Training Segmentation Models for Extractive and Generative NLP Tasks with Reinforcement Learning

Akash Bharadwaj
akashb@andrew.cmu.edu

Chaitanya Ahuja
cahuja@andrew.cmu.edu

Abstract

Various NLP tasks arguably involve a segmentation task. Text Summarization involves finding key pieces of information in a source document that need to be included in the summary. Extractive Text comprehension involves selectively locating contiguous spans of answers within a source document in response to a question. Phrase Based Machine Translation (which often out-performs neural MT in low data scenarios) involves finding phrases in the source language that are translated as a whole. In this project we investigate how Stack LSTMs [9] can be used to achieve this segmentation. For many tasks however, this segmentation is unobserved. We propose to use reinforcement learning to allow Stack LSTM segmenters to be trained in an unsupervised fashion to optimize performance in an end task of interest.

1 Introduction

Stack LSTMs [9] are a control structure for sequential, decision making processes that have become popular in NLP tasks. They involve at least two data structures: a buffer of elements yet to be processed, and a stack of elements affected by an action that must be chosen by the Stack LSTM. A transition set (set of possible actions) determines how elements are moved from the buffer to the stack and how stack elements are affected by an action. The transition set itself can vary from task to task. For a task like extractive text comprehension or summarization, the transition set would include actions to move an item from the buffer to the stack, retain all current stack items for the downstream task, or to discard all stack elements. For a task like translation, the transition set could instead include operations to move a word from the buffer to the stack or translate the current stack contents and append it to the translation so far (thus building the translation left-to-right).

As noted in the abstract, all of these tasks call for segmentation in some form even though the segmentation is not explicitly observed. This poses a problem for stack LSTMs since they need explicit supervision for this by means of an oracle [9]. To deal with the lack of an oracle, we propose to use the REINFORCE algorithm [20] to replace direct supervision with a reward maximization objective instead, where the reward is primarily determined by a downstream tasks where the segmentation output is consumed. An additional challenge is that some tasks such as summarization further enforce a sparsity constraint on the segmentation, i.e., most input is irrelevant to the summary and should be discarded. We also address this by introducing parsimony penalties and allowing the user to induce a bias to the segmentation process by incentivizing intuitively favorable segments (eg. retaining named entities, numbers/dates, noun phrases etc.) This achieves all of the following:

1. Proposes a general method for unsupervised training of stack LSTMs in the absence of an oracle
2. Allows the model designer to debug the model by altering the reward structure of the reinforcement learning problem to bias segmentation correctly and penalize avoidable errors, based on insights about the end task.
3. Provides a sparse alternative to attention for tasks that involve verbose input that actually reduces the input size, thus minimizing the quadratic complexity associated with attention mechanisms.
4. Makes the model's 'rationale' evident due to the explicit connection between segmentation and the end task.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

2 Prior Work

The review of prior work is organized into 2 sections: Review of prior work for summarization and text comprehension (tasks we plan to use for evaluation), and a review of stack LSTM literature.

Text summarization has been standardized around the DUC competitions [1] since 2001 and a variety of methods have been proposed. Banko et al. [3] and Wubben et al.[22] borrowed ideas from phrase based MT for summarization. Zajic et al.[23] merged linguistically motivated source transformation with a topic detection algorithm to identify key phrases. Cohn and Lapata[6] support more arbitrary transformations via tree transduction rules extracted from aligned parsed texts. Woodsend et al. instead use context free parses and dependency parses with a quasi-synchronous grammar approach to produce legible summaries. More recently, neural seq2seq models have become popular. Rush et al. [18] obtain competitive results using an attentional seq2seq neural model to summarize single sentence inputs. Cheng and Lapata [5] and Nallapati et al. [13] proposed neural models for scenarios when the input involves multiple sentences, by incorporating elaborate and expensive attention mechanisms over all of the input.

Reading comprehension has evolved over the past decade from an extractive[11] to a generative [14] one. Consider a tuple $(d, q, a, \mathcal{C})$ where d is the document, q is a query, a is the answer and $\mathcal{C}$ is the set of all the candidate answers. For extractive tasks, it is assumed that for $\forall c \in \mathcal{C}$ at least one of the tokens can be found in the document d. The task is to find the answer a from the document d which answers the given query q. In contrast to this, generative reading comprehension does not have $\mathcal{C}$ and the answer a answers the question q based on the document d with a free-form sentence composed of words from the vocabulary of the language model. Deep models have easily outweighed shallow methods involving handcrafted features, hence we will take deep models for reading comprehension in literature as baselines. There are broadly two models to tackle the extractive problem (1) Attention Models [4, 8]: These focus on different parts of the document at different times (2) Multiple-Hop Methods [19]: They refine the representation of tokens while training. Both these approaches can be directly extended to a generative model by using a encoder-decoder architecture.

Unlike phrase-based approaches and answers based on handcrafted features, neural models are not easily interpretable and attention mechanisms can often behave unintuitively. Even though the numerical scores for neural models are state of the art, the designer does not have a mechanism to control over the how the model decides to attend to its input. Moreover, such attention mechanisms incur a quadratic time penalty which can be significant for verbose input. Lastly, the task of reading comprehension is inherently compressive and involves discarding the bulk of the input and retaining only what is most relevant. Being able to do this competently would not only reveal the rationale of a summarization model and make it interpretable, but also make it significantly faster. These are the gaps in prior work that we seek to address.

Stack LSTMs [9] were originally proposed by Dyer et al. for the task of dependency parsing where they obtained state of the art results. They have since been used for a variety of other tasks such as Named Entity Recognition [12], constituency parsing [7] [10] and language modeling [10]. As noted before, in each of these tasks it is trivial to design a deterministic oracle that explicitly supervises the sequence of decisions made by the stack LSTM. To the best of our knowledge, no attempt has been made to train stack LSTMs without such supervision.

3 Methods

As depicted in figure 1, the overall architecture consists of two parts: a segmenter (stack LSTM) and a generator (seq2seq). For the purpose of explanation, let use consider the task of summarization with the following simple document and its corresponding ideal summary:

Document: "The **chinese president Xi Jinping** arrived in Washington yesterday and **met president Obama**. After a long day of constructive dialog between the two premiers, they **announced the need for strong ties** between the two nations."

Summary: "The Chinese and American presidents agree to strengthen ties"

The key phrases in the document are highlighted in bold. Ideally, the segmenter should extract them and the seq2seq generator ensures the final summary is a fluent composition of the relevant phrases extracted by the segmenter. Note that the generated summary is not a simple concatenation of phrases from the input.

The model works in a step-wise fashion. The stack LSTM is initialized with an empty stack (S), empty action history (A) and a buffer (B) full of words in the document. The transition set of the stack LSTM restricts the

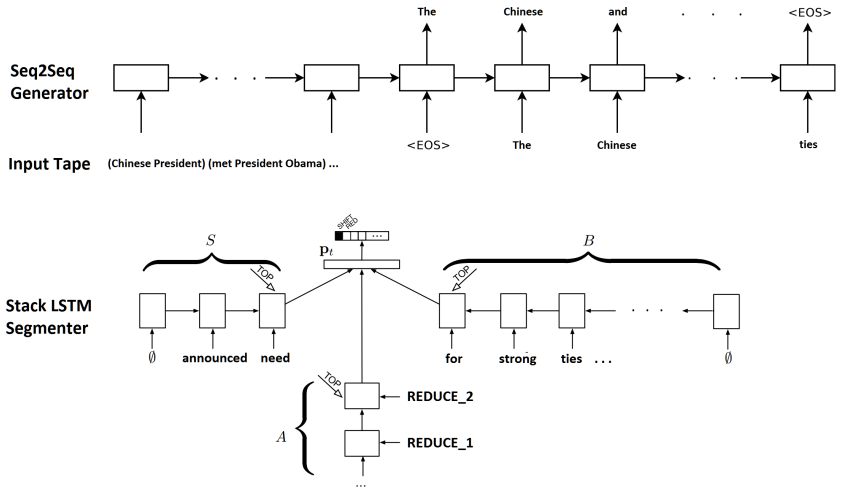

Figure 1: Model Schematic

number of possible actions at each step. This is an advantage of stack LSTMs since we have direct control over the action space through the transition set. For the task of summarization, the transition set has exactly 3 actions:

1. SHIFT: Moves the top of B to the top of S
2. REDUCE_1: Appends all stack elements to the input tape and pops them off the stack S
3. REDUCE_2: Discards all stack elements

At each time step the stack LSTM chooses to perform one of these 3 actions. Since each word can be pushed onto the stack only once and can be reduced only once, the document is parsed in linear time. As the document is parsed by the stack LSTM segmenter, the input tape is populated with a sequence of key phrases. The next step is for the seq2seq generator to consume the input tape and decode a fluent summary.

At train time, we have the additional task of learning parameters. The seq2seq generator can be trained in a supervised fashion using a typical categorical cross entropy loss. The segmenter cannot be trained in this fashion due to the lack of relevant annotations in the data. Thus, we use the REINFORCE algorithm[20]. The intuition is that better segmentations lead to better summaries. In the extremal case, a segmenter that discards all input will lead to the same summary for any input which is clearly sub-optimal. Thus, the metric for the downstream task (eg. BLEU, ROUGE, METEOR) can be used as reward for the REINFORCE algorithm.

As noted in [17], training parameters from scratch using REINFORCE can result in catastrophic divergence while training, especially if rewards are sparse and have high variance. In [17], this is addressed by seeding the parametric model with MLE parameters obtained by training on a small, suitably annotated training set. We propose a different strategy. We train the segmenter to initially mimic a naive policy of retaining all inputs. This means a regular seq2seq model (such as the one used in [18]) is the fall back option and any change in the segmenter's behavior should lead to an improvement. Furthermore, the seq2seq generator is pre-trained to consume all input and generate a summary. By using this kind of a curriculum during training, we can hopefully avoid divergence during optimization.

Another aspect worth talking about in more detail is the rewards themselves. The obvious reward is the score (higher being better) obtained in the downstream task of interest. Unlike most applications of reinforcement learning where it is possible to take thousands of actions before any reward is seen, for a sentence with N words, we take at most 2N actions until a reward is obtained. This coupled with a small transition set for the stack LSTM should aid in learning the optimal policy.

In addition to the score obtained in the downstream task, we can provide more fine grained intermediate rewards to induce specific biases in the segmenter based on error analysis or human insight, without actual supervision. For example, we can incentivize the retention of named entities, noun phrases etc. Similarly, we can penalize negative segmentation behaviours such as verbosity/retention of the whole input (after the policy initialization phase), retention of stop words etc.

3.1 Sequence to Sequence Model

To demonstrate the effectiveness of the approach described in the paper, we use a state-of-the-art sequence to sequence model for generation. Based on the quality of the generated outputs, we quantify rewards which are used to estimate the policy that decides the actions of the segmenter. In this section, we describe the formulation of sequence to sequence model used for our experiments.

Consider a model consisting of an encoder, denoted by $e(.)$, and a decoder, denoted by $d(.)$. These encoders and decoders are composed of LSTM cells. Also, with the sequence of inputs $x_1, x_2, \dots x_T$ which represent the document in the summarization task, we can write the transformations as

$$[z_1, z_2, \dots, z_T], m = e(x_1, x_2, \dots x_T), \tag{1}$$

where $z_1, z_2, \dots z_T$ are the hidden state outputs of the encoder, T is the number of time steps in the encoder and m is the memory states at the final time-step. These memory states will be used as the starting memory of the LSTM cell of the decoder. Hence,

$$[\hat{y_1}, \hat{y_2}, \dots, \hat{y_R}] = d([y_1, y_2, \dots, y_R], m) \tag{2}$$

where $y_1, y_2, \dots y_R$ are the ground-truths (gold-label summaries) and $\hat{y_1}, \hat{y_2}, \dots, \hat{y_R}$ are the predicted values by the decoder.

Clearly, the encoding m is the only vector that contains information of the documents, and with increasing number of time steps, a lot of information gets lost in LSTM cells because of gated transformations. Hence, it is a good idea to make use of the hidden state outputs $(z_1, z_2, \dots z_T)$ of the encoder. This approach is called attention [2] modeling.

The approach described by Bahdanau et al. [2] uses a generic function approximator $g(.)$ which quantifies the importance of all $z_1, z_2, \dots z_T$ in form of weights,

$$\alpha_{ij} = g(z_i, y_j) \tag{3}$$

where $\sum_{i=1}^{T} \alpha_{ij} = 1$. α_{ij}'s are used as the weights for z_i's. The resulting embedding is concatenated with y_j and used inputs to the decoder.

$$a_j = \sum_{i=1}^{T} \alpha_{ij} z_i \tag{4}$$

where the new input for the decoder is $[y_j, a_j]$.

It is interesting to note that any good approximator can be used to estimate the importance of the sequence of input words. In our case, for a fixed j and $i = 1, 2 \dots T$,

$$\alpha'_{ij} = g(z_i, y_j) =< z_i, y_j >, \tag{5}$$

where <.,.> denotes dot product.

$$\alpha_{1j}, \alpha_{2j}, \dots, \alpha_{Tj} = \text{softmax}\left([\alpha'_{1j}, \alpha'_{2j}, \dots, \alpha'_{Tj}]\right) \tag{6}$$

We use dot-product as the function approximator as it forces the network to learn correlation between encoder and decoder embeddings.

3.2 Stack LSTM

In this section, we specify the mathematical details of the stack LSTM model. It is trained using two different techniques:

3.2.1 Imitation Learning

Imitation learning is used to pre-train the stack LSTM. In this setting, an oracle (expert) that executes the naive policy perfectly is used. We first define some entities

1. Segmenter state at before the t^{th} action:

$$p_t = relu\{W[s_t; b_t; a_t] + d\} \tag{7}$$

where W is a learned parameter matrix, d is the corresponding vector of biases and $s_t,\ b_t,\ a_t$ are the tops of the stack, buffer and action history respectively.

2. Probability of the t^{th} action z_t:

$$p(z_t|p_t) = \frac{\exp\left(g_{z_t}^T p_t + q_{z_T}\right)}{\sum_{z' \in \mathcal{A}(S,B)} \exp\left(g_{z'}^T p_t + q_{z'}\right)} \tag{8}$$

where g_z is an embedding corresponding to an action z from the valid transition set given the current contents of stack S and buffer B ($\mathcal{A}(S, B)$). q_z is the corresponding bias term.

Since we use LSTMs to encode the contents of the buffer, stack and history of actions (refer to Dyer et. al [9] for details) in the segmenter state p_t, we are not making any independence/markovian assumptions when we define the probability of an action sequence Z (given all the input W) as:

$$p(Z|W) = \prod_{t=1}^{|Z|} p(Z_t|p_t) \tag{9}$$

During imitation learning, the oracle/expert prescribes the sequence of actions Z. Training the stack LSTM to mimic the expert involves maximizing the likelihood of this sequence, which is p(Z|W). To do so, we simply apply a categorical cross entropy loss at each action prediction step. This yields the following loss which we minimize:

$$\text{Loss to minimimize } = -\sum_{t=1}^{|Z|} (\log(p(z_t|p_t))) \tag{10}$$

3.2.2 REINFORCE

The imitation learning procedure is merely a pre-training step. The next step is to use REINFORCE to learn a better policy than the naive policy. For this, we modify the approach proposed in prior work by Ranzato et. al [17]. Since the stack LSTM has been trained to mimic a naive policy (which is deterministic), we instead use an epsilon greedy policy that falls back to the stack LSTM's policy with probability $1 - \epsilon$ and selects an action uniformly with probability ϵ. This encourages exploration of policies very different from the naive one.

Let r_t denote the discounted future reward for the t^{th} action z_t based on a segmentation obtained using the ϵ-greedy policy followed by summary generation via the sequence to sequence model using the generated segmentation. This future reward can include the log likelihood of the gold summary as well as more fine grained rewards such as the number of target summary words retained by a reduce operation, parsimony penalties for retaining too many words etc.

Let G be the gold summary (sequence of words) and Z be a segmentation generated by following the ϵ-greedy policy. P(Z|W) as defined in equation 9. P(G|Z) is given by the sequence to sequence model.

$$p(G, Z|W) = p(Z|W) * p(G|Z) \tag{11}$$

$$p(G, Z|W) = \left(\prod_{t=1}^{|z|} p(z_t|p_t)\right) * p(w_1^g, ..., w_T^g) \tag{12}$$

if the segmentations were observed in the dataset, we could directly optimize this for the gold segmentation Z to obtain MLE parameters for the stack LSTM, and optimize for the gold summary G to obtain MLE parameters for the sequence to sequence model, jointly. However, we do not observe segmentations. We only know G. Thus, assuming any arbitrary performance metric r(Z,W,G) (higher being better), we define the following loss:

$$L = -\sum_{\forall Z} p(G, Z|W) * r(Z, W, G) \tag{13}$$

$$\tag{14}$$

This is essentially the negated expected reward. The sum over all possible segmentations Z is intractable, so we use a point estimate. Following Ranzato et. al [17], the gradient of the loss L with respect to stack LSTM parameters θ is:

$$\frac{\partial L}{\partial \theta} = \sum_{t=1}^{|Z|} \left(\frac{\partial L}{\partial o_t} * \frac{\partial o_t}{\partial \theta} \right) \tag{15}$$

where o_t is the input to the softmax at each action prediction step of the stack LSTM.

$$\frac{\partial L}{\partial o_t} = r(Z, W, G) * (p(z_t|W) - 1(z_t)) \tag{16}$$

where the second term in the product is the gradient of softmax. In practice, we apply the variance reduction technique where the reward term is mean centered around the average reward obtained for the t^{th} action (r_t):

$$\frac{\partial L}{\partial o_t} = (r(Z, W, G) - r_t) * (p(z_t|W) - 1(z_t)) \tag{17}$$

We use SGD to minimize the loss using these gradients. Intuitively, this is incentivizing the stack lstm to prefer actions that lead to positive net rewards. The use of the baseline is important, since we don't want to incentivize all actions that leads to positive rewards, but only those that result in an improvement over the average.

We estimate r_t using a parametric regression model using the segmenter state and simultaneously train it using a simple MSE loss. Special care is taken to ensure the MSE loss doesn't back-propagate into the stack LSTM parameters.

4 Dataset

GigaWord for summarization [18] - This is a collection of News articles modified to suit the summarization task. The first line of each article is used as a descriptive sentence and the corresponding headline is treated as the summary of that sentence. This claim is reasonable as the first sentence of an article generally describes the headline in more detail. There are over 3.8 million train examples, while we use around 190 thousand examples for validation.

5 Experimental Design

5.1 Experiments

Prior to the full fledged experiments on summarization/text comprehension, we have run experiments to individually verify the two components of our model, namely the stack-lstm and the seq2seq model. We evaluate them separately on two different tasks which they are individually suitable for. The stack LSTM is evaluated on a dependency parsing task. Training and evaluation is done on a subset of the Penn Treebank annotated with dependency arcs and labels. The seq2seq model is evaluated on a summarization dataset extracted from the English Gigaword corpus, as described in [18]. The summarization dataset can be found at http://opennmt.net/Models.

5.1.1 Results

For the stack LSTM evaluation on dependency parsing, we benchmark against MaltParser [15], which is the most famous off the shelf dependency parser and has achieved state of the art results on multiple languages. For the benchmark below, it is run out of the box using POS tags and various other engineered features. The stack LSTM is trained using a procedure similar to that described in [9] with a few exceptions. We use the GLOVE pretrained word embeddings [16] and we use single layer LSTMs for encoding the stack, buffer and action history. No hand engineered features other than POS tags are used.

Model	**LAS**	**UAS**
Malt Parser	82.73%	79.86%
Stack LSTM	**83.08%**	**79.92%**

Table 1:

The results show that our stack LSTM implementation is correct and is better than a competitive off the shelf parser on English.

We implemented sequence to sequence model with one layer of LSTM for both encoder and decoder. Instead of using pre-trained embeddings, like in the case of StackLSTM, we learn the embeddings from scratch. The model was trained, with supervision, on the task of text summarization and a procedure similar to that described in [18] to ensure a fair comparison. The processed version of GigaWord dataset was used, which already had '<unk>' tokens and the digits were replaced by '#'. Hence the vocabulary was not truncated any further.

Model	ROUGE-1	ROUGE-2	ROUGE-3	ROUGE-L
Seq2Seq	**26.89**	**8.08**	2.55	**24.78**
Seq2Seq w/ Segmenter	25.31	7.18	**2.60**	23.41

Table 2:

The results shown in table 2 are the scores on the validation set. As we can see that the high ROUGE-1 and ROUGE-L scores imply that the generated summaries have a lot of uni-grams in common, while ROUGE-2 and ROUGE-3 scores are on the lower side. The performance will improve with the inclusion of attention which we will incorporate in the final model.

On comparing the seq2seq model with one that uses the segmenter, we see that the segmenter leads to an improvement in ROUGE-3 while all other ROUGE scores are slightly lower. We attribute this to the fact that the REINFORCE implementation is currently slow and only about 1000 samples have been used for training. Thus, the segmenter has not explored enough samples to arrive at a good, generally applicable policy via REINFORCE. On running REINFORCE for longer and for more samples, we expect to see learning curves where there is an initial decrease in performance (due to exploration), followed by an eventual improvement (exploiting a good discovered policy). Another possible reason is that the initial exploration is too random. Currently, we use ϵ = 0.5 annealed down to 0.05 over the first million samples. It might be better to start at ϵ = 0.05 (or even 0 for no forced exploration) and let the segmenter deviate from its naive policy as necessary.

We have added randomly picked (not cherry-picked) examples in Table 3 which show that the generated models make grammatical sense for most cases. In some cases like Row 9, the resulting summary is different from the ground truth but makes complete grammatical and semantic sense.

5.2 Qualitative Analysis

It was interesting to see the generated summaries on the validation dataset. We have listed some of those in Table 3. Example 1 shows a simple summary for the document at hand and is very similar to the reference summary. This document was is unambiguous and does not require a lot of contextual information, hence was probably easy to generate. While, example 2 is much more complicated. The summary had the country 'Mongolia' which is not mentioned in the document itself. On closer inspection, it is seen that 'ulan bator' which is mentioned in the document is the capital of Mongolia. This is really interesting as the model figured out the country based on a city in the original text. In example 4, the generated summary is 'british coach claims era of conspiracy' which is in-line with the document. In fact the word 'conspiracy' does not even turn up in the document, but the model is able to understand the context and predict a word describing the whole scenario. Similarly, in example 5, there is no reference of 'genetic' in the document, but based on the words like 'chemical' and other related word, 'genetic' was predicted. Even, though it is does not correspond very well to the original document, the summarization aspect of the model performs reasonably well. In example 9, the summary is inherently wrong as the subject of the sentence was switched from ('Britain') to the other country ('Sudan') mentioned in the text. Finally, example 8 is very interesting as the summary is 'thailand to beef up foreign beef ban' as the sentence makes complete sense, even after the well known problem of cyclic repetition in the sequence to sequence models. Although, it is unclear, if the generation is such because it makes semantic sense, or cycle due to a recurrent neural network.

6 Conclusions

In this project, we implemented stack LSTMs and sequence to sequence models for summarization. We evaluated them individually on dependency parsing and summarization respectively. We proposed a way by which the training of stack LSTMs for segmentation tasks can be done without explicit supervision (using imitation learning followed by REINFORCE) and performed experiments to demonstrate that performance is comparable to baselines (and beats the baseline on ROUGE-3). We are currently running summarization experiments to monitor long term convergence behavior and will eventually run more insightful qualitative analysis to interpret the segmenter outputs one it has converged to a good policy.

As future work, we would like to apply the proposed approach to machine translation, where phrase segmentation is likely to be useful, as demonstrated by phrase based machine translation systems which are still state-of-the-art for many languages.

Table 3: Generated Summaries by Vanilla seq2seq model

	Article	Headline(Ground Truth)	Model Generated Summarie
1.	kuwait stock exchange index closed at #,###.# points tuesday , #.# points up from monday 's finish .	kuwait stock exchange index up	kuwait stock exchange closes higher
2.	visiting chinese president hu jintao left here on wednesday for ulan bator to continue his first overseas trip as chinese head of state after winding up his two - day state visit to kazakhstan . .	chinese president leaves kazakhstan for mongolia	chinese president hu leaves for mongolia
3.	hollywood is planning a new sequel to adventure flick " ocean 's eleven , " with star george clooney set to reprise his role as a charismatic thief in " ocean 's thirteen , " the entertainment press said wednesday .	hollywood shores up support for ocean 's thirteen	us hills to run for oceanic ocean marathon
4.	britain 's top swimming coach bill sweetenham , who was cleared by an independent investigation of bullying members of the british team , still has questions to answer his employers when he returns at the end of the month .	cleared,facing another grilling from british swim bosses	british coach claims era of conspiracy
5.	basf , the world 's biggest chemicals company , prefers that its planned takeover of us firm engelhard was a friendly rather than a hostile move , basf chairman juergen hambrecht said in a newspaper interview published on thursday .	basf prefers friendly takeover of us firm engelhard	basf has leading genetic aid research maker
6.	several thousand people gathered on wednesday evening on the main square in zagreb for a public draw and an open air party to celebrate the croatian capital 's second chance to host the women 's slalom world cup .	thousands of croatians celebrate before world cup slalom	zagreb fans celebrate international storm
7.	the philippine government said thursday it wants a " swift resolution " of journalists ' murders , after a global press watchdog said the country was the world 's deadliest place for journalists next to iraq in #### .	philippines vows swift resolution of press murders	philippine govt looks to truth journalists
8.	thailand will discuss lifting its ban on us beef , imposed two years ago over mad cow fears , when negotiators from the two nations meet next week for free trade talks , health officials said thursday .	thailand may lift ban on us beef	thailand to beef up foreign beef ban
9.	britain 's un envoy on wednesday urged stronger international support ,,including greater eu funding , for the african union -lrb- au -rrb-,peacekeeping mission in sudan 's troubled darfur region to improve,security on the ground .	britain urges stronger international support for au in darfur	sudan urges un peacekeeping operations in darfur
10.	us president george w. bush defied congress again wednesday as he placed a slew of controversial political allies in key defense and foreign policy posts in his administration by circumventing the requisite approval process in the senate .	bush defies congress names defense foreign policy posts	bush takes on triumph as republicans face dismissal

References

[1] Document understanding conference. *EMNLP*, 2016. URL `http://duc.nist.gov/`.

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

[3] Michele Banko, Vibhu O Mittal, and Michael J Witbrock. Headline generation based on statistical translation. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 318–325. Association for Computational Linguistics, 2000.

[4] Danqi Chen, Jason Bolton, and Christopher D Manning. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*, 2016.

[5] Jianpeng Cheng and Mirella Lapata. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*, 2016.

[6] Trevor Cohn and Mirella Lapata. Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 137–144. Association for Computational Linguistics, 2008.

[7] James Cross and Liang Huang. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. *EMNLP*, 2016.

[8] Bhuwan Dhingra, Hanxiao Liu, William W Cohen, and Ruslan Salakhutdinov. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549*, 2016.

[9] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. Transition-based dependency parsing with stack long short-term memory. *ACL*, 2015.

[10] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. Recurrent neural network grammars. *arXiv preprint arXiv:1602.07776*, 2016.

[11] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015.

[12] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *NAACL*, 2016.

[13] Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

[14] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

[15] Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135, 2007.

[16] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[17] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

[18] Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

[19] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[20] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[21] Kristian Woodsend, Yansong Feng, and Mirella Lapata. Generation with quasi-synchronous grammar. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 513–523. Association for Computational Linguistics, 2010.

[22] Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 1015–1024. Association for Computational Linguistics, 2012.

[23] David Zajic, Bonnie Dorr, and Richard Schwartz. Bbn/umd at duc-2004: Topiary. In *Proceedings of the HLT-NAACL 2004 Document Understanding Workshop, Boston*, pages 112–119, 2004.