

# Team 3 - Topic summarization

<b>Summary</b>	In this codelab, you'll build a Progressive Web App, which loads quickly, even on flaky networks, has an icon on the homescreen, and loads as a top-level, full screen experience.
<b>URL</b>	your-first-pwapp
<b>Category</b>	Web
<b>Environment</b>	web, kiosk, io2016, pwa-dev-summit, pwa-roadshow, chrome-dev-summit-2016, io2017, typtwd17, gdd17, cds17, io2018, tag-web, jsconfeu, devfest18
<b>Status</b>	Draft
<b>Feedback Link</b>	<a href="https://github.com/googlecodelabs/your-first-pwapp/issues">https://github.com/googlecodelabs/your-first-pwapp/issues</a>
<b>Author</b>	petele

---

## Introduction

Text summarization is the task of generating a headline or a short summary consisting of a few sentences that captures the salient ideas of an article or a passage. The main idea behind automatic text summarization is to be able to find a short subset of the most essential information from the entire set and present it in a human-readable format. As online textual data grows, text summarization have potential to be very helpful because more useful information can be read in a short time.

### What is the need for text summarization?

- Today, our world is parachuted by the gathering and dissemination of huge amounts of data. Thus, there is need to develop machine learning algorithms that can automatically shorten longer texts and deliver accurate summaries that can fluently pass the intended messages.

- Applying text summarization reduces reading time, accelerates the process of researching for information, and increases the amount of information that can fit in an area.

## **What are the main approaches to text summarization?**

There are two broad approaches to summarization:

- Extraction-based summarization

The extractive text summarization technique involves pulling keyphrases from the source document and combining them to make a summary. The extraction is made according to the defined metric without making any changes to the texts.

Here is an example:

*Source text: Joseph and Mary rode on a donkey to attend the annual event in Jerusalem. In the city, Mary gave birth to a child named Jesus.*

*Extractive summary: Joseph and Mary attend event Jerusalem. Mary birth Jesus.*

- Abstraction-based summarization

When abstraction is applied for text summarization in deep learning problems, it can generate novel words and phrases not featured in the source text – as a human-written abstract usually does, it is not a mere selection of a few existing passages or sentences extracted from the source.

Here is the example:

*Abstractive summary: Joseph and Mary came to Jerusalem where Jesus was born.*

---

# Getting Start With the Research Paper

## Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond

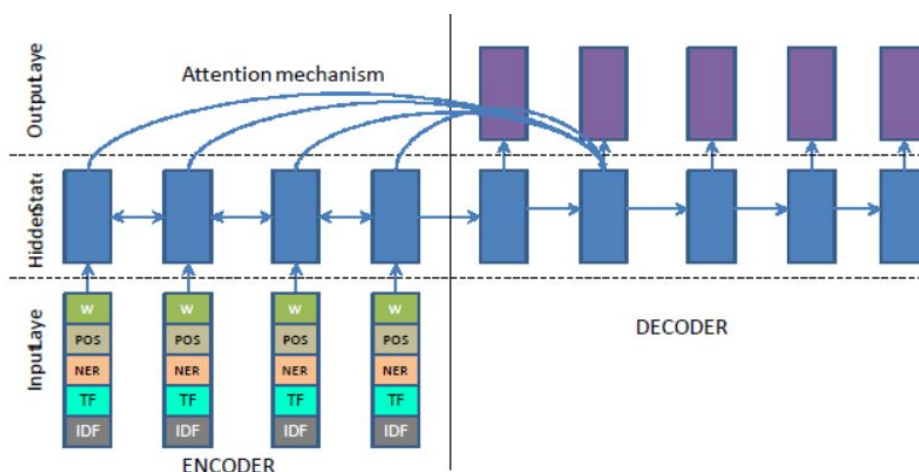
*Author: Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar G˘ulçehre, Bing Xiang (All from IBM Watson)*

This paper is an academic summary of what IBM Watson lab improved over the text summarization, they modeled abstractive text summarization using Attentional Encoder-Decoder Recurrent Neural Networks, and proposed several novel models that address critical problems in summarization, also conducted a new dataset consisting of multi-sentence summaries, and establish performance benchmarks for further research.

- Models
  1. Encoder-Decoder RNN with Attention and Large Vocabulary Trick

The aim of this technique is to reduce the size of the soft-max layer of the decoder which is the main computational bottleneck. In addition, this technique also speeds up convergence by focusing the modeling effort only on the words that are essential to a given example. This technique is particularly well suited to summarization since a large proportion of the words in the summary come from the source document in any case.

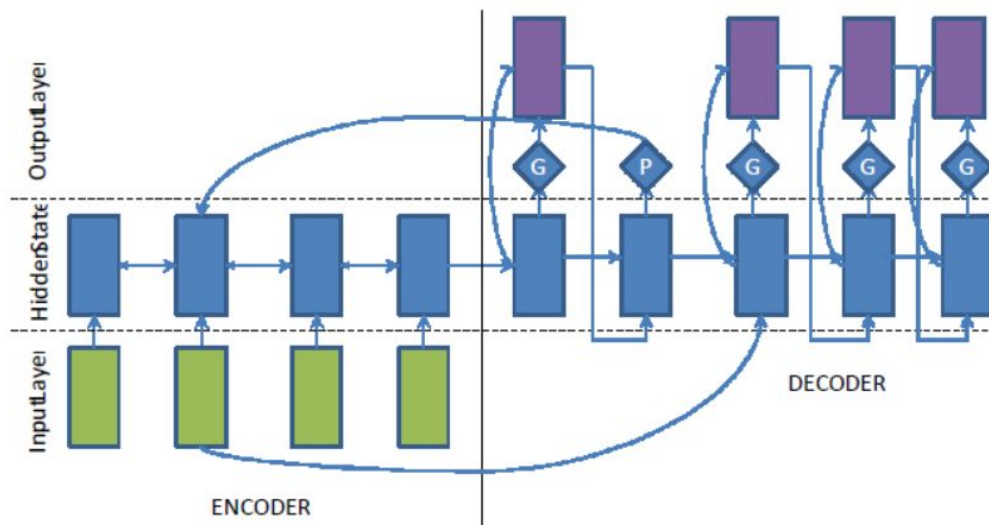
2. Capturing Keywords using Feature-rich Encoder



This technique uses one embedding vector each for POS, NER tags and discretized TF and IDF values, which are concatenated together with word-based embeddings as input to the encoder, in order to deal with one of the key challenges in summarization, which is to identify the key concepts and key entities in the document, around which the story revolves.

### 3. Modeling Rare/Unseen Words using Switching Generator-Pointer

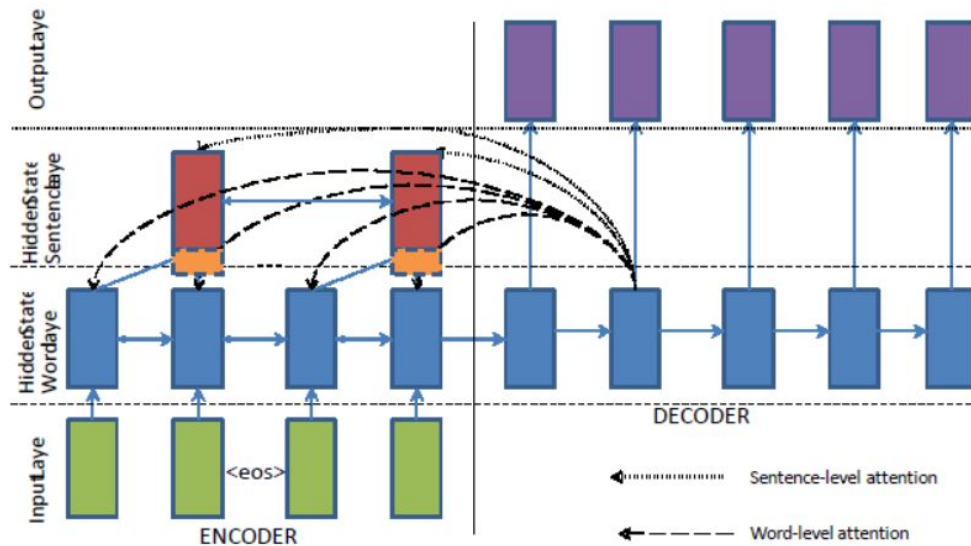
This technique uses a novel switching decoder/pointer architecture which is to simply point to their location in the source document, in order to handle those out-of-vocabulary (OOV) and rare/unseen-words issue in summarization.



As the graph above: When the switch shows 'G', the traditional generator consisting of the softmax layer is used to produce a word, and when it shows 'P' the pointer network is activated to copy the word from one of the source document positions. When the pointer is activated, the embedding from the source is used as input for the next time-step as shown by the arrow from the encoder to the decoder at the bottom.

### 4. Capturing Hierarchical Document Structure with Hierarchical Attention

In datasets where the source document is very long, in addition to identifying the keywords in the document, it is also important to identify the key sentences from which the summary can be drawn. This model uses hierarchical encoder with hierarchical attention to capture this notion.



As the graph above, the attention weights at the word level, represented by the dashed arrows are re-scaled by the corresponding sentence level attention weights, represented by the dotted arrows. The dashed boxes at the bottom of the top layer RNN represent sentence-level positional embeddings concatenated to the corresponding hidden states.

- Experiments and Results

They used the annotated Gigaword corpus, DUC corpus, and a new dataset which comprised multi-sentence summaries based on the original CNN/Daily Mail Corpus.

The results shows that their performance is marginally higher on DUC-2004 corpus compared with RAS-Elman's, but for Gigawords, since the sentences are longer, which could appear the model advantages, which can be seen as the graph below.

#	Model name	Rouge-1	Rouge-2	Rouge-L	Src. copy rate (%)
Full length F1 on our internal test set					
1	words-lvt2k-1sent	34.97	17.17	32.70	75.85
2	words-lvt2k-2sent	35.73	17.38	33.25	79.54
3	words-lvt2k-2sent-hieratt	36.05	18.17	33.52	78.52
4	feats-lvt2k-2sent	35.90	17.57	33.38	78.92
5	feats-lvt2k-2sent-ptr	<b>*36.40</b>	<b>17.77</b>	<b>*33.71</b>	78.70
Full length F1 on the test set used by (Rush et al., 2015)					
6	ABS+ (Rush et al., 2015)	29.78	11.89	26.97	91.50
7	words-lvt2k-1sent	32.67	15.59	30.64	74.57
8	RAS-Elman (Chopra et al., 2016)	33.78	15.97	31.15	
9	words-lvt5k-1sent	<b>*35.30</b>	<b>16.64</b>	<b>*32.62</b>	

('\*' indicates statistical significance of the corresponding model with respect to the baseline model on its dataset as given by the 95% confidence interval in the official Rouge script.)

Explicit modeling of important information such as multiple source sentences, word-level linguistic features, using the switch mechanism to point to source words when needed, and hierarchical attention, solve specific problems in summarization, each boosting performance incrementally.

- Key Conclusion

In this work, they applied the attentional encoder-decoder for the task of abstractive summarization with very promising results, outperforming state-of-the-art results significantly on two different datasets. Each of their proposed novel models addresses a specific problem in abstractive summarization, yielding further improvement in performance. They also proposed a new dataset for multi-sentence summarization and establish benchmark numbers on it.

As part of their future work, they planned to build more robust models for summaries consisting of multiple sentences.

# Get To The Point: Summarization with Pointer-Generator Networks

*Author: Abigail See(Stanford University), Peter J. Liu(Google Brain), Christopher D. Manning(Stanford University)*

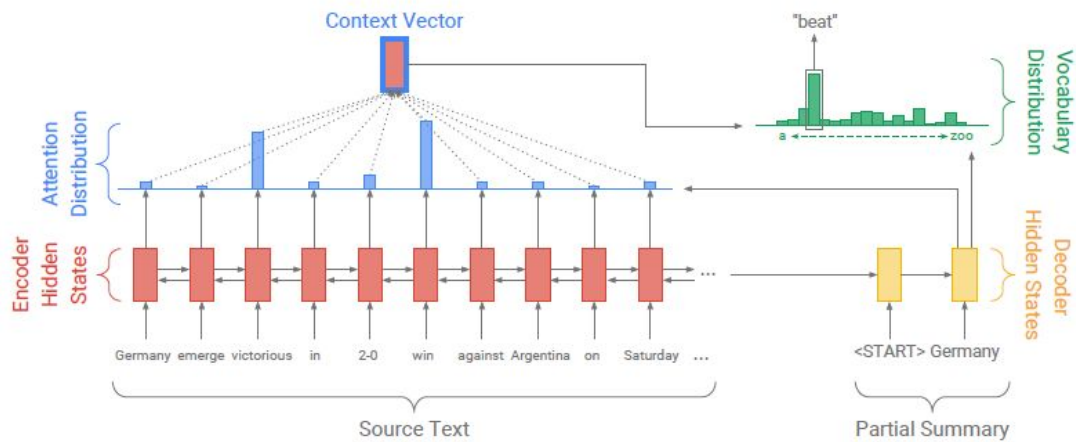
They pointed out that neural sequence-to-sequence models have provided a viable new approach for abstractive text summarization. However, these models have two shortcomings: they are liable to reproduce factual details inaccurately, and they tend to repeat themselves. So in this work what make their architecture novel are due to they combined “Pointing the unknown words” and “Modeling Coverage for Neural Machine Translation” together, thus augments the standard sequence-to-sequence attentional model in two orthogonal ways.

- Models

First, they use a hybrid pointer-generator network that can copy words from the source text via pointing, which aids accurate reproduction of information, while retaining the ability to produce novel words through the generator. Second, they used coverage to keep track of what has been summarized, which discourages repetition.

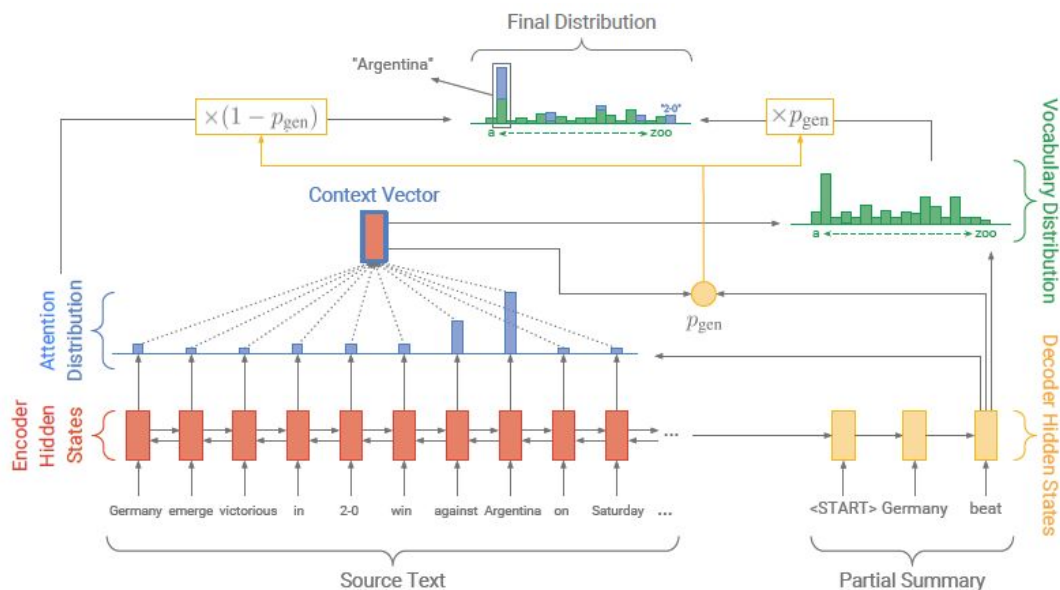
1. Sequence-to-sequence attentional model

This is the baseline model and is depicted in the figure below. The tokens of the article  $W_i$  are fed one-by-one into the encoder (a single-layer bidirectional LSTM), producing a sequence of encoder hidden states  $H_i$ . On each step  $t$ , the decoder (a single-layer unidirectional LSTM) receives the word embedding of the previous word and has decoder state  $S_t$ ; The attention distribution can be viewed as a probability distribution over the source words, that tells the decoder where to look to produce the next word. Next, the attention distribution is used to produce a weighted sum of the encoder hidden states, known as the context vector; Then the probability distribution over all words in the vocabulary will provides us with the final distribution from which to predict words  $w$ .



## 2. Pointer-generator network

Their pointer-generator network is a hybrid between the baseline and a pointer network, as it allows both copying words via pointing, and generating words from a fixed vocabulary. In addition, they used generation probability for timestep  $t$  which is calculated from the context vector, the decoder state and the decoder input. And The ability to produce OOV words is one of the primary advantages of pointer-generator models.



## 3. Coverage mechanism



Since repetition is a common problem for sequence-to-sequence models, and is especially pronounced when generating multi-sentence text, so they adapt the coverage model to solve the problem. The coverage vector is used as extra input to the attention mechanism, and it is necessary to additionally define a coverage loss to penalize repeatedly attending to the same locations.

- Experiments and Results

For the data and experiments, they used CNN/Daily Mail dataset and modeled with 256-dimensional hidden states and 128-dimensional word embeddings. They compared the ROUGE F1 and METEOR scores as evaluation, and the results can be seen in the following graph:

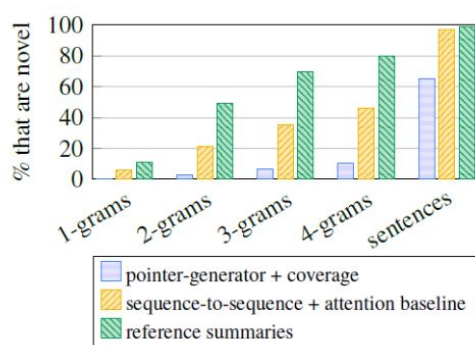


Figure 6: Although our **best model** is abstractive, it does not produce novel  $n$ -grams (i.e.,  $n$ -grams that don't appear in the source text) as often as the **reference summaries**. The **baseline model** produces more novel  $n$ -grams, but many of these are erroneous (see section 7.2).

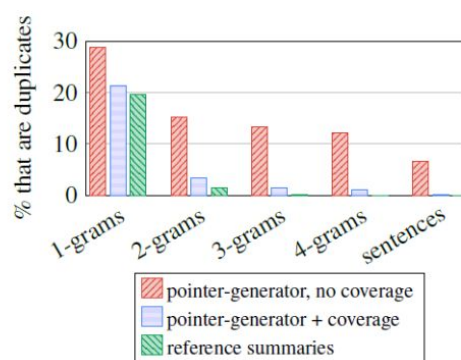


Figure 4: Coverage eliminates undesirable repetition. Summaries from our **non-coverage model** contain many duplicated  $n$ -grams while our **coverage model** produces a similar number as the **reference summaries**.

- Key Conclusion

The key conclusion in this work is that they presented a hybrid pointer-generator architecture with coverage, and showed that it reduces inaccuracies and repetition. They applied their model to a challenging longtext dataset, and significantly outperformed the abstractive state-of-the-art result. Their model exhibits many abstractive abilities, but attaining higher levels of abstraction remains an open research question.

# A Neural Attention Model for Abstractive Sentence Summarization

- *3<sup>rd</sup> Sept 2015*
- *Alexander M. Rush, Facebook AI Research / Harvard SEAS*
- *Sumit Chopra, Facebook AI Research*
- *Jason Weston, Facebook AI Research*

**Proposal:** A fully data-driven approach to abstractive sentence summarization.

**Method:** A local attention-based model that generates each word of the summary conditioned on the input sentence.

**About model:** Easily trained end-to-end and scales to a large amount of training data. Shows significant performance gains on the DUC-2004 shared task compared with several strong baselines.

**Results:** approach outperforms a machine translation system trained on the same large-scale dataset and yields a large improvement over the highest scoring system in the DUC-2004 competition.

**Aim** of text-summarization is to produce a condensed representation of an input text that captures the core meaning of the original.

**Extractive approach** - crop out and stitch together portions of the text to produce a condensed version.

**Abstractive summarization** - attempts to produce a bottom-up summary, aspects of which may not appear as part of the original.

**Work:**

The task of sentence-level summarization. Deletion-based sentence compression techniques. Apply various other operations while condensing, such as [paraphrasing](#), [generalization](#), and [reordering](#). Combine a neural language model with a contextual input encoder. encoder is modelled off of the attention-based encoder, in that it learns a latent soft alignment over the input text to help form the summary.

Both the encoder and the generation model are trained jointly on the sentence summarization task. incorporates a [beam-search decoder](#) as well as additional features to model extractive elements. Incorporates less linguistic structure than abstractive summarization.

The score of generating the next word in summary depends on the context (for...against) and input ( $x_1 \dots x_{18}$ ). Generalize (russian defense minister or russia) and paraphrase (for combating to against), compressing (dropping the creation of)

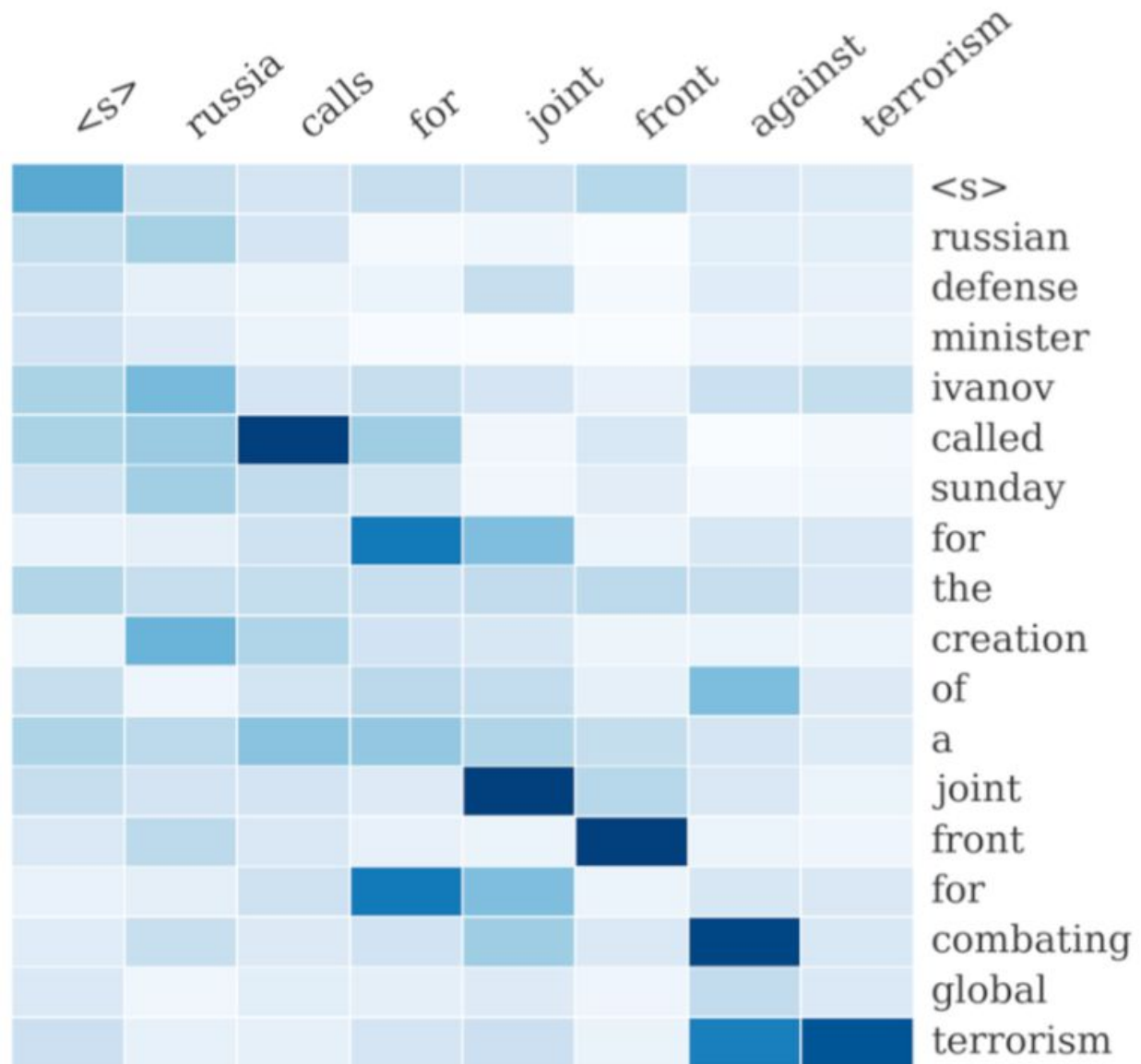
Makes no assumptions about the vocabulary of the generated summary it can be trained directly on any document-summary pair.

Input ( $\mathbf{x}_1, \dots, \mathbf{x}_{18}$ ). First sentence of article:

russian defense minister ivanov called sunday for the creation of a joint front for combating global terrorism

Output ( $\mathbf{y}_1, \dots, \mathbf{y}_8$ ). Generated headline:

*russia calls for joint front against **terrorism***  $\Leftarrow g(\text{terrorism}, \mathbf{x}, \text{for}, \text{joint}, \text{front}, \text{against})$



## 2. Encoders:

Incorporating in 'enc' and training the two elements jointly incorporated the input text into generation. Used Bag-Of-Words encoder, embedded to size H (hidden layer). It has the potential

to capture the relative importance of words to distinguish content words from stop words or embellishments and learn to combine words. It can allow interactions between words not requiring  $y$  context. Utilize an attention-based contextual encoder that constructs a representation based on the generation context. Where  $G$  is an embedding of the context,  $P$  is a new weight matrix parameter mapping between the context embedding and input embedding, and  $Q$  is a smoothing window. The soft alignment is then used to weight the smoothed version of the input  $\bar{x}$  when constructing the representation.

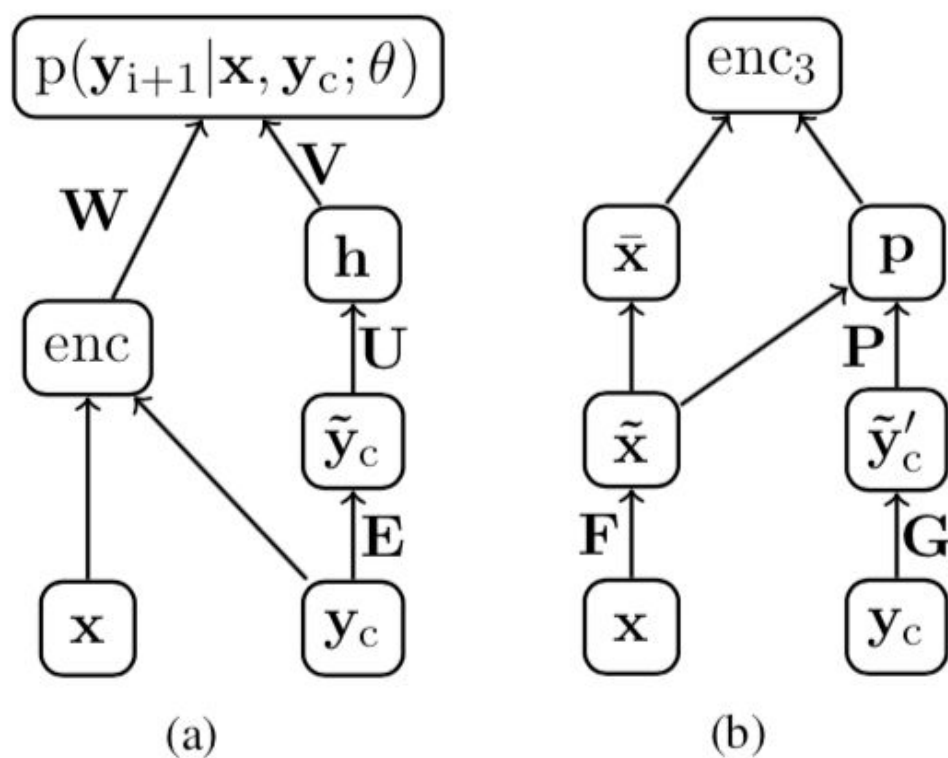


Figure 3: (a) A network diagram for the NNLM decoder with additional encoder element. (b) A network diagram for the attention-based encoder  $enc_3$ .

### 3. Training:

$$\begin{aligned}
\text{NLL}(\theta) &= - \sum_{j=1}^J \log p(\mathbf{y}^{(j)} | \mathbf{x}^{(j)}; \theta), \\
&= - \sum_{j=1}^J \sum_{i=1}^{N-1} \log p(\mathbf{y}_{i+1}^{(j)} | \mathbf{x}^{(j)}, \mathbf{y}_c; \theta).
\end{aligned}$$

NLL built using mini batch stochastic gradient descent

4. **Training** process: each round of beam search is dominated by computing  $p(\mathbf{y}_i | \mathbf{x}, \mathbf{y}_c)$  for each of the  $K$  hypotheses from left to right, computed as mini-batch, which reduces the factor of  $K$ .

5. **Issues**: the abstractive model does not have the capacity to find extractive word matches when necessary, for example transferring unseen proper noun phrases from the input.

**Solved with** – fine tuning small set of additional features that trade- off the abstractive/extractive tendency of the system, done by modifying our scoring function to directly estimate the probability of a summary using a log-linear model

Included indicators of uni- gram, bigram, and trigram match with the input as well as reordering of input words.

$$p(\mathbf{y} | \mathbf{x}; \theta, \alpha) \propto \exp(\alpha^\top \sum_{i=0}^{N-1} f(\mathbf{y}_{i+1}, \mathbf{x}, \mathbf{y}_c)).$$

Evaluation: Used minimum- error rate training (MERT) to tune for the summarization metric on tuning data, ROUGE metric, recall evaluation,

**Application**: News headline generation.

**Dataset:** DUC-2003 and DUC- 2004 shared tasks, 500 news articles from the New York Times and Associated Press Wire services each paired with 4 different human-generated reference summaries

**Trained on:** pair the headline of each article with its first sentence to create an input- summary pair.

**Results:**

Model	ROUGE-1	DUC-2004		ROUGE-1	Gigaword		Ext. %
		ROUGE-2	ROUGE-L		ROUGE-2	ROUGE-L	
IR	11.06	1.67	9.67	16.91	5.55	15.58	29.2
PREFIX	22.43	6.49	19.65	23.14	8.25	21.73	100
COMPRESS	19.77	4.02	17.30	19.63	5.13	18.28	100
W&L	22	6	17	-	-	-	-
TOPIARY	25.12	6.46	20.12	-	-	-	-
MOSES+	26.50	8.13	22.85	28.77	12.10	26.44	70.5
ABS	26.55	7.06	22.05	30.88	12.22	27.77	85.4
ABS+	28.18	8.49	23.81	31.00	12.65	28.34	91.5
REFERENCE	29.21	8.38	24.46	-	-	-	45.6

### Rouge metric?

Recall-Oriented Understudy for Gisting Evaluation metrics compare an automatically produced summary or translation against a reference or a set of references (human-produced) summary or translation.

**Framework:** Torch

### Mathematics of attention based model:

1. Input  $x_1$  to  $x_M$  words
2. Vocabulary size =  $V$ ,  $x_1..x_M$  belong to  $V$
3. Each word as an indicator vector  $x_i$  belongs to  $\{0,1\}$  of size  $V$  and  $I$  belongs to  $1..M$
4. Each sentence is a sequence of indicators
5.  $X$  is a set of possible input sentences
6. Output  $y$  of length  $N < M$
7. Words in summary belong to  $V$
8. Output sequence is  $y_1..y_N$
9. Given that  $N$  is already known by the generator

10.  $\mathcal{Y}$  belongs to  $\{0,1\}^V$  of size  $V$ , all possible  $N$  sentences of length  $N$  (possible summaries),  $y_i$  is indicator
11. Abstractive summary is optimal sequence from set of  $\mathcal{Y}$
12.  $S$  is scoring function: of size input vector into output vector

$$\arg \max_{\mathbf{y} \in \mathcal{Y}} s(\mathbf{x}, \mathbf{y}), \quad (1)$$

13. Extractive summary transfers words from input

$$\arg \max_{m \in \{1, \dots, M\}^N} s(\mathbf{x}, \mathbf{x}_{[m_1, \dots, m_N]}), \quad (2)$$

14. Compression is deleting words from input for summary

$$\arg \max_{m \in \{1, \dots, M\}^N, m_{i-1} < m_i} s(\mathbf{x}, \mathbf{x}_{[m_1, \dots, m_N]}). \quad (3)$$

15. Freedom = less constrains of:
  - a. Same words from input
  - b. Just deleting from input
  - c. Fits with wider range of training data
16. Factored scoring function: Take into account fixed window of previous words

$$s(\mathbf{x}, \mathbf{y}) \approx \sum_{i=0}^{N-1} g(\mathbf{y}_{i+1}, \mathbf{x}, \mathbf{y}_c), \quad (4)$$

17. Conditional log probability  
 Markov assumption: Length of context is  $C$   
 If  $i < 1$  then  $y_i$  is a special symbol  $\{S\}$



$$\log p(\mathbf{y}|\mathbf{x}; \theta) \approx \sum_{i=0}^{N-1} \log p(\mathbf{y}_{i+1}|\mathbf{x}, \mathbf{y}_c; \theta),$$

$$p(\mathbf{y}_{i+1}|\mathbf{x}, \mathbf{y}_c; \theta).$$

18. Determine local conditional distribution

19. Previous model = a conditional summarization model

$$\arg \max_{\mathbf{y}} \log p(\mathbf{y}|\mathbf{x}) = \arg \max_{\mathbf{y}} \log p(\mathbf{y})p(\mathbf{x}|\mathbf{y})$$

20. Original distribution of neural network

- a. Neural probability language model
- b. Encoder = conditional prob model

21. In language model, the contextual probability of next word is obtained and used for next word in summary

### **Conclusion:**

neural attention-based model for abstractive summarization, based on recent developments in neural machine translation. combine this probabilistic model with a generation algorithm which produces accurate abstractive summaries












**Next Steps:** Improve the grammaticality of the summaries in a data-driven way, as well as scale this system to generate paragraph-level summaries. Both pose additional challenges in terms of efficient alignment and consistency in generation.

### **Word embedding matrix:**

Word Embeddings are the texts converted into numbers and there may be different numerical representations of the same text. A vector representation of a word may be a one-hot encoded vector where 1 stands for the position where the word exists and 0 everywhere else

## Preprocess the CNN Dataset

- Download and unzip the stories directories for CNN Dataset

<input type="checkbox"/> Name	Date modified	Type
 000cd1ee0098c4d510a03ddc97d11764448ebac2	7/31/2017 1:53 AM	STORY File
 00a2aef1e18d125960da51e167a3d22ed8416c09	7/31/2017 1:53 AM	STORY File
 00a6c0823ba92e5a055d90133e8b66fa9028c2d7	7/31/2017 1:55 AM	STORY File
 00a51d5454f2ef7dbf4c53471223a27fb9c20681	7/31/2017 1:53 AM	STORY File
 00a57bef588a7e7efd0438fba62f07356c59b036	7/31/2017 1:54 AM	STORY File
 00a79ca59be700e845bf9e90bf08c5106b4a72bb	7/31/2017 1:53 AM	STORY File
 00aa8e3ba59f5fe71a096a5549f80faece10aa83	7/31/2017 1:55 AM	STORY File
 00ac6fa0167e99167565ab9a12c1c96bf48ac414	7/31/2017 1:54 AM	STORY File
 00ac91a5794537f1f37a8962a41fea6dbea2779b	7/31/2017 1:53 AM	STORY File
 00ae31ff5ede1de6cd39eccf36bae935c882527b	7/31/2017 1:57 AM	STORY File
 00af3b469010a46f154734cfba77bb001701517f	7/31/2017 1:52 AM	STORY File

- Download Stanford CoreNLP to tokenize the data

Stanford CoreNLP provides a set of human language technology tools. It can give the base forms of words, their parts of speech, whether they are names of companies, people, etc., normalize dates, times, and numeric quantities, mark up the structure of sentences in terms of phrases and syntactic dependencies, indicate which noun phrases refer to the same entities, indicate sentiment, extract particular or open-class relations between entity mentions, get the quotes people said, etc.

### Named Entity Recognition:

	Person	Loc	ORDINAL	Location	
1	President Xi Jinping of China	on his	first	state visit to the United States	showed off his familiarity with
	Misc		Date	Time	
	American history and pop culture	on Tuesday	night		

### Coreference:

	Mention	Coref	M
1	President Xi Jinping of China, on his first state visit to the United States		his
	showed off		familiarity with American history and pop culture on Tuesday night

- Tokenized the stories

```
def tokenize_stories(stories_dir, tokenized_stories_dir):
    """Maps a whole directory of .story files to a tokenized version using Stanford CoreNLP Tokenizer"""
    print("Preparing to tokenize %s to %s..." % (stories_dir, tokenized_stories_dir))

    stories = os.listdir(stories_dir)
    # make IO list file
    print("Making list of files to tokenize...")

    with open("mapping.txt", "w") as f:
        for s in stories:
            f.write("%s \t %s\n" % (os.path.join(stories_dir, s), os.path.join(tokenized_stories_dir, s)))
    command = ['java', 'edu.stanford.nlp.process.PTBTOKENIZER', '-ioFileList', '-preserveLines', 'mapping.txt']
    print("Tokenizing %i files in %s and saving in %s..." % (len(stories), stories_dir, tokenized_stories_dir))

    subprocess.call(command)
    print("Stanford CoreNLP Tokenizer has finished.")

    os.remove("mapping.txt").

    # Check that the tokenized stories directory contains the same number of files as the original directory
    #num_orig = len(os.listdir(stories_dir))
    #num_tokenized = len(os.listdir(tokenized_stories_dir))

    if num_orig != num_tokenized:
        raise Exception("The tokenized stories directory %s contains %i files, but it should contain the same number")
    print("Successfully finished tokenizing %s to %s.\n" % (stories_dir, tokenized_stories_dir))
```

- Create article.txt and abstract.txt files for chatbot building

```

def get_art_abs(story_file):
    lines = read_text_file(story_file)

    # Lowercase everything
    lines = [line.lower() for line in lines]

    '''
    Put periods on the ends of lines that are missing them
    this is a problem in the dataset because many image captions don't end in periods
    consequently they end up in the body of the article as run-on sentences
    '''
    lines = [fix_missing_period(line) for line in lines]

    # Separate out article and abstract sentences
    article_lines = []
    highlights = []
    next_is_highlight = False

    for idx,line in enumerate(lines):
        if line == "":
            continue # empty line

        elif line.startswith("@highlight"):
            next_is_highlight = True

        elif next_is_highlight:
            highlights.append(line)

        else:
            article_lines.append(line)

    # Make article into a single string
    article = ' '.join(article_lines)

    # Make abstract into a single string, putting <s> and </s> tags around the sentences
    abstract = ' '.join(highlights)

    return article, abstract

```

```

def create_corpus(cnn_tokenized_stories_dir, cnn_corpus_dir):
    cnn_stories = os.listdir(cnn_tokenized_stories_dir)

    cnn_content_file = os.path.join(cnn_corpus_dir, 'articles.txt')
    cnn_summary_file = os.path.join(cnn_corpus_dir, 'abstracts.txt')

    with open(cnn_content_file, 'w', encoding='utf-8') as content, open(cnn_summary_file, 'w', encoding='utf-8') as summary:
        for s in cnn_stories:
            s_file = os.path.join(os.path.join(os.getcwd(), cnn_tokenized_stories_dir), s)
            article, abstract = get_art_abs(s_file)
            if len(article.split()) != 0:
                content.write(article.rstrip('\n') + '\n')
                summary.write(abstract.rstrip('\n') + '\n')

```

```

cnn_corpus_dir = "cnn_stories_tokenized"

```

```

create_corpus(cnn_tokenized_stories_dir, cnn_corpus_dir)

```

## Building the Chatbot