size.

5.2.1 Analysis

Comparing our method to GraphSeg, we can see that GraphSeg gives a much better result on the synthetic Choi dataset. However, their success on the Choi dataset does not carry over to natural Wikipedia data, where they achieve a result only slightly better than a random baseline. We explain this by noting that since Choi is a synthetic dataset created by concatenating unrelated articles, even a simple word counting method as in Choi (2000) can achieve reasonable success. GraphSeg uses a similarity measure between word embedding vectors to surpass the word counting method, but in a real setting, considering the similarity of words in a direct manner is not a good method. At the world level, two articles concerning completely different topics are much easier to differentiate than two sections in one article .

[4]Best parameters

[5]With default parameters

[6]With default parameters

We compare our method to Chen et al. (2009) on the two small Wikipedia datasets from their paper. Our method beats theirs on Wiki-Cities and obtains a worse result on Wiki-Elements, where presumably our word embeddings were of lower quality, having been trained on Google News, where one might expect that few technical words from the domain of Chemistry have been used. We consider this result convincing, since we do not require that all articles have similar sections and order, and did not train specifically for these datasets, while still showing competitive performance.

5.2.2 Run Time

Our method's runtime is linear in the number of words and the number of sentences in a document. Conversely, GraphSeg has a much worse asymptotic complexity of $O(N^3 + V^k)$ where N is the length of the longest sentence, V the number of sentences, and k the largest clique size. Moreover, neural network models are highly parallelizable, and can be run on GPUs.

This results in our method being much faster than GraphSeg. Measurements are found in 3. Note that we could not reasonably run GraphSeg on the entire test set of Wiki-819k. Instead we measured the total run-time on 50 articles from English Wikipedia.

Table 3: Run time in seconds

	Wiki-819k	Wiki50	Manifesto
Ours (GPU)	1641	2	3
Ours (CPU)		135	131
Graph-Seg		3480	3042

6 Conclusions and Future Work

We believe the approach we have shown shows great promise on real-life data. It is only trained once on Wikipedia data, does not require domain-specific training, and results in a single model that is versatile enough to tackle generic English text and has very good runtime on a GPU. It has shown to be accurate on a large variety of real-world text, even outperforming Chen et al. (2009) on one of the datasets they have trained specifically on. We also vastly outperform Glavaš et al. (2016), a prominent method that is not restricted to a set number of topics, on natural Wikipedia data.

6.1 Further Improvements

Simply enlarging the model might improve it significantly - the model we trained has 2.4 million parameters, while deep learning models typically have 100 times as many. Training a larger model is a little more challenging, since a larger model's full forward-backward pass on a large Wikipedia article requires more memory than current GPUs have.

Setting τ to obtain a specific average segment length may also obtain better results than simply setting it to 0.3. Another option is to abandon greedy decoding and attempt to enforce heuristic rules on the output segmentations using a global decoding method.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Doug Beeferman, Adam Berger, and John Lafferty. Statistical models for text segmentation. *Machine learning*, 34(1):177–210, 1999.

Harr Chen, SRK Branavan, Regina Barzilay, and David R Karger. Global models of document structure using latent permutations. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 371–379. Association for Computational Linguistics, 2009.

Freddy YY Choi. Advances in domain independent linear text segmentation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pp. 26–33. Association for Computational Linguistics, 2000.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*, 2017.

Facebook. Pytorch deep learning framework, 2017. URL `https://github.com/pytorch/pytorch`.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

Goran Glavaš, Federico Nanni, and Simone Paolo Ponzetto. Unsupervised text segmentation using semantic relatedness graphs. Association for Computational Linguistics, 2016.

Google. Google pre-trained word2vec model, 2013. URL `https://code.google.com/archive/p/word2vec/`.

Martin Riedl and Chris Biemann. Topictiling: a text segmentation algorithm based on lda. In *Proceedings of ACL 2012 Student Research Workshop*, pp. 37–42. Association for Computational Linguistics, 2012.

Wikipedia Dump. English wikipedia dump, 2017. URL `https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2`. [Online; accessed 1-July-2017].

APPENDIX: MODEL DIAGRAM

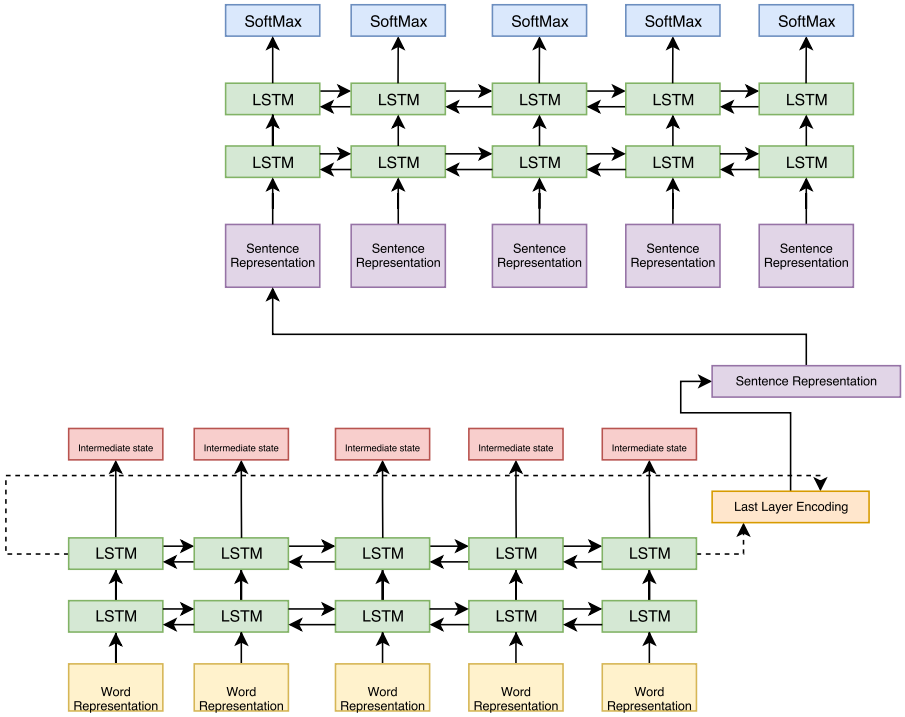

Figure 1: Our model consists of two modules. The bottom half in the above figure shows the sentence encoding module. It consists of two bidirectional LSTM Layers, and its output is determined by the LSTM's last hidden states or output vectors. The second part of our model takes the sentence representations and predicts segmentation probabilities. It consists of two bidirectional LSTM layers, and a linear projection and softmax to obtain a 2 dimensional probability vector for each sentence.