CS772: Deep Learning for Natural Language Processing (DL-NLP)

Summarization, Opinion Summarization, DNN

Pushpak Bhattacharyya
Computer Science and Engineering Department
IIT Bombay

Week 13 of 3rd April, 2023

Re-cap

Gricean Maxims: Cooperative Principle in Converstaion (Wikipedia)

- **Quantity, Quality, Relation, and Manner**
- Paul Grice, philosopher of language
- "*Make your contribution such as is required, at the stage at which it occurs, by the accepted purpose or direction of the talk exchange in which you are engaged*".
- Captures the LINK between utterances

AI chatbots compared: Bard vs. Bing vs. ChatGPT

https://www.theverge.com/2023/3/24/23653377/ai-chatbots-comparison-bard-bing-chatgpt-gpt-4

Comparison: Chatbots

Google’s Bard (https://bard.google.com/),

Microsoft’s Bing (https://www.theverge.com/2023/3/24/23653377/ai-chatbots-comparison-bard-bing-chatgpt-gpt-4),

OpenAI’s ChatGPT (https://chat.openai.com/chat#)

3 stages of LLM based CAI

- Generative Pretraining (GP)
- Supervised Fine Tuning (SFT)
- Reinforcement Learning from Human Feedback (RLHF)

Enter Pragmatics

Modeling

***P(e)*: "language" model**

$$e^* = \arg\max_e P(e \mid f)$$

$$= \arg\max_e [P(e)P(f \mid e)]$$

- Dialogue Act Classification (DAC): *f→ Dialogue Sequence, e→Dialogue turn labels*
- Dialogue Intent: *f→ dialogue sequence, e→ dialogue turns with Intent like 'question', 'elaboration', 'affirmation', 'command/request' etc.*

Elements of Pragmatics (1/2)

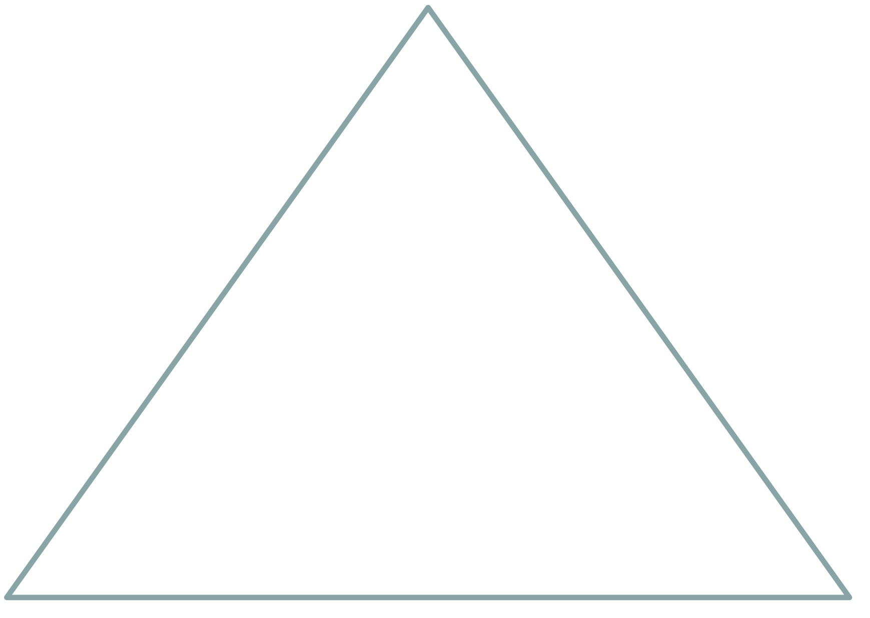
- Deixis (literally, ‘pointing with words’: temporal- *now, then;* spatial- *here, there*; personal- *I, you, he, they*; definite-indefinite- *this, that, those)*

- Presupposition: (*untie the shoe* → presupposes *the shoe was tied before*)

Elements of Pragmatics (2/2)

- Speech Acts: (*I pronounce you man and wife)*- **locutionary, illocutionary, and perlocutionary**
- Implicatures: (*A: shall we go for a walk? B: It is raining outside*)
- Politeness: (*close the door → please close the door → can you close the door → would you mind closing the door)*
- Information Structure: ordering of information (??*The table is under the flower pot*- odd: smaller object first mention) credit: Handke

The Trinity of Pragmatics



Diexis

Credit:
https://doi.org/10.1093/acrefore/9780199384655.013.213

Speech Act

Kinds of Speech Act

- Locutionary
- Illocutionary
- Perlocutionary
- Performative Speech acts

Implicatures

Computational Perspective: Conversational AI

Dialogue Based Computation

Zihao He, Leili Tavabi, Kristina Lerman, and Mohammad Soleymani. 2021. Speaker Turn Modeling for Dialogue Act Classification. In Findings of the Association for Computational Linguistics: EMNLP 2021, pages 2150–2157, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tulika Saha, Aditya Patra, Sriparna Saha and Pushpak Bhattacharyya, *Towards Emotion-aided Multi-modal Dialogue Act Classification*, Association of Computational Linguistics Conference (**ACL 2020**), Seattle USA, 5-10 July, 2020.

Summarization

SUMMARIZATION

- Task of automatically creating a compressed version of the text document (set of tweets, web-page, single/multi-document) that should be relevant, non-redundant and representative of the main idea of the text.
- A text that is produced from one or more texts that conveys important information in the original text(s), and that is no longer than half of the original text(s) and usually significantly less than that.
- Metric:

$$\textit{Compression Ratio} = \#word_{summary} / \#word_{document}$$

NLP Layer

Pragmatics

Semantics

Syntax

Lexical Level

Summarization Categorization

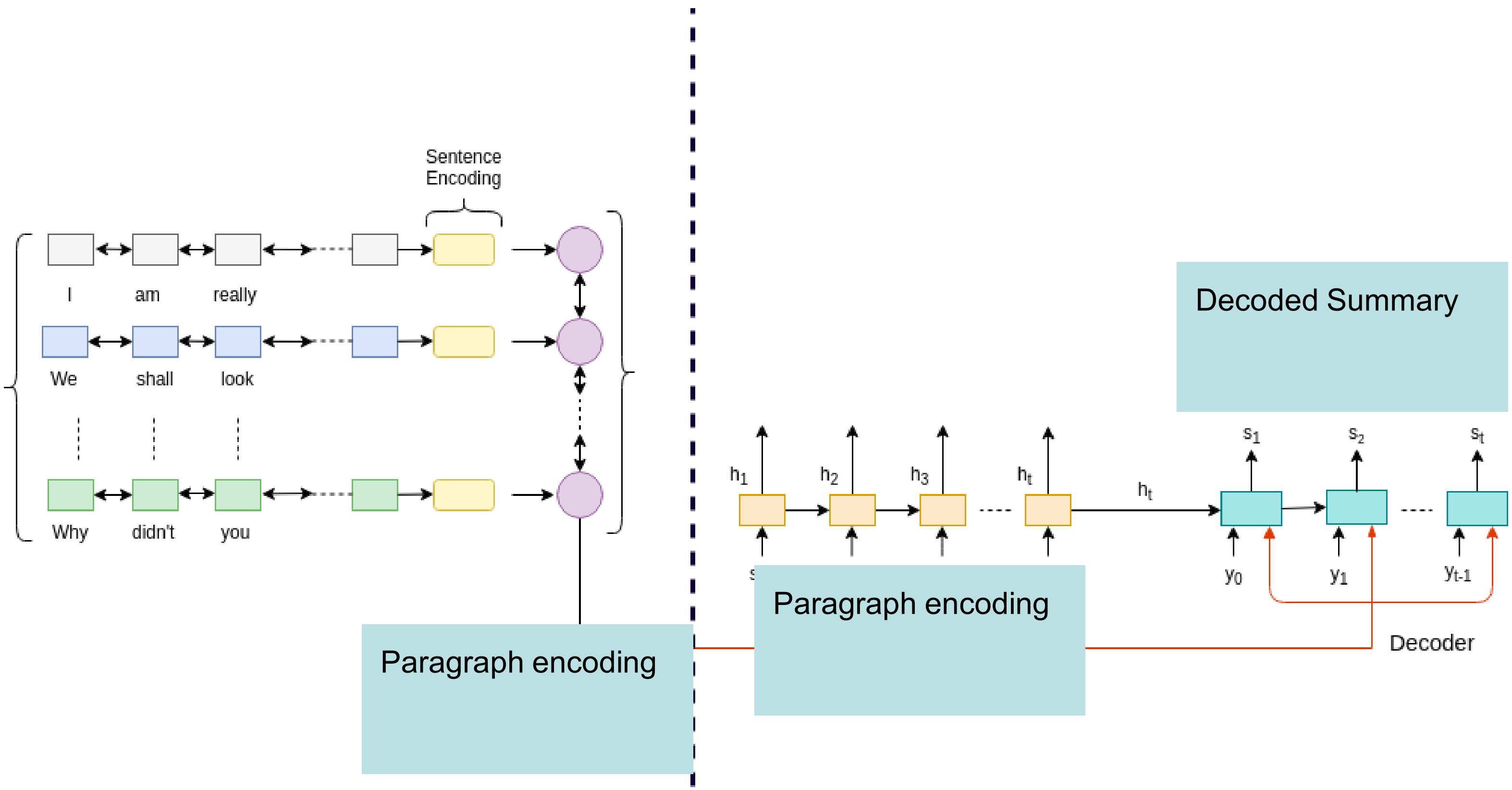
- Broad Categorization
  - Extractive: sentences from the input text form part of summary
  - Abstractive: Essence+Natural Language Generation
- Other categorizations:
  - # Document: Single and Multi document
  - Purpose: Generic and Query focused
  - Miscellaneous: Personalized, Sentiment-based, Update, E-mail-based, web-based

Handling Morphology in Abstractive Summarization

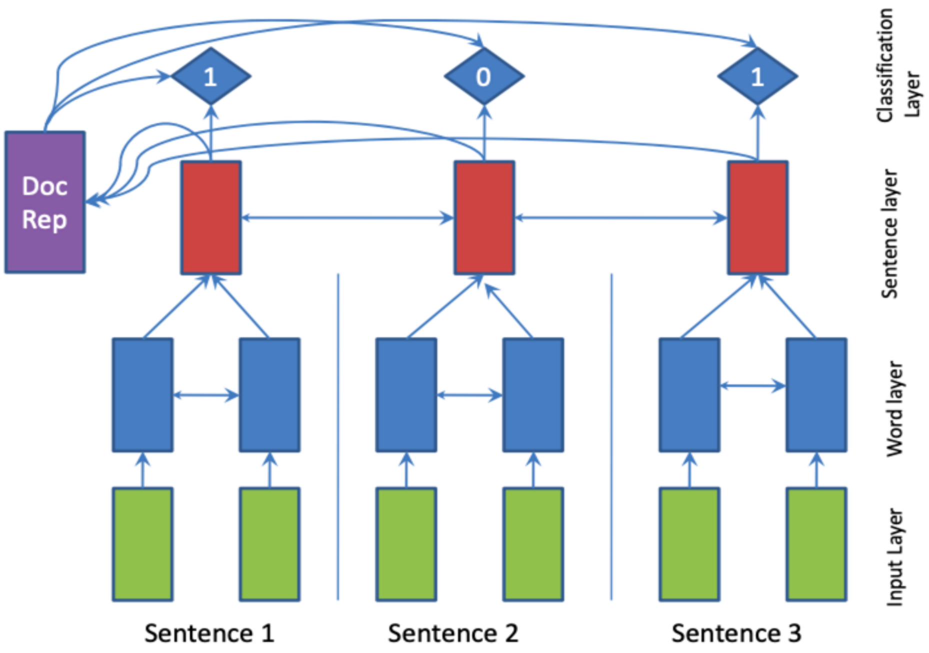
- Fasttext tried solving the morphology generation problem by BPE (byte pair encoding)
- Given “going”, divide the string into “go” and “ing”
- Use these parts to generate say “walking”
- Each subword will have its own probability
- If not subwording, then no way other than showing all forms of the root word: *go, went, going, gone*
- Languages differ in morphological complexity
- French more complex than English

Computation of Summaries

Hierarchical Encoder-Decoder



SummaRuNNer



[1]

Figure 1: SummaRuNNer: A two-layer RNN based sequence classifier

Summarization with Pointer-Generator Network

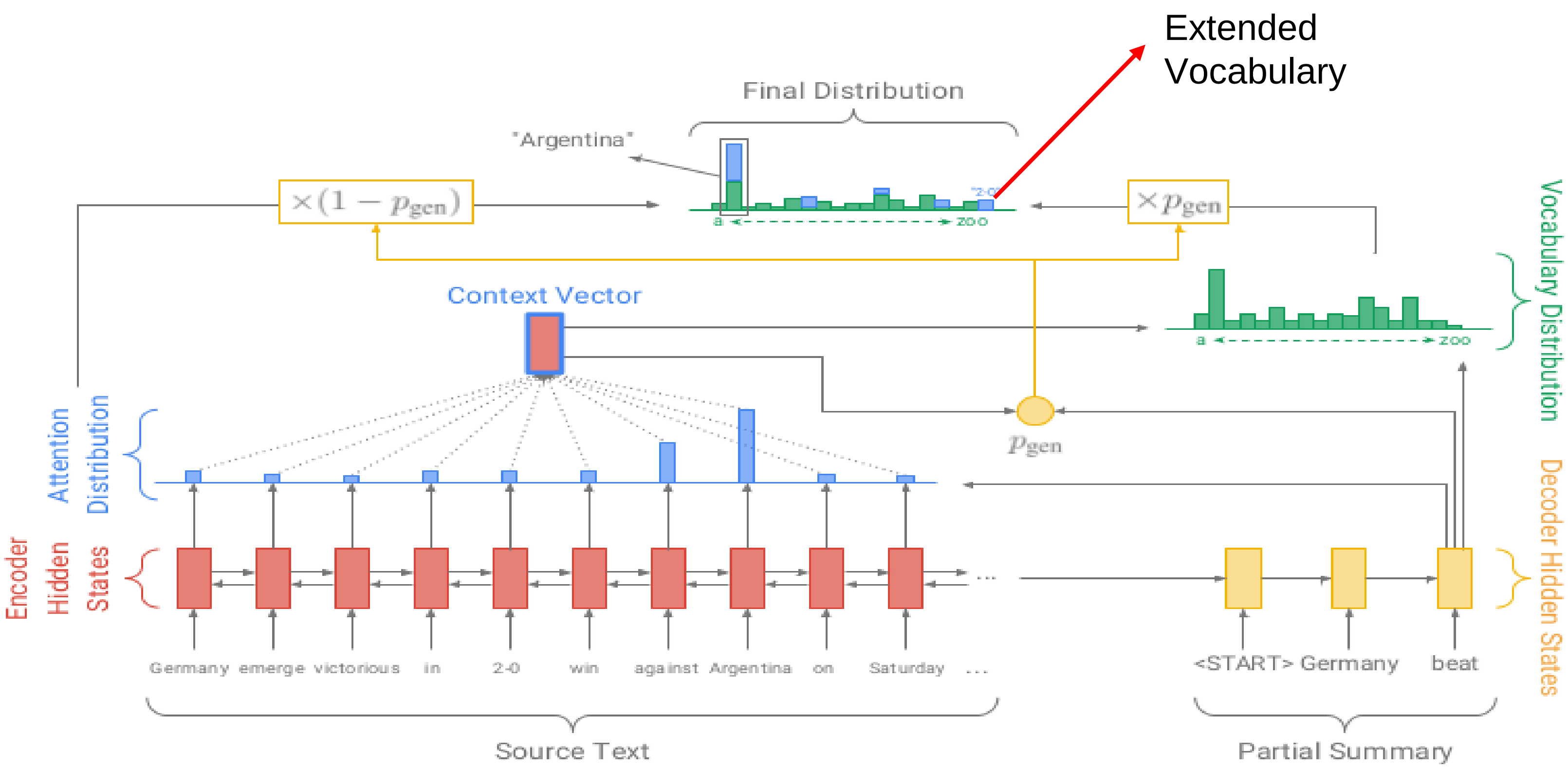


Figure: Pointer-generator Model [2]

BART

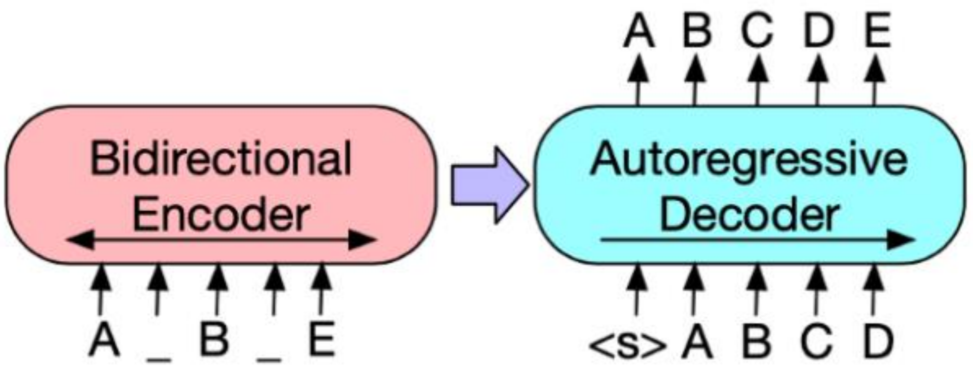


Fig 1: BART architecture.

- BERT (12 layers) + GPT (12 layers)
- Pre-trained on 160GB of news, books and web text
- Fine-tuned on CNN/DM dataset

Lewis, Mike, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension." *arXiv preprint arXiv:1910.13461* (2019).

Now GPT…

1. Generative Pre-training
2. Supervised Fine Tuning
3. Reinforcement Learning with Human Feedback (RLHF)

Opinion/Review Summaries

Properties of Opinion Summaries

- **Monotonicity**: As more sentences are added to opinion summary, subjectivity increases along with information content
- **Diminishing Return:** If multiple sentences of varying intensity are added to opinion summary, the effect of lower intensity diminishes in presence of higher intensity bearing polar sentences

Examples from cricket: diminishing return

A: *Rahul Dravid is a great batsman*

B: *Rahul Dravid is a very consistent player*

$A \cup B$:

Rahul Dravid is a great batsman. He is a very consistent player

Compare *B* and $A \cup B$; "effect" of *B* diminished in presence of *A*

When asked to summarize $A \cup B$ *in one sentence*, *B is likely to be dropped*

Example from cricket: coverage

A: *"Sachin is a great batsman"*
B: "His backfoot batting is unmatchable"
C: "He also bowls decent spin"

- If the budget allows only two subjective sentences, then picking up *A* and *B* have captured only batting
- Picking up *C* with the one of *A* and *B* would have covered both aspects (i.e. batting and bowling)
- Sentences are not overlapping in aspects, hence no diminishing return
- Higher intensity dominates

Submodular Function (1/2)

Finite set V

Set Function $F: 2^V \rightarrow R$, $F(\varphi) = 0$

Definition: $F : 2^V \rightarrow R$ is submodular iff

$$\forall A, B \subset V, F(A) + F(B) \geq F(A \cap B) + F(A \cup B)$$

Submodular Function (2/2)

Equivalent definition:

$\forall k \in V, \; \forall A \subset V,$

$F(A \cup \{k\}) - F(A)$ is non-increasing

(diminishing return)

$\Leftrightarrow$

$\forall A \subset B, \; \forall k \notin A,$

$$F(A \cup \{k\}) - F(A) \geq F(B \cup \{k\}) - F(B)$$

Example of Submodular Functions: *Cut Functions, Set Cover*

Extractive Summarization and Submodularity

- Find a set $S \subseteq V$

- *S* is set of sentences in summary, *V* is set of sentences in Document

- which maximizes a submodular function *f(S)* subject to budget constraints.

Monotone Submodular Objective

$$F(S) = L(S) + \lambda R(S)$$

$F(S)$ -> Total Utility of summary

$L(S)$ -> Relevance

$R(S)$ -> Diversity

RELEVANCE

- Two properties of a good summary: *relevance* and *non-redundancy*
- *L(S)* measures the coverage, or "fidelity", of summary set *S* to the document, *R(S)* rewards diversity in *S*, *λ* is a trade-off coefficient

$$L(S) = \Sigma_{i \in V} min\{Ci(S), \alpha Ci(V)\}$$

$$C_i(S) = \Sigma_{j \in S}\, w(i,j)$$

w(i,j)>0 measures the similarity between *i* and *j* and thus *C* measures the similarity of summary with the document

NON-REDUNDANCY

- $R(S) = \sum_{i=1}^{K} \sqrt{\sum_{j \in P_i \cap S} r_j}$
- R(S) rewards diversity
- As soon as an element is selected from a cluster, other elements from the same cluster start having diminishing return
- r_j is the average similarity of sentence *i* with the rest of the document

$$r_i = \frac{1}{N} \sum_j w(i,j)$$

w(i,j): similarity between sentences *i* and *j*

Pointer Generator Network

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks, ACL.

Abstract (1/2)

- Proposes a novel architecture that augments the standard sequence-to-sequence attentional model in two orthogonal ways.
- First: uses a hybrid pointer-generator network that can copy words from the source text via *pointing*, which aids accurate reproduction of information, while retaining theability to produce novel words through the *generator*

Abstract (2/2)

- Second: uses *coverage* to keep track of what has been summarized, which discourages repetition
- Applies the model to the *CNN/Daily Mail* summarization task, outperforming the current abstractive state-of-the-art by at least 2 ROUGE points

Basic seq2seq n/w

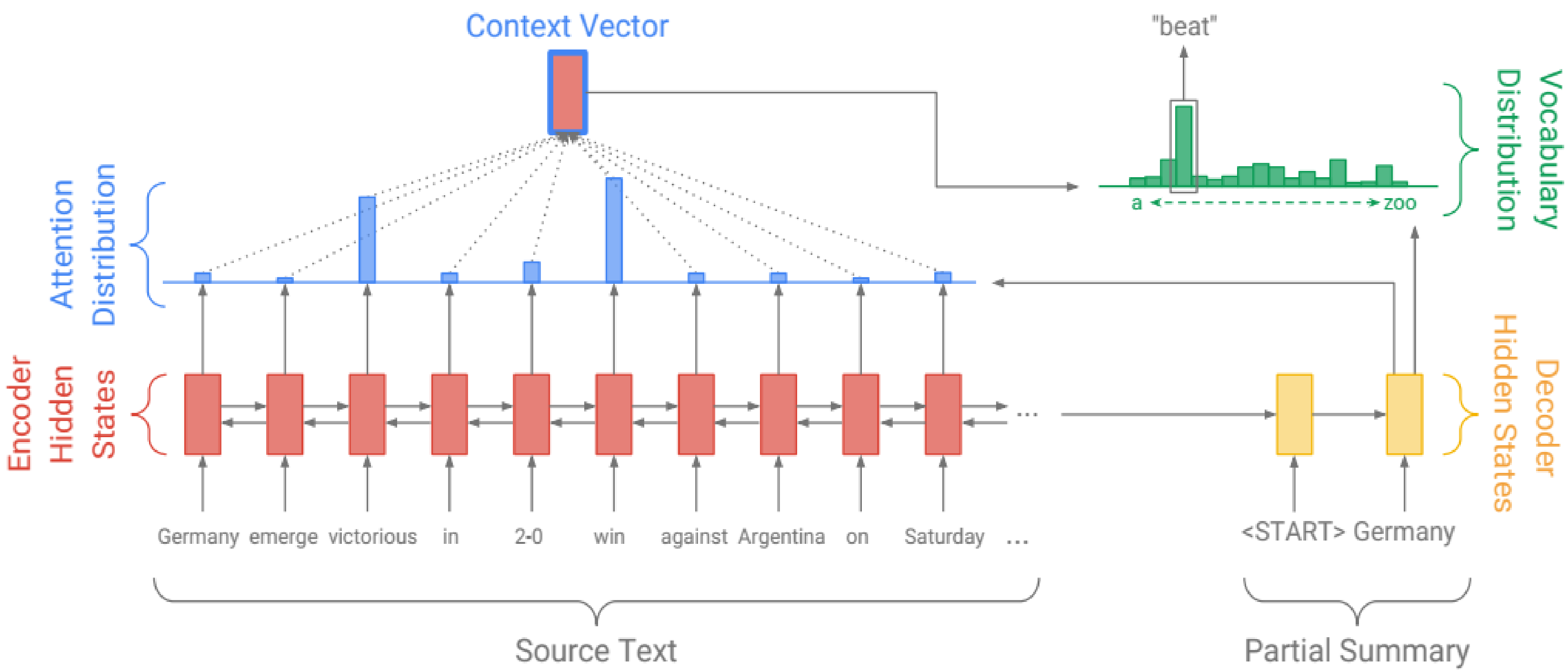


Figure 2: Baseline sequence-to-sequence model with attention. The model may attend to relevant words in the source text to generate novel words, e.g., to produce the novel word *beat* in the abstractive summary *Germany* ***beat*** *Argentina 2-0* the model may attend to the words *victorious* and *win* in the source text.

Pointer Generator N/W: copy word *vs.* new word

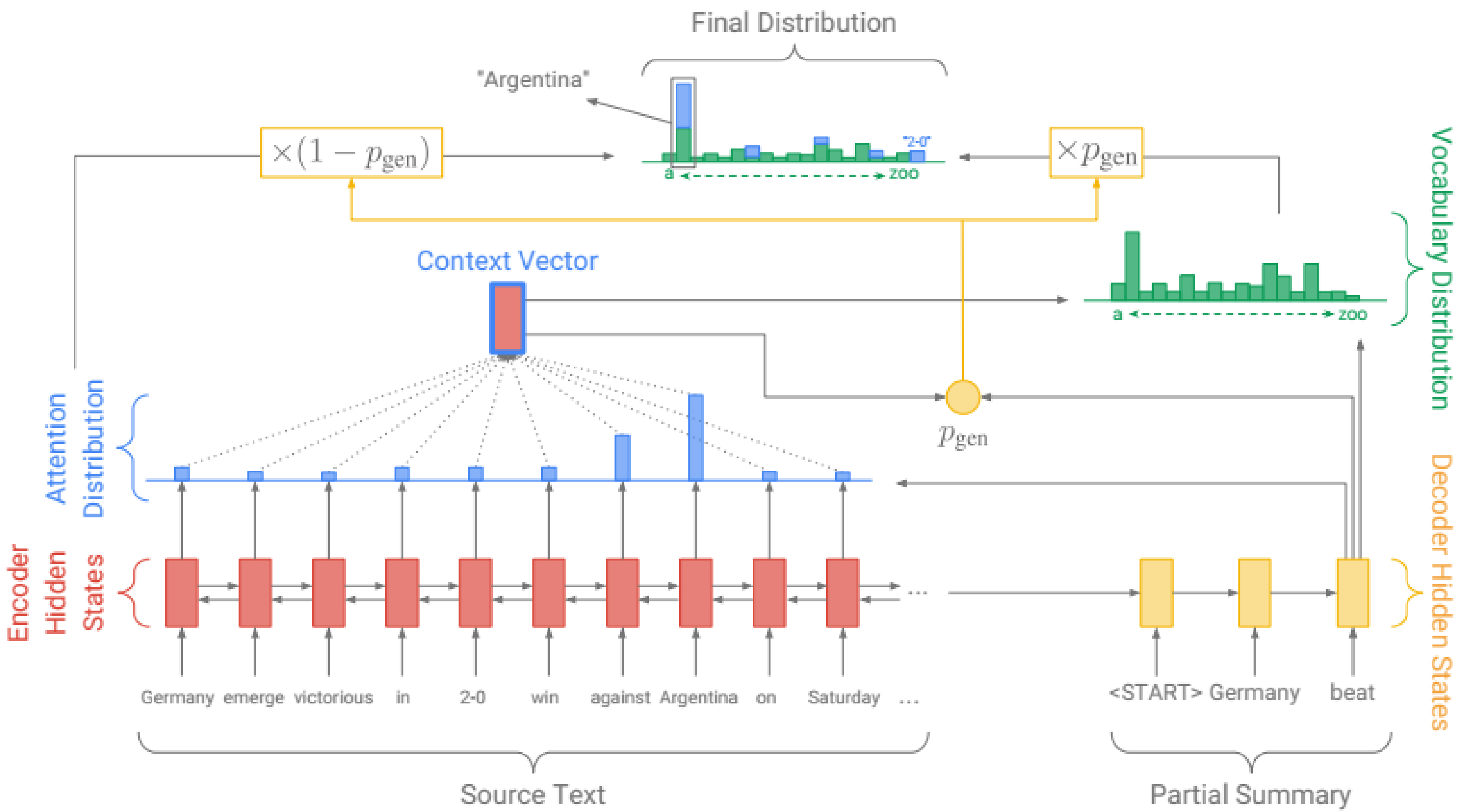


Figure 3: Pointer-generator model. For each decoder timestep a generation probability $p_{\text{gen}} \in [0,1]$ is calculated, which weights the probability of *generating* words from the vocabulary, versus *copying* words from the source text. The vocabulary distribution and the attention distribution are weighted and summed to obtain the final distribution, from which we make our prediction. Note that out-of-vocabulary article

Modeling: input processing

- Tokens w_i fed one-by-one into the encoder (a single-layer bidirectional LSTM), producing a sequence of *encoder hidden states* h_i

- At each step t, the decoder (a single-layer unidirectional LSTM) receives the word embedding of the previous word

Modeling: encoder hidden states

- While training, this is the previous word of the reference summary;
- at test time it is the previous word emitted by the decoder), and has *decoder state* s_t.

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + b_{\text{attn}}) \quad (1)$$

$$a^t = \text{softmax}(e^t) \quad (2)$$

where v, W_h, W_s and b_{attn} are learnable parameters.

Modeling: encoder hidden states

- Attention is 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* h_t^*

$$h_t^* = \sum_i a_i^t h_i \tag{3}$$

Modeling: vocab distribution

- The context vector is a fixed size representation of what has been read from the source for this step
- It is concatenated with the decoder state s_t and fed through two linear layers to produce the vocabulary distribution P_{vocab}

$$P_{\text{vocab}} = \text{softmax}(V'(V[s_t, h_t^*] + b) + b') \qquad (4)$$

where V , V_0, b and b_0 are learnable parameters.

Modeling: probability distribution over vocab

- *Pvocab* is a probability distribution over all words in the vocabulary, and provides the final distribution from which to predict words *w*.

$$P(w) = P_{\text{vocab}}(w) \quad (5)$$

Modeling: Loss

- During training, the loss for timestep *t* is the negative log likelihood of the target word w_t^* for that time step
- Overall loss for the whole sequence is:

$$\text{loss}_t = -\log P(w_t^*) \tag{6}$$

$$\text{loss} = \frac{1}{T}\sum_{t=0}^{T} \text{loss}_t \tag{7}$$

Modeling: Pointer Generator

- Allows both copying words via pointing, and generating words from a fixed vocabulary.
- *Generation probability* $p_{gen} \in [0,1]$ for timestep t is calculated from the context vector h_t^*, the decoder state s_t and the decoder input x_t

$$p_{\text{gen}} = \sigma(w_{h^*}^T h_t^* + w_s^T s_t + w_x^T x_t + b_{\text{ptr}}) \qquad (8)$$

Pointer Generator N/W: copy word *vs.* new word

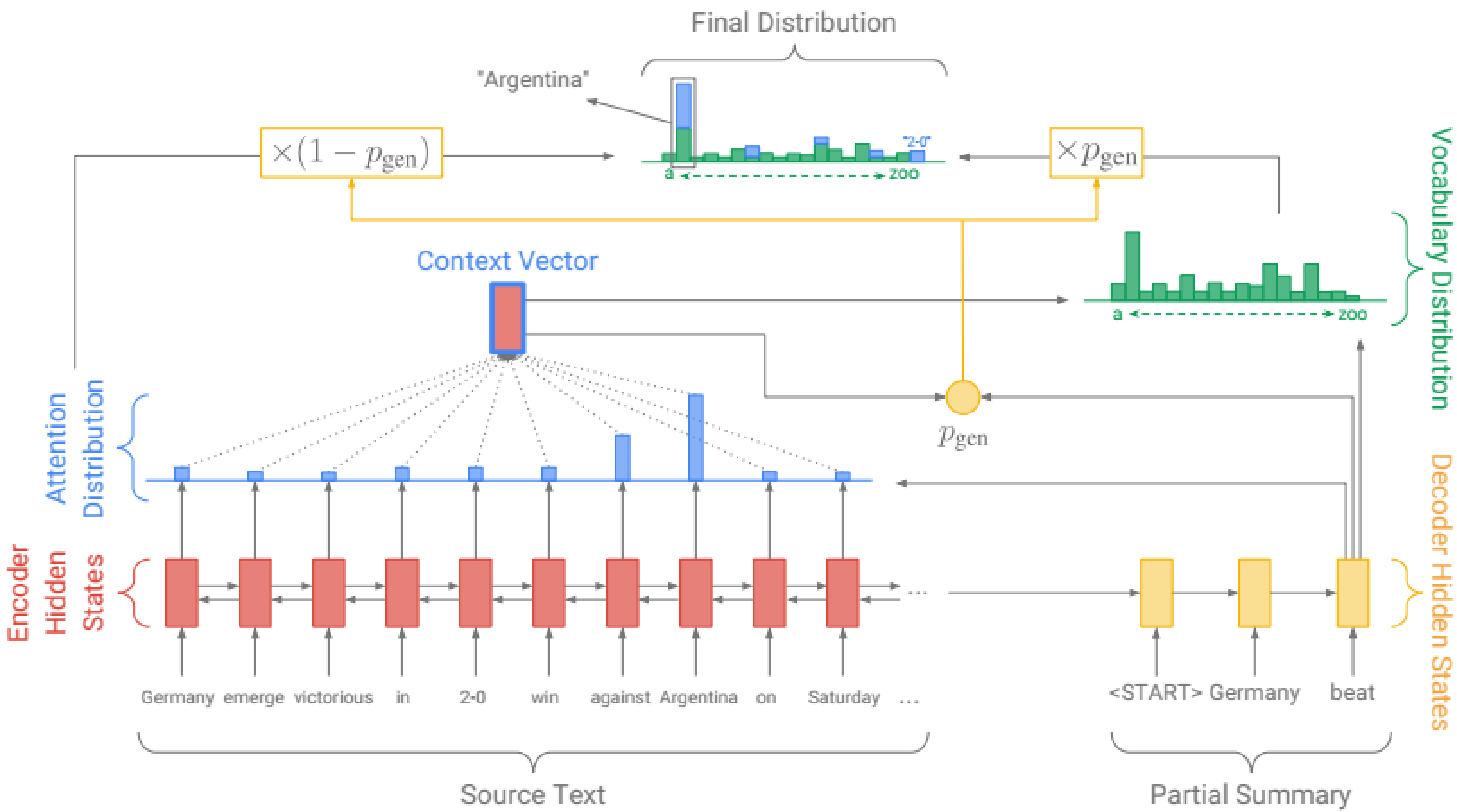


Figure 3: Pointer-generator model. For each decoder timestep a generation probability $p_{\text{gen}} \in [0,1]$ is calculated, which weights the probability of *generating* words from the vocabulary, versus *copying* words from the source text. The vocabulary distribution and the attention distribution are weighted and summed to obtain the final distribution, from which we make our prediction. Note that out-of-vocabulary article

Modeling: Generator Probability

- Allows both copying words via pointing, and generating words from a fixed vocabulary.
- *Generation probability* $p_{gen} \in [0,1]$ for timestep t is calculated from the context vector h_t^*, the decoder state s_t and the decoder input x_t

$$p_{\mathrm{gen}} = \sigma(w_{h^*}^T h_t^* + w_s^T s_t + w_x^T x_t + b_{\mathrm{ptr}}) \qquad (8)$$

where vectors w_{h^*}, w_s, w_x and scalar b_{ptr} are learnable parameters and σ is the sigmoid function.

Modeling: to point or to generate

- p_{gen} is used as a soft switch to choose between *generating* a word from the vocabulary by sampling from P_{vocab}, or *copying* a word from the input sequence by sampling from the attention distribution a_t

$$P(w) = p_{\text{gen}} P_{\text{vocab}}(w) + (1 - p_{\text{gen}}) \sum_{i:w_i = w} a_i^t \quad (9)$$

- if w is an out-of-vocabulary (OOV) word, then Pvocab(w) is zero; similarly if w does not appear in the source document, then $\sum_{i:w_i=w} a_t^i$ is zero.

Result: superiority of pointer-generator

	ROUGE			METEOR	
	1	2	L	exact match	+ stem/syn/para
abstractive model (Nallapati et al., 2016)*	35.46	13.30	32.65	-	-
seq-to-seq + attn baseline (150k vocab)	30.49	11.17	28.08	11.65	12.86
seq-to-seq + attn baseline (50k vocab)	31.33	11.81	28.83	12.03	13.20
pointer-generator	36.44	15.66	33.42	15.35	16.65
pointer-generator + coverage	**39.53**	**17.28**	**36.38**	17.32	18.72
lead-3 baseline (ours)	40.34	17.70	36.57	20.48	22.21
lead-3 baseline (Nallapati et al., 2017)*	39.2	15.7	35.5	-	-
extractive model (Nallapati et al., 2017)*	39.6	16.2	35.3	-	-

Table 1: ROUGE F_1 and METEOR scores on the test set. Models and baselines in the top half are abstractive, while those in the bottom half are extractive. Those marked with * were trained and evaluated on the anonymized dataset, and so are not strictly comparable to our results on the original text. All our ROUGE scores have a 95% confidence interval of at most ± 0.25 as reported by the official ROUGE script. The METEOR improvement from the 50k baseline to the pointer-generator model, and from the pointer-generator to the pointer-generator+coverage model, were both found to be statistically significant using an approximate randomization test with $p < 0.01$.

Giving importance to Recall: Ref n-grams: ROUGE

ROUGE

- **R**ecall-**O**riented **U**nderstudy for **G**isting **E**valuation
- ROUGE is a package of metrics: ROUGE-N, ROUGE-L, ROUGE-W and ROUGE-S

ROUGE-N

ROUGE-N

$$= \frac{\sum_{S \in \{ReferemceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\boxed{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)}}$$

$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{n\text{-}gram \in C} Count_{clip}(n\text{-}gram)}{\boxed{\sum_{C' \in \{Candidates\}} \sum_{n\text{-}gram' \in C'} Count(n\text{-}gram')}}$$

ROUGE-N incorporates Recall

Will BLEU be able to understand quality of long sentences?

Reference translation:
क्या ब्लू लंबे वाक्य की गुणवत्ता को समझ पाएगा?
Kya bloo lambe waakya ki guNvatta ko samajh paaega?

Candidate translation:
लंबे वाक्य
Lambe vaakya

ROUGE-N: 1 / 8
Modified n-gram Precision: 1

Other ROUGEs

- ROUGE-L
  – Considers longest common subsequence
- ROUGE-W
  – Weighted ROUGE-L: All common subsequences are considered with weight based on length
- ROUGE-S
  – Precision/Recall by matching skip bigrams

ROUGE v/s BLEU

	ROUGE	BLEU
Handling incorrect words	Skip bigrams, ROUGE-N	N-gram mismatch
Handling incorrect word order	Longest common sub-sequence	N-gram mismatch
Handling recall	ROUGE-N incorporates missing words	Precision cannot detect 'missing' words. Hence, brevity penalty!

$$\text{ROUGE-N} = \frac{\sum_{S \in \{ReferemceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)}$$

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n \right)$$