The Referential Reader: A Recurrent Entity Network for Anaphora Resolution

Fei Liu *
The University of Melbourne
Victoria, Australia

Luke Zettlemoyer
Facebook AI Research
University of Washington
Seattle, USA

Jacob Eisenstein
Facebook AI Research
Seattle, USA

Abstract

We present a new architecture for storing and accessing entity mentions during online text processing. While reading the text, entity references are identified, and may be stored by either updating or overwriting a cell in a fixed-length memory. The update operation implies coreference with the other mentions that are stored in the same cell; the overwrite operation causes these mentions to be forgotten. By encoding the memory operations as differentiable gates, it is possible to train the model end-to-end, using both a supervised anaphora resolution objective as well as a supplementary language modeling objective. Evaluation on a dataset of pronoun-name anaphora demonstrates strong performance with purely incremental text processing.

1 Introduction

Reference resolution is fundamental to language understanding. Current state-of-the-art systems employ the *mention-pair* model, in which a classifier is applied to all pairs of spans (e.g., Lee et al., 2017). This approach is expensive in both computation and labeled data, and it is also cognitively implausible: human readers interpret text in a nearly online fashion (Tanenhaus et al., 1995).

We present a new method for reference resolution, which reads the text left-to-right while storing entities in a fixed-size working memory (Figure 1). As each token is encountered, the reader must decide whether to: (a) link the token to an existing memory, thereby creating a coreference link, (b) overwrite an existing memory and store a new entity, or (c) disregard the token and move ahead. As memories are reused, their salience increases, making them less likely to be overwritten.

This online model for coreference resolution is based on the memory network architecture (Weston et al., 2015), in which memory operations are differentiable, enabling end-to-end training from gold anaphora resolution data. Furthermore, the memory can be combined with a recurrent hidden state, enabling prediction of the next word. This makes it possible to train the model from unlabeled data using a language modeling objective.

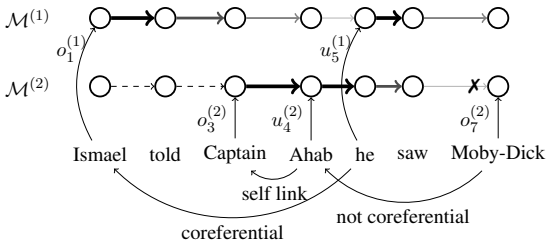

Figure 1: A referential reader with two memory cells. Overwrite and update are indicated by $o_t^{(i)}$ and $u_t^{(i)}$; in practice, these operations are continuous gates. Thickness and color intensity of edges between memory cells at neighboring steps indicate memory salience; ✗ indicates an overwrite.

To summarize, we present a model that processes the text incrementally, resolving references on the fly (Schlangen et al., 2009). The model yields promising results on the GAP dataset of pronoun-name references.[1]

2 Model

For a given document consisting of a sequence of tokens $\{w_t\}_{t=1}^T$, we represent text at two levels:

- Tokens: represented as $\{\boldsymbol{x}_t\}_{t=1}^T$, where the vector $\boldsymbol{x}_t \in \mathbb{R}^{D_x}$ is computed from any token-level encoder.
- Entities: represented by a fixed-length memory $\mathcal{M}_t = \{(\boldsymbol{k}_t^{(i)}, \boldsymbol{v}_t^{(i)}, s_t^{(i)})\}_{i=1}^N$, where each memory is a tuple of a key $\boldsymbol{k}_t^{(i)} \in \mathbb{R}^{D_k}$, a

*Work carried out as an intern at Facebook AI Research

[1]Code available at: https://github.com/liufly/refreader

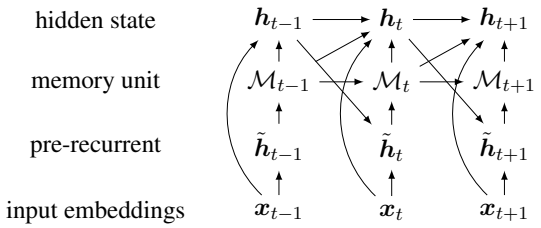

Figure 2: Overview of the model architecture.

value $\boldsymbol{v}_t^{(i)} \in \mathbb{R}^{D_v}$, and a salience $s_t^{(i)} \in [0, 1]$.

There are two components to the model: the memory unit, which stores and tracks the states of the entities in the text; and the recurrent unit, which controls the memory via a set of gates. An overview is presented in Figure 2.

2.1 Recurrent Unit

The recurrent unit is inspired by the Coreferential-GRU, in which the current hidden state of a gated recurrent unit (GRU; Chung et al., 2014) is combined with the state at the time of the most recent mention of the current entity (Dhingra et al., 2018). However, instead of relying on the coreferential structure to construct a dynamic computational graph, we use an external memory unit to keep track of previously mentioned entities and let the model learn to decide what to store in each cell.

The memory state is summarized by the weighted sum over values: $\boldsymbol{m}_t = \sum_{i=1}^{N} s^{(i)} \boldsymbol{v}_t^{(i)}$. The current hidden state and the input are combined into a **pre-recurrent state** $\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}\boldsymbol{h}_{t-1} + \boldsymbol{U}\boldsymbol{x}_t)$, which is used to control the memory operations; the matrices $\boldsymbol{W}$ and $\boldsymbol{U}$ are trained parameters. To compute the next hidden state $\boldsymbol{h}_t$, we perform a recurrent update:

$$\boldsymbol{h}_t = \text{GRU}(\boldsymbol{x}_t, (1 - c_t) \times \boldsymbol{h}_{t-1} + c_t \times \boldsymbol{m}_t) \quad (1)$$

where $c_t = \min(\sigma(\boldsymbol{W}_c \tilde{\boldsymbol{h}}_t + b_c), \sum_i s_t^{(i)})$ is a gate that measures the importance of the memory network to the current token. This gate is a sigmoid function of the pre-recurrent state, clipped by the sum of memory saliences. This ensures that the memory network is used only when at least some memories are salient.

2.2 Memory Unit

The memory gates are a collection of scalars $\{(u_t^{(i)}, o_t^{(i)})\}_{i=1}^{N}$, indicating **u**pdates and **o**verwrites to cell i at token w_t. To compute these gates, we first determine whether w_t is an entity mention, using a sigmoid-activated gate $e_t = \sigma(\boldsymbol{\phi}_e \cdot \tilde{\boldsymbol{h}}_t)$, where $\boldsymbol{\phi}_e \in \mathbb{R}^{D_h}$ is a learnable vector. We next decide whether w_t refers to a *previously* mentioned entity: $r_t = \sigma(\boldsymbol{\phi}_r \cdot \tilde{\boldsymbol{h}}_t) \times e_t$, where $\boldsymbol{\phi}_r \in \mathbb{R}^{D_h}$ is a learnable vector.

Updating existing entities. If w_t is a referential entity mention ($r_t \approx 1$), it may refer to an entity in the memory. To compute the compatibility between w_t and each memory, we first summarize the current state as a query vector, $\boldsymbol{q}_t = f_q(\tilde{\boldsymbol{h}}_t)$, where f_q is a two-layer feed-forward network. The query vector is then combined with the memory keys and the reference gate to obtain attention scores, $\alpha_t^{(i)} = r_t \times \text{SoftMax}(\boldsymbol{k}_{t-1}^{(i)} \cdot \boldsymbol{q}_t + b)$, where the softmax is computed over all cells i, and b is a learnable bias term, inversely proportional to the likelihood of introducing a new entity. The update gate is then set equal to the query match $\alpha_t^{(i)}$, clipped by the salience, $u_t^{(i)} = \min(\alpha_t^{(i)}, 2s_{t-1}^{(i)})$. The upper bound of $2s_{t-1}^{(i)}$ ensures that an update can at most triple the salience of a memory.

Storing new entities. Overwrite operations are used to store new entities. The total amount to overwrite is $\tilde{o}_t = e_t - \sum_{i=1}^{N} u_t^{(i)}$, which is the difference between the entity gate and the sum of the update gates. We prefer to overwrite the memory with the lowest salience. This decision is made differentiable using the Gumbel-softmax distribution (GSM; Jang et al., 2017), $o_t^{(i)} = \tilde{o}_t \times \text{GSM}^{(i)}(-\boldsymbol{s}_{t-1}, \tau)$ and $\boldsymbol{s}_t = \{s_t^{(i)}\}_{i=1}^{N}$.[2]

Memory salience. To the extent that each memory is not updated or overwritten, it is copied along to the next timestep. The weight of this copy operation is: $r_t^{(i)} = 1 - u_t^{(i)} - o_t^{(i)}$. The salience decays exponentially,

$$\lambda_t = (e_t \times \gamma_e + (1 - e_t) \times \gamma_n) \quad (2)$$

$$s_t^{(i)} = \lambda_t \times r_t^{(i)} \times s_{t-1}^{(i)} + u_t^{(i)} + o_t^{(i)}, \quad (3)$$

where γ_e and γ_n represent the salience decay rate upon seeing an entity or non-entity.[3]

[2] Here τ is the "temperature" of the distribution, which is gradually decreased over the training period, until the distribution approaches a one-hot vector indicating the argmax.

[3] We set $\gamma_e = \exp(\log(0.5)/\ell_e)$ with $\ell_e = 4$ denoting the entity half-life, which is the number of entity mentions before the salience decreases by half. The non-entity halflife γ_n is computed analogously, with $\ell_n = 30$.

Memory state. To update the memory states, we first transform the pre-recurrent state $\tilde{\boldsymbol{h}}_t$ into the memory domain, obtaining overwrite candidates for the keys and values, $\tilde{\boldsymbol{k}}_t = f_k(\tilde{\boldsymbol{h}}_t)$ and $\tilde{\boldsymbol{v}}_t = f_v(\tilde{\boldsymbol{h}}_t)$, where f_k is a two-layer residual network with $\tanh$ nonlinearities, and f_v is a linear projection with a $\tanh$ non-linearity. Update candidates are computed by GRU recurrence with the overwrite candidate as the input. This yields the state update,

$$\boldsymbol{k}_t^{(i)} = u_t^{(i)}\text{GRU}_k(\boldsymbol{k}_{t-1}^{(i)}, \tilde{\boldsymbol{k}}_t) + o_t^{(i)}\tilde{\boldsymbol{k}}_t + r_t^{(i)}\boldsymbol{k}_{t-1}^{(i)}$$
$$\boldsymbol{v}_t^{(i)} = u_t^{(i)}\text{GRU}_v(\boldsymbol{v}_{t-1}^{(i)}, \tilde{\boldsymbol{v}}_t) + o_t^{(i)}\tilde{\boldsymbol{v}}_t + r_t^{(i)}\boldsymbol{v}_{t-1}^{(i)}.$$

2.3 Coreference Chains

To compute the probability of coreference between the mentions w_{t_1} and w_{t_2}, we first compute the probability that each cell i refers to the same entity at both of those times,

$$\omega_{t_1,t_2}^{(i)} = \prod_{t=t_1+1}^{t_2} (1 - o_t^{(i)}) \tag{4}$$

Furthermore, the probability that mention t_1 is stored in memory i is $u_{t_1}^{(i)} + o_{t_1}^{(i)}$. The probability that two mentions corefer is then the sum over memory cells,

$$\hat{\psi}_{t_1,t_2} = \sum_{i=1}^{N} (u_{t_1}^{(i)} + o_{t_1}^{(i)}) \times u_{t_2}^{(i)} \times \omega_{t_1,t_2}^{(i)}. \tag{5}$$

2.4 Training

The coreference probability defined in Equation 5 is a differentiable function of the gates, which in turn are computed from the inputs $w_1, w_2, \ldots w_T$. We can therefore train the entire network end-to-end from a cross-entropy objective, where a loss is incurred for incorrect decisions on the level of token pairs. Specifically, we set $y_{i,j} = 1$ when w_i and w_j corefer (coreferential links), and also when both w_i and w_j are part of the same mention span (self links). The coreference loss is then the cross-entropy $\sum_{i=1}^{T}\sum_{j=i+1}^{T} H(\hat{\psi}_{i,j}, y_{i,j})$.

Because the hidden state $\boldsymbol{h}_t$ is computed recurrently from $w_{1:t}$, the reader can also be trained from a language modeling objective, even when coreference annotations are unavailable. Word probabilities $P(w_{t+1} \mid \boldsymbol{h}_t)$ are computed by projecting the hidden state $\boldsymbol{h}_t$ by a matrix of output embeddings, and applying the softmax operation.

3 Experiments

As an evaluation of the ability of the referential reader to correctly track entity references in text, we evaluate against the GAP dataset, recently introduced by Webster et al. (2018). Each instance consists of: (1) a sequence of tokens $w_1, \ldots, w_T$ extracted from Wikipedia biographical pages; (2) two person names (A and B, whose token index spans are denoted s_A and s_B); (3) a single-token pronoun (P with the token index s_P); and (4) two binary labels (y_A and y_B) indicating whether P is referring to A or B.

Language modeling. Given the limited size of GAP, it is difficult to learn a strong recurrent model. We therefore consider the task of language modeling as a pre-training step. We make use of the page text of the original Wikipedia articles from GAP, the URLs to which are included as part of the data release. This results in a corpus of 3.8 million tokens, which is used for pre-training. The reader is free to use the memory to improve its language modeling performance, but it receives no supervision on the coreference links that might be imputed on this unlabeled data.

Prediction. At test time, we make coreference predictions using the procedure defined in § 2.3. Following Webster et al. (2018), we do not require exact string match for mention detection: if the selected candidate is a substring of the gold span, we consider it as a predicted coreferential link between the pronoun and person name. Concretely, we focus on the token index s_P of the pronoun and predict the positive coreferential relation of the pronoun P and person name A if any (in the span of s_A) of $\hat{\psi}_{s_A,s_P}$ (if $s_A < s_P$) or $\hat{\psi}_{s_P,s_A}$ (otherwise) is greater than a threshold value (selected on the validation set).[4]

Evaluation. Performance is measured on the GAP test set, using the official evaluation script. We report the overall F_1, as well as the scores by gender (**M**asculine: F_1^M and **F**eminine: F_1^F), and the bias (the ratio of F_1^F to F_1^M: $\frac{F_1^F}{F_1^M}$).

Systems. We benchmark our model (RefReader) against a collection of strong baselines presented in the work of Webster et al. (2018): (1) a state-of-the-art mention-pair coreference resolution (Lee

[4] As required by Webster et al. (2018), the model is responsible for detecting mentions; only the scoring function accesses labeled spans.

	F_1^M	F_1^F	$\frac{F_1^F}{F_1^M}$	F_1
Clark and Manning (2015)†	53.9	52.8	0.98	53.3
Lee et al. (2017)†	67.7	60.0	0.89	64.0
Lee et al. (2017), re-trained	67.8	66.3	0.98	67.0
Parallelism†	69.4	64.4	0.93	66.9
Parallelism+URL†	72.3	68.8	0.95	70.6
RefReader, LM objective‡	61.6	60.5	0.98	61.1
RefReader, coref objective‡	69.6	68.1	0.98	68.9
RefReader, LM + coref‡	**72.8**	**71.4**	**0.98**	**72.1**
RefReader, coref + BERT⋆	**80.3**	**77.4**	**0.96**	**78.8**

Table 1: GAP test set performance. †: reported in Webster et al. (2018); ‡: strictly incremental processing; ⋆: average over 5 runs.

et al., 2017); (2) a version of (1) that is retrained on GAP; (3) a rule-based system based on syntactic parallelism (Webster et al., 2018); (4) a domain-specific variant of (3) that incorporates the lexical overlap between each candidate and the title of the original Wikipedia page (Webster et al., 2018). We evaluate a configuration of RefReader that uses two memory cells; other details are in the supplement (Appendix A).

Results. As shown in Table 1, RefReader achieves state-of-the-art performance, outperforming strong pretrained and retrained systems (e.g., Lee et al., 2017), as well as domain-specific heuristics (Parellelism+URL). Language model pretraining yields an absolute gain of 3.2 in F_1. This demonstrates the ability of RefReader to leverage unlabeled text, which is a distinctive feature in comparison with prior work. When training is carried out in the unsupervised setting (with the language modeling objective only), the model is still capable of learning the latent coreferential structure between pronouns and names to some extent, outperforming a supervised coreference system that gives competitive results on OntoNotes (Clark and Manning, 2015).

We also test a combination of RefReader and BERT (Devlin et al., 2019), using BERT's contextualized word embeddings as base features $\boldsymbol{x}_t$ (concatenation of the top 4 layers), which yields substantial improvements in accuracy. While this model still resolves references incrementally, it cannot be said to be *purely* incremental, because BERT uses "future" information to build its contextualized embeddings.[5] Note that the gender bias increases slightly, possibly due to bias in the data used to train BERT.

GAP examples are short, containing just a few entity mentions. To test the applicability of our method to longer instances, we produce an alternative test set in which pairs of GAP instances are concatenated together, doubling the average number of tokens and entity mentions. Even with a memory size of two, performance drops to $F_1 = 70.2$ (from 72.1 on the original test set). This demonstrates that the model is capable of reusing memory cells when the number of entities is larger than the size of the memory. We also test a configuration of RefReader with four memory cells, and observe that performance on the original test set decreases only slightly, to $F_1 = 71.4$ (against RefReader LM + coref).

Case study and visualization. Figure 3 gives an example of the behavior of the referential reader, as applied to a concatenation of two instances from GAP.[6] The top panel shows the salience of each entity as each token is consumed, with the two memory cells distinguished by color and marker. The figure elides long spans of tokens whose gate activations are nearly zero. These tokens are indicated in the x-axis by ellipsis; the corresponding decrease in salience is larger, because it represents a longer span of text. The bottom panel shows the gate activations for each token, with memory cells again distinguished by color and marker, and operations distinguished by line style. The gold token-entity assignments are indicated with color and superscript.

The reader essentially ignores the first name, *Braylon Edwards*, making a very weak overwrite to memory 0 (m0). It then makes a large overwrite to m0 on the pronoun *his*. When encountering the token *Avant*, the reader makes an update to the same memory cell, creating a cataphoric link between *Avant* and *his*. The name *Padbury* appears much later (as indicated by the ellipsis), and at this point, m0 has lower salience than m1. For this reason, the reader chooses to overwrite m0 with this name. The reader ignores the name *Cathy Vespers* and overwrites m1 with the adverb *coincidentally*. On encountering the final pronoun *she*, the reader is conflicted, and makes a partial

[5] Future work may explore the combination of RefReader and large-scale pretrained incremental language models (e.g., Radford et al., 2019).

[6] For an example involving multi-token spans, see Appendix B.

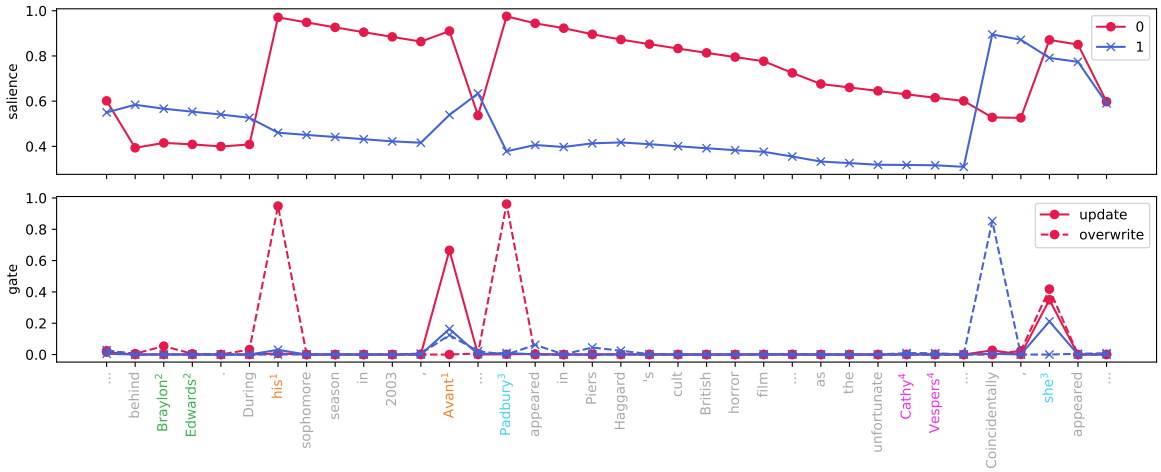

Figure 3: An example of the application the referential reader to a concatenation of two instances from GAP. The ground truth is indicated by the color of each token on the x-axis as well as the superscript.

overwrite to m0, a partial update (indicating coreference with *Padbury*), and a weaker update to m1. If the update to m0 is above the threshold, then the reader may receive credit for this coreference edge, which would otherwise be scored as a false negative.

The reader ignores the names *Braylon Edwards*, *Piers Haggard*, and *Cathy Vespers*, leaving them out of the memory. *Edwards* and *Vespers* appear in prepositional phrases, while *Haggard* is a possessive determiner of the object of a prepositional phrase. Centering theory argues that these syntactic positions have low salience in comparison with subject and object position (Grosz et al., 1995). It is possible that the reader has learned this principle, and that this is why it chooses not to store these names in memory. However, the reader also learns from the GAP supervision that pronouns are important, and therefore stores the pronoun *his* even though it is also a possessive determiner.

4 Related Work

Memory networks provide a general architecture for online updates to a set of distinct memories (Weston et al., 2015; Sukhbaatar et al., 2015). The link between memory networks and incremental text processing was emphasized by Cheng et al. (2016). Henaff et al. (2017) used memories to track the states of multiple entities in a text, but they predefined the alignment of entities to memories, rather than learning to align entities with memories using gates. The incorporation of entities into language models has also been explored in prior work (Yang et al., 2017; Kobayashi et al., 2017); similarly, Dhingra et al. (2018) augment the gated recurrent unit (GRU) architecture with additional edges between coreferent mentions. In general, this line of prior work assumes that coreference information is available at test time (e.g., from a coreference resolution system), rather than determining coreference in an online fashion. Ji et al. (2017) propose a generative entity-aware language model that incorporates coreference as a discrete latent variable. For this reason, importance sampling is required for inference, and the model cannot be trained on unlabeled data.

5 Conclusion

This paper demonstrates the viability of incremental reference resolution, using an end-to-end differentiable memory network. This enables semi-supervised learning from a language modeling objective, which substantially improves performance. A key question for future work is the performance on longer texts, such as the full-length news articles encountered in OntoNotes. Another direction is to further explore semi-supervised learning, by reducing the amount of training data and incorporating linguistically-motivated constraints based on morphosyntactic features.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable feedback, Yinhan Liu, Abdelrahman Mohamed, Omer Levy, Kellie Webster, Vera Axelrod, Mandar Joshi, Trevor Cohn and Timothy Baldwin for their help and comments.

References

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Proceedings of the NIPS 2014 Deep Learning and Representation Learning Workshop*.

Kevin Clark and Christopher D. Manning. 2015. Entity-centric coreference resolution with model stacking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1405–1415.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Bhuwan Dhingra, Qiao Jin, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2018. Neural models for reasoning over multiple mentions using coreference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 42–48. Association for Computational Linguistics.

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

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2017. Tracking the world state with recurrent entity networks. In *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *Proceedings of the 5th International Conference on Learning Representations*.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A. Smith. 2017. Dynamic entity representations in neural language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1830–1839. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Sosuke Kobayashi, Naoaki Okazaki, and Kentaro Inui. 2017. A neural language model for dynamically representing the meanings of unknown words and entities in a discourse. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 473–483, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. `https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf`.

David Schlangen, Timo Baumann, and Michaela Atterer. 2009. Incremental reference resolution: The task, metrics for evaluation, and a Bayesian filtering model that is sensitive to disfluencies. In *Proceedings of the SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 30–37.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Proceedings of Advances in Neural Information Processing Systems*, pages 2440–2448, Montréal, Canada.

Michael K Tanenhaus, Michael J Spivey-Knowlton, Kathleen M Eberhard, and Julie C Sedivy. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268(5217):1632–1634.

Kellie Webster, Marta Recasens, Vera Axelrod, and Jason Baldridge. 2018. Mind the GAP: A balanced corpus of gendered ambiguous pronouns. *Transactions of the Association for Computational Linguistics*, 6:605–617.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, USA.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-aware language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1850–1859.

A Supplemental information

Model configuration. Training is carried out on the development set of GAP with the Adam optimizer (Kingma and Ba, 2014) and a learning rate of 0.001. Early stopping is applied based on the performance on the validation set. We use the following hyperparameters:

- embedding size $D_x = 300$;
- memory key size $D_k = 16$ (32 with BERT) and value size $D_v = 300$; the hidden layers in the memory key/value updates f_k and f_v are also set to 16 (32 with BERT) and 300 respectively;
- number of memory cells $N = 2$;
- pre-recurrent and hidden state sizes $D_h = 300$;
- salience half-life for words and entity mentions are 30 and 4 respectively;
- Gumbel softmax starts at temperature $\tau = 1.0$ with an exponential decay rate of 0.5 applied every 10 epochs;
- dropout is applied to the embedding layer, the pre-recurrent state $\tilde{\boldsymbol{h}}_t$, and the GRU hidden state $\boldsymbol{h}_t$, with a rate of 0.5;
- self and coreferential links are weighted differently in the coreference loss cross-entropy in § 2.4 with 0.1 and 5.0 and negative coreferential links weighted higher than positive ones with a ratio of 10:1 to penalize false positive predictions.

For the RefReader model trained only on coreference annotations, the base word embeddings ($\boldsymbol{x}_t$) are fixed to the pretrained GloVe embeddings (Pennington et al., 2014). In the RefReader models that include language model pretraining, embeddings are learned on the language modeling task. Language modeling pre-training is carried out using the same configuration as above; the embedding update and early stopping are based on perplexity on a validation set.

B Multi-token Span Example

In the example shown in Figure 4, the system must handle multi-token spans *Paul Sabatier* and *Wilhelm Normann*. It does this by overwriting on the first token, and updating on the second token, indicating that both tokens are part of the name of a single entity. The reader also correctly handles an example of cataphora (*During **his** tenure, **Smith** voted* ...). It stores *Paul Sabatier* in the same memory as *Smith*, but overwrites that memory so as not to create a coreference link. The reader reuses memory one for both entities because in the intervening text, memory zero acquired more salience. Finally, the model perceives some ambiguity on the pronoun *he* at the end: it narrowly favors coreference with *Normann*, but assigns some probability to the creation of a new entity.

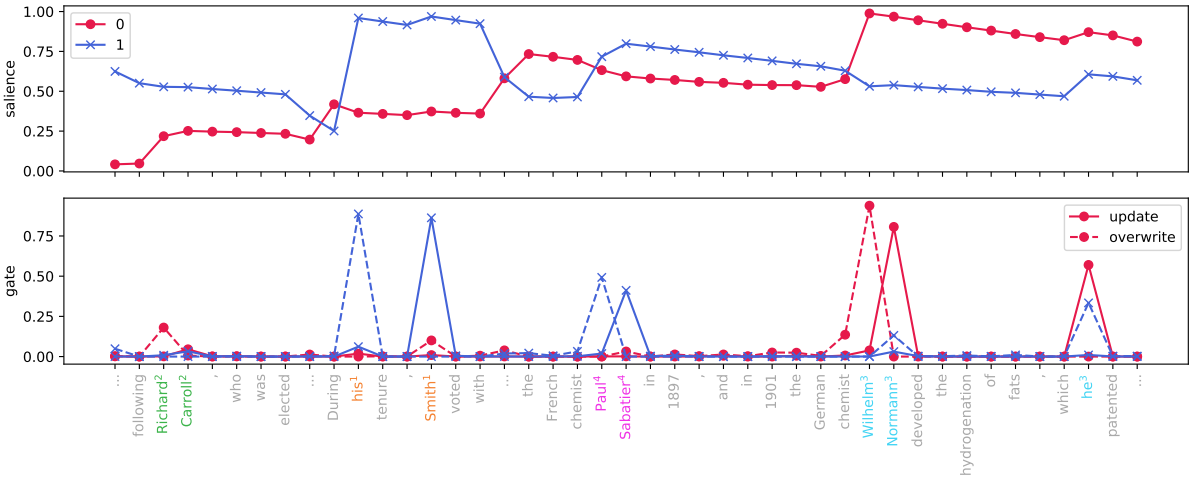

Figure 4: Another example of the referential reader, as applied to a concatenation of two instances from GAP. Again, the ground truth is indicated by the color of each token on the x-axis as well as the super-script.