PeTra: A Sparsely Supervised Memory Model for People Tracking

Shubham Toshniwal[1], Allyson Ettinger[2], Kevin Gimpel[1], Karen Livescu[1]
[1]Toyota Technological Institute at Chicago
[2]Department of Linguistics, University of Chicago
{shtoshni, kgimpel, klivescu}@ttic.edu, aettinger@uchicago.edu

Abstract

We propose PeTra, a memory-augmented neural network designed to track entities in its memory slots. PeTra is trained using sparse annotation from the GAP pronoun resolution dataset and outperforms a prior memory model on the task while using a simpler architecture. We empirically compare key modeling choices, finding that we can simplify several aspects of the design of the memory module while retaining strong performance. To measure the people tracking capability of memory models, we (a) propose a new diagnostic evaluation based on counting the number of unique entities in text, and (b) conduct a small scale human evaluation to compare evidence of people tracking in the memory logs of PeTra relative to a previous approach. PeTra is highly effective in both evaluations, demonstrating its ability to track people in its memory despite being trained with limited annotation.

1 Introduction

Understanding text narratives requires maintaining and resolving entity references over arbitrary-length spans. Current approaches for coreference resolution (Clark and Manning, 2016b; Lee et al., 2017, 2018; Wu et al., 2019) scale quadratically (without heuristics) with length of text, and hence are impractical for long narratives. These models are also cognitively implausible, lacking the incrementality of human language processing (Tanenhaus et al., 1995; Keller, 2010). Memory models with finite memory and online/quasi-online entity resolution have linear runtime complexity, offering more scalability, cognitive plausibility, and interpretability.

Memory models can be viewed as general problem solvers with external memory mimicking a Turing tape (Graves et al., 2014, 2016). Some of the earliest applications of memory networks in language understanding were for question answering, where the external memory simply stored all of the word/sentence embeddings for a document (Sukhbaatar et al., 2015; Kumar et al., 2016). To endow more structure and interpretability to memory, key-value memory networks were introduced by Miller et al. (2016). The key-value architecture has since been used for narrative understanding and other tasks where the memory is intended to learn to track entities while being guided by varying degrees of supervision (Henaff et al., 2017; Liu et al., 2018a,b, 2019a).

We propose a new memory model, PeTra, for entity tracking and coreference resolution, inspired by the recent Referential Reader model (Liu et al., 2019a) but substantially simpler. Experiments on the GAP (Webster et al., 2018) pronoun resolution task show that PeTra outperforms the Referential Reader with fewer parameters and simpler architecture. Importantly, while Referential Reader performance degrades with larger memory, PeTra improves with increase in memory capacity (before saturation), which should enable tracking of a larger number of entities. We conduct experiments to assess various memory architecture decisions, such as learning of memory initialization and separation of memory slots into key/value pairs.

To test interpretability of memory models' entity tracking, we propose a new diagnostic evaluation based on entity counting—a task that the models are not explicitly trained for—using a small amount of annotated data. Additionally, we conduct a small scale human evaluation to assess quality of people tracking based on model memory logs. PeTra substantially outperforms Referential Reader on both measures, indicating better and more interpretable tracking of people.[1]

[1]Code available at https://github.com/shtoshni92/petra

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 5415–5428
July 5 - 10, 2020.

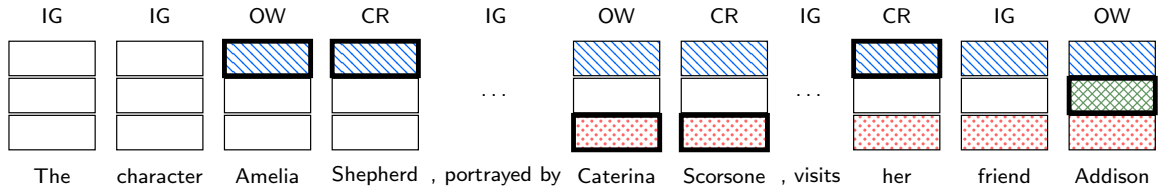


Figure 1: Illustration of memory cell updates in an example sentence where IG = ignore, OW = overwrite, CR = coref. Different patterns indicate the different entities, and an empty pattern indicates that the cell has not been used. The updated memory cells at each time step are highlighted.

2 Model

Figure 2 depicts PeTra, which consists of three components: an *input encoder* that given the tokens generates the token embeddings, a *memory module* that tracks information about the entities present in the text, and a *controller network* that acts as an interface between the encoder and the memory.

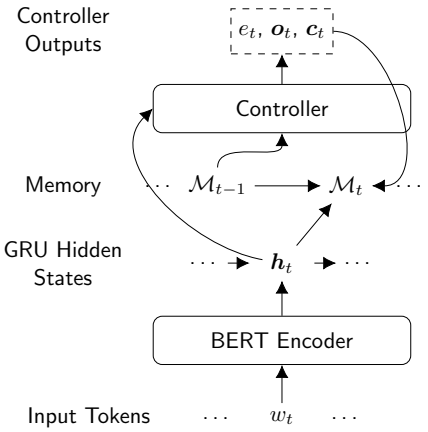


Figure 2: Proposed model.

2.1 Input Encoder

Given a document consisting of a sequence of tokens $\{w_1, \cdots, w_T\}$, we first pass the document through a fixed pretrained BERT model (Devlin et al., 2019) to extract contextual token embeddings. Next, the BERT-based token embeddings are fed into a single-layer unidirectional Gated Recurrent Unit (GRU) (Cho et al., 2014) running left-to-right to get task-specific token embeddings $\{\boldsymbol{h}_1, \cdots, \boldsymbol{h}_T\}$.

2.2 Memory

The memory $\mathcal{M}_t$ consists of N memory cells. The i^{th} memory cell state at time step t consists of a tuple $(\boldsymbol{m}_t^i, u_t^i)$ where the vector $\boldsymbol{m}_t^i$ represents the content of the memory cell, and the scalar $u_t^i \in [0, 1]$ represents its recency of usage. A high value of u_t^i is intended to mean that the cell is tracking an entity that has been recently mentioned.

Initialization Memory cells are initialized to the null tuple, i.e. $(\mathbf{0}, 0)$; thus, our memory is parameter-free. This is in contrast with previous entity tracking models such as EntNet (Henaff et al., 2017) and the Referential Reader (Liu et al., 2019a) where memory initialization is learned and the cells are represented with separate *key* and *value* vectors. We will later discuss variants of our memory with some of these changes.

2.3 Controller

At each time step t the controller network determines whether token t is part of an entity span and, if so, whether the token is coreferent with any of the entities already being tracked by the memory. Depending on these two variables, there are three possible actions:

(i) IGNORE: The token is not part of any entity span, in which case we simply ignore it.

(ii) OVERWRITE: The token is part of an entity span *but* is not already being tracked in the memory.

(iii) COREF: The token is part of an entity span and the entity is being tracked in the memory.

Therefore, the two ways of updating the memory are OVERWRITE and COREF. There is a strict ordering constraint to the two operations: OVERWRITE precedes COREF, because it is not possible to corefer with a memory cell that is not yet tracking anything. That is, the COREF operation cannot be applied to a previously unwritten memory cell, i.e. one with $u_t^i = 0$. Figure 1 illustrates an idealized version of this process.

Next we describe in detail the computation of the probabilities of the two operations for each memory cell at each time step t.

First, the **entity mention probability** e_t, which reflects the probability that the current token w_t is part of an entity mention, is computed by:

$$e_t = \sigma(\text{MLP}_1(\boldsymbol{h}_t)) \tag{1}$$

where MLP_1 is a multi-layer perceptron and σ is the logistic function.

Overwrite and Coref If the current token w_t is part of an entity mention, we need to determine whether it corresponds to an entity being currently tracked by the memory or not. For this we compute the similarity between the token embedding $\boldsymbol{h}_t$ and the contents of the memory cells currently tracking entities. For the i^{th} memory cell with memory vector $\boldsymbol{m}^i_{t-1}$ the similarity with $\boldsymbol{h}_t$ is given by:

$$sim^i_t = \text{MLP}_2([\boldsymbol{h}_t; \boldsymbol{m}^i_{t-1}; \boldsymbol{h}_t \odot \boldsymbol{m}^i_{t-1}; u^i_{t-1}]) \tag{2}$$

where MLP_2 is a second MLP and $\odot$ is the Hadamard (elementwise) product. The usage scalar u^i_{t-1} in the above expression provides a notion of distance between the last mention of the entity in cell i and the potential current mention. The higher the value of u^i_{t-1}, the more likely there was a recent mention of the entity being tracked by the cell. Thus u^i_{t-1} provides an alternative to distance-based features commonly used in pairwise scores for spans (Lee et al., 2017).

Given the entity mention probability e_t and similarity score sim^i_t, we define the **coref score** cs^i_t as:

$$cs^i_t = sim^i_t - \infty \cdot \mathbb{1}[u^i_{t-1} = 0] \tag{3}$$

where the second term ensures that the model does not predict coreference with a memory cell that has not been previously used, something not enforced by Liu et al. (2019a).[2] Assuming the coref score for a new entity to be 0,[3] we compute the **coref probability** c^i_t and **new entity probability** n_t as follows:

$$\begin{pmatrix} c^1_t \\ \vdots \\ c^N_t \\ n_t \end{pmatrix} = e_t \cdot \text{ softmax} \begin{pmatrix} cs^1_t \\ \vdots \\ cs^N_t \\ 0 \end{pmatrix} \tag{4}$$

Based on the memory usage scalars u^i_t and the new entity probability n_t, the **overwrite probability** for each memory cell is determined as follows:

$$o^i_t = n_t \cdot \mathbb{1}_{i = \arg\min_j u^j_{t-1}} \tag{5}$$

Thus we pick the cell with the lowest usage scalar u^j_{t-1} to OVERWRITE. In case of a tie, a cell is picked randomly among the ones with the lowest usage scalar. The above operation is non-differentiable, so during training we instead use

$$o^i_t = n_t \cdot \text{GS}\left(\frac{1 - u^i_{t-1}}{\tau}\right)_i \tag{6}$$

where $\text{GS}(.)$ refers to Gumbel-Softmax (Jang et al., 2017), which makes overwrites differentiable.

For each memory cell, the memory vector is updated based on the three possibilities of ignoring the current token, being coreferent with the token, or considering the token to represent a new entity (causing an overwrite):

$$\boldsymbol{m}^i_t = \overbrace{(1 - (o^i_t + c^i_t))\boldsymbol{m}^i_{t-1}}^{\text{IGNORE}} + \overbrace{o^i_t \cdot \boldsymbol{h}_t}^{\text{OVERWRITE}} + \underbrace{c^i_t \cdot \text{MLP}_3([\boldsymbol{h}_t; \boldsymbol{m}^i_{t-1}])}_{\text{COREF}} \tag{7}$$

In this expression, the coreference term takes into account both the previous cell vector $\boldsymbol{m}^i_{t-1}$ and the current token representation $\boldsymbol{h}_t$, while the overwrite term is based only on $\boldsymbol{h}_t$. In contrast to a similar memory update equation in the Referential Reader which employs a pair of GRUs and MLPs for each memory cell, our update parameter uses just MLP_3 which is memory cell-agnostic.

Finally, the memory usage scalar is updated as

$$u^i_t = \min(1, o^i_t + c^i_t + \gamma \cdot u^i_{t-1}) \tag{8}$$

where $\gamma \in (0, 1)$ is the decay rate for the usage scalar. Thus the usage scalar u^i_t keeps decaying with time unless the memory is updated via OVERWRITE or COREF in which case the value is increased to reflect the memory cell's recent use.

Memory Variants In vanilla PeTra, each memory cell is represented as a single vector and the memory is parameter-free, so the total number of model parameters is independent of memory size. This is a property that is shared with, for example, differentiable neural computers (Graves et al., 2016). On the other hand, recent models for entity tracking, such as the EntNet (Henaff et al., 2017) and the Referential Reader (Liu et al., 2019a), learn

[2] A threshold higher than 0 can also be used to limit coreference to only more recent mentions.

[3] The new entity coref score is a free variable that can be assigned any value, since only the relative value matters.

memory initialization parameters and separate the memory cell into key-value pairs. To compare these memory cell architectures, we investigate the following two variants of PeTra:

1. *PeTra + Learned Initialization*: memory cells are initialized at $t = 0$ to learned parameter vectors.

2. *PeTra + Fixed Key*: a fixed dimensions of each memory cell are initialized with learned parameters and kept fixed throughout the document read, as in EntNet (Henaff et al., 2017).

Apart from initialization, the initial cell vectors are also used to break ties for overwrites in Eqs. (5) and (6) when deciding among unused cells (with $u_t^i = 0$). The criterion for breaking the tie is the similarity score computed using Eq. (2).

2.4 Coreference Link Probability

The probability that the tokens w_{t_1} and w_{t_2} are coreferential according to, say, cell i of the memory depends on three things: (a) w_{t_1} is identified as part of an entity mention and is either overwritten to cell i or is part of an earlier coreference chain for an entity tracked by cell i, (b) Cell i is not overwritten by any other entity mention from $t = t_1 + 1$ to $t = t_2$, and (c) w_{t_2} is also predicted to be part of an entity mention and is coreferential with cell i. Combining these factors and marginalizing over the cell index results in the following expression for the **coreference link probability**:

$$P_{\text{CL}}(w_{t_1}, w_{t_2}) = \sum_{i=1}^{N} (o_{t_1}^i + c_{t_1}^i) \cdot \prod_{j=t_1+1}^{t_2} (1 - o_j^i) \cdot c_{t_2}^i \quad (9)$$

2.5 Losses

The GAP (Webster et al., 2018) training dataset is small and provides sparse supervision with labels for only two coreference links per instance. In order to compensate for this lack of supervision, we use a heuristic loss $\mathcal{L}_{ent}$ over entity mention probabilities in combination with the end task loss $\mathcal{L}_{coref}$ for coreference. The two losses are combined with a tunable hyperparameter λ resulting in the following total loss: $\mathcal{L} = \mathcal{L}_{coref} + \lambda \mathcal{L}_{ent}$.

2.5.1 Coreference Loss

The coreference loss is the binary cross entropy between the ground truth labels for mention pairs and the coreference link probability P_{CL} in Eq. (9). Eq. (9) expects a pair of tokens while the annotations are on pairs of spans, so we compute the loss for all ground truth token pairs: $\mathcal{L}_{coref} =$

$$\sum_{(s_a, s_b, y_{ab}) \in \text{G}} \left(\sum_{w_a \in s_a} \sum_{w_b \in s_b} H(y_{ab}, P_{\text{CL}}(w_a, w_b)) \right)$$

where G is the set of annotated span pairs and $H(p, q)$ represents the cross entropy of the distribution q relative to distribution p.

Apart from the ground truth labels, we use "implied labels" in the coreference loss calculation. For handling multi-token spans, we assume that all tokens following the head token are coreferential with the head token (self-links). We infer more supervision based on knowledge of the setup of the GAP task. Each GAP instance has two candidate names and a pronoun mention with supervision provided for the {name, pronoun} pairs. By design the two names are different, and therefore we use them as a negative coreference pair.

Even after the addition of this implied supervision, our coreference loss calculation is restricted to the three mention spans in each training instance; therefore, the running time is $\mathcal{O}(T)$ for finite-sized mention spans. In contrast, Liu et al. (2019a) compute the above coreference loss for all token pairs (assuming a negative label for all pairs outside of the mentions), which results in a runtime of $\mathcal{O}(T^3)$ due to the $\mathcal{O}(T^2)$ pairs and $\mathcal{O}(T)$ computation per pair, and thus will scale poorly to long documents.

2.5.2 Entity Mention Loss

We use the inductive bias that most tokens do not correspond to entities by imposing a loss on the average of the entity mention probabilities predicted across time steps, after masking out the labeled entity spans. For a training instance where spans s_A and s_B correspond to the person mentions and span s_P is a pronoun, the entity mention loss is

$$\mathcal{L}_{ent} = \frac{\sum_{t=1}^{T} e_t \cdot m_t}{\sum_{t=1}^{T} m_t}$$

where $m_t = 0$ if $w_t \in s_A \cup s_B \cup s_P$ and $m_t = 1$ otherwise.

Each GAP instance has only 3 labeled entity mention spans, but the text typically has other entity mentions that are not labeled. Unlabeled entity mentions will be inhibited by this loss. However, on average there are far more tokens outside entity spans than inside the spans. In experiments without

this loss, we observed that the model is susceptible to predicting a high entity probability for all tokens while still performing well on the end task of pronoun resolution. We are interested in tracking people beyond just the entities that are labeled in the GAP task, for which this loss is very helpful.

3 Experimental Setup

3.1 Data

GAP is a gender-balanced pronoun resolution dataset introduced by Webster et al. (2018). Each instance consists of a small snippet of text from Wikipedia, two spans corresponding to candidate names along with a pronoun span, and two binary labels indicating the coreference relationship between the pronoun and the two candidate names. Relative to other popular coreference datasets (Pradhan et al., 2012; Chen et al., 2018), GAP is comparatively small and sparsely annotated. We choose GAP because its small size allows us to do extensive experiments.

3.2 Model Details

For the input BERT embeddings, we concatenate either the last four layers of $\text{BERT}_{\text{BASE}}$, or layers 19–22 of $\text{BERT}_{\text{LARGE}}$ since those layers have been found to carry the most information related to coreference (Liu et al., 2019b). The BERT embeddings are fed to a 300-dimensional GRU model, which matches the dimensionality of the memory vectors.

We vary the number of memory cells N from 2 to 20. The decay rate for the memory usage scalar γ is 0.98. The MLPs used for predicting the entity probability and similarity score consist of two 300-dimensional ReLU hidden layers. For the *Fixed Key* variant of PeTra we use 20 dimensions for the learned key vector and the remaining 280 dimensions as the value vector.

3.3 Training

All models are trained for a maximum of 100 epochs with the Adam optimizer (Kingma and Ba, 2015). The learning rate is initialized to 10^{-3} and is reduced by half, until a minimum of 10^{-4}, whenever there is no improvement on the validation performance for the last 5 epochs. Training stops when there is no improvement in validation performance for the last 15 epochs. The temperature τ of the Gumbel-Softmax distribution used in the OVERWRITE operation is initialized to 1 and halved every 10 epochs. The coreference loss terms in Section 2.5.1 are weighted differently for different coreference links: (a) self-link losses for multi-token spans are given a weight of 1, (b) positive coreference link losses are weighted by 5, and (c) negative coreference link losses are multiplied by 50. To prevent overfitting: (a) we use early stopping based on validation performance, and (b) apply dropout at a rate of 0.5 on the output of the GRU model. Finally, we choose $\lambda = 0.1$ to weight the entity prediction loss described in Section 2.5.2.

3.4 People Tracking Evaluation

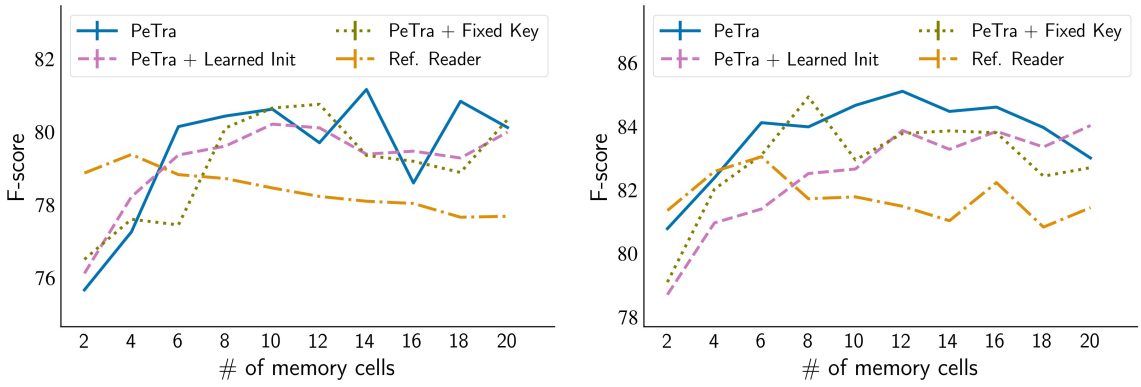
One of the goals of this work is to develop memory models that not only do well on the coreference resolution task, but also are interpretable in the sense that the memory cells actually track entities. Hence in addition to reporting the standard metrics on GAP, we consider two other ways to evaluate memory models.

As our first task, we propose an auxiliary entity-counting task. We take 100 examples from the GAP validation set and annotate them with the number of unique people mentioned in them.[4] We test the models by predicting the number of people from their memory logs as explained in Section 3.5. The motivation behind this exercise is that if a memory model is truly tracking entities, then its memory usage logs should allow us to recover this information.

To assess the people tracking performance more holistically, we conduct a human evaluation in which we ask annotators to assess the memory models on people tracking performance, defined as:(a) detecting references to people including pronouns, and (b) maintaining a 1-to-1 correspondence between people and memory cells. For this study, we pick the best run (among 5 runs) of PeTra and the Referential Reader for the 8-cell configuration using $\text{BERT}_{\text{BASE}}$ (PeTra: 81 F1; Referential Reader: 79 F1). Next we randomly pick 50 documents (without replacement) from the GAP dev set and split those into groups of 10 to get 5 evaluation sets. We shuffle the original 50 documents and follow the same steps to get another 5 evaluation sets. In the end, we have a total of 10 evaluation sets with 10 documents each, where each unique document belongs to exactly 2 evaluation sets.

We recruit 10 annotators for the 10 evaluation sets. The annotators are shown memory log visualizations as in Figure 5, and instructed to compare

[4] In the GAP dataset, the only relevant entities are people.



(a) BERT$_{\text{BASE}}$ (b) BERT$_{\text{LARGE}}$

Figure 3: Mean F1 score on the GAP validation set as a function of the number of memory cells.

the models on their people tracking performance (detailed instructions in Appendix A.3). For each document the annotators are presented memory logs of the two models (ordered randomly) and asked whether they prefer the first model, prefer the second model, or have no preference (neutral).

3.5 Inference

GAP Given a pronoun span s_P and two candidate name spans s_A & s_B, we have to predict binary labels for potential coreference links between (s_A, s_P) and (s_B, s_P). Thus, for a pair of entity spans, say s_A and s_P, we predict the coreference link probability as:

$$P_{\text{CL}}(s_A, s_P) = \max_{w_A \in s_A, w_P \in s_P} P_{\text{CL}}(w_A, w_P)$$

where $P_{\text{CL}}(w_A, w_P)$ is calculated using the procedure described in Section 2.4[5]. The final binary prediction is made by comparing the probability against a threshold.

Counting unique people For the test of unique people counting, we discretize the overwrite operation, which corresponds to new entities, against a threshold α and sum over all tokens and all memory cells to predict the count as follows:

$$\#\text{ unique people} = \sum_{t=1}^{T} \sum_{i=1}^{N} \mathbb{1}[o_t^i \geq \alpha]$$

3.6 Evaluation Metrics

For GAP we evaluate models using F-score.[6] First, we pick a threshold from the set $\{0.01, 0.02, \cdots, 1.00\}$ which maximizes the validation F-score. This threshold is then used to evaluate performance on the GAP test set.

For the interpretability task of counting unique people, we choose a threshold that minimizes the absolute difference between ground truth count and predicted count summed over the 100 annotated examples. We select the best threshold from the set $\{0.01, 0.02, \cdots, 1.00\}$. The metric is then the number of errors corresponding to the best threshold.[7]

3.7 Baselines

The Referential Reader (Liu et al., 2019a) is the most relevant baseline in the literature, and the most similar to PeTra. The numbers reported by Liu et al. (2019a) are obtained by a version of the model using BERT$_{\text{BASE}}$, with only two memory cells. To compare against PeTra for other configurations, we retrain the Referential Reader using the code made available by the authors.[8]

We also report the results of Joshi et al. (2019) and Wu et al. (2019), although these numbers are not comparable since both of them train on the much larger OntoNotes corpus and just test on GAP.

4 Results

4.1 GAP results

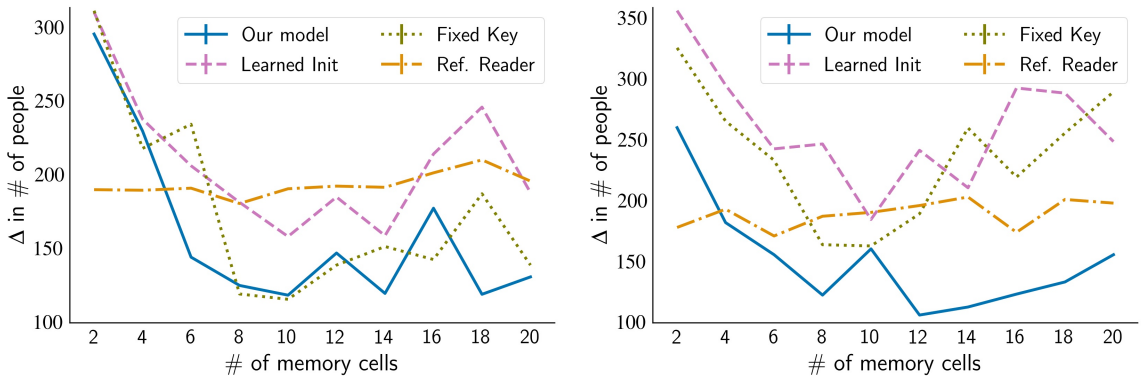
We train all the memory models, including the Referential Reader, with memory size varying from $\{2, 4, \cdots, 20\}$ memory cells for both BERT$_{\text{BASE}}$ and BERT$_{\text{LARGE}}$, with each configuration being trained 5 times. Figure 3 shows the performance of the

[5]The computation of this probability includes the mention detection steps required byWebster et al. (2018).

[6]GAP also includes evaluation related to gender bias, but this is not a focus of this paper so we do not report it.

[7]Note that the error we report is therefore a best-case result. We are not proposing a way of counting unique people in new test data, but rather using this task for analysis.

[8]https://github.com/liufly/refreader



(a) $\text{BERT}_{\text{BASE}}$ (b) $\text{BERT}_{\text{LARGE}}$

Figure 4: Error in counting unique people as a function of number of memory cells; lower is better.

	$\text{BERT}_{\text{BASE}}$	$\text{BERT}_{\text{LARGE}}$
PeTra	**81.5 ± 0.6**	**85.3 ± 0.6**
+ Learned Init.	80.9 ± 0.7	84.4 ± 1.2
+ Fixed Key	81.1 ± 0.7	85.1 ± 0.8
Ref. Reader	78.9 ± 1.3	83.7 ± 0.8
Ref. Reader (2019a)	78.8	-
Joshi et al. (2019)	82.8	85.0
Wu et al. (2019)	-	87.5 (SpanBERT)

Table 1: Results (%F1) on the GAP test set.

models on the GAP validation set as a function of memory size. The Referential Reader outperforms PeTra (and its memory variants) when using a small number of memory cells, but its performance starts degrading after 4 and 6 memory cells for $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$ respectively. PeTra and its memory variants, in contrast, keep improving with increased memory size (before saturation at a higher number of cells) and outperform the best Referential Reader performance for all memory sizes ≥ 6 cells. With larger numbers of memory cells, we see a higher variance, but the curves for PeTra and its memory variants are still consistently higher than those of the Referential Reader.

Among different memory variants of PeTra, when using $\text{BERT}_{\text{BASE}}$ the performances are comparable with no clear advantage for any particular choice. For $\text{BERT}_{\text{LARGE}}$, however, vanilla PeTra has a clear edge for almost all memory sizes, suggesting the limited utility of initialization. The results show that PeTra works well without learning vectors for initializing the key or memory cell contents. Rather, we can remove the key/value distinction and simply initialize all memory cells with the zero vector.

To evaluate on the GAP test set, we pick the memory size corresponding to the best validation performance for all memory models. Table 1 shows that the trends from validation hold true for test as well, with PeTra outperforming the Referential Reader and the other memory variants of PeTra.

4.2 Counting unique people

Figure 4 shows the results for the proposed interpretability task of counting unique people. For both $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$, PeTra achieves the lowest error count. Interestingly, from Figure 4b we can see that for ≥ 14 memory cells, the other memory variants of PeTra perform worse than the Referential Reader while being better at the GAP validation task (see Figure 3b). This shows that a better performing model is not necessarily better at tracking people.

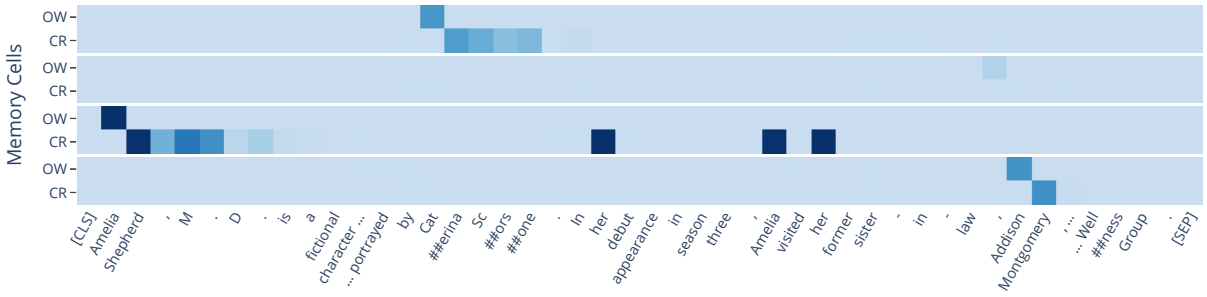
	$\text{BERT}_{\text{BASE}}$	$\text{BERT}_{\text{LARGE}}$
PeTra	**0.76**	**0.69**
+ Learned Init	0.72	0.60
+ Fixed Key	0.72	0.65
Ref. Reader	0.49	0.54

Table 2: Spearman's correlation between GAP validation F1 and negative error count for unique people.

To test the relationship between the GAP task and the proposed interpretability task, we compute the correlation between the GAP F-score and the negative count of unique people for each model separately.[9] Table 2 shows the Spearman's correlation between these measures. For all models we see a positive correlation, indicating that a dip in coreference performance corresponds to an increase in error on counting unique people. The correlations for PeTra are especially high, again suggesting it's greater interpretability.

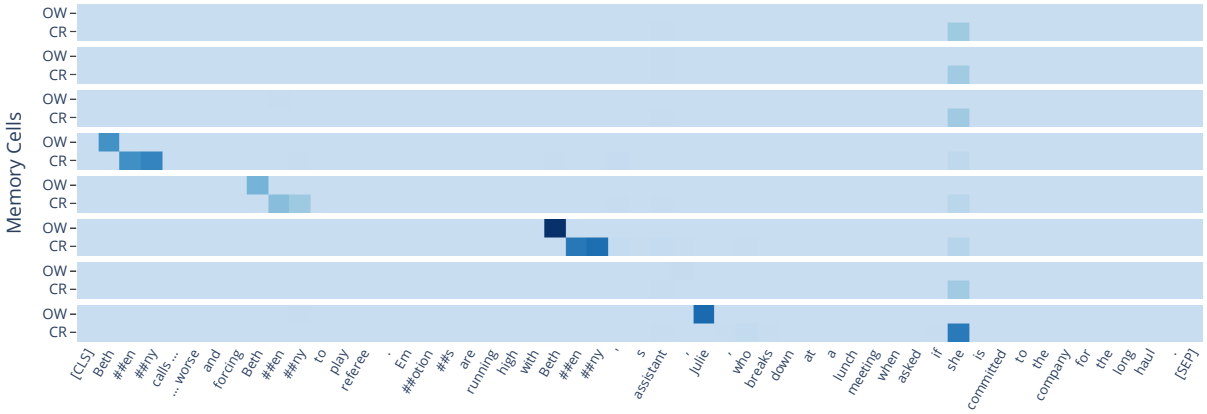
[9]Each correlation is computed over 50 runs (5 runs each for 10 memory sizes).

Amelia Shepherd$_1$, M.D. is a fictional character on the ABC American television medical drama Private Practice, and the spinoff series' progenitor show, Grey's Anatomy, portrayed by *Caterina Scorsone*$_2$. In **her**$_1$ debut appearance in season three, **Amelia**$_1$ visited her former sister-in-law, Addison Montgomery$_3$, and became a partner at the Oceanside Wellness Group.



(a) A successful run of PeTra with 4 memory cells. The model accurately links all the mentions of "Amelia" to the same memory cell while also detecting other people in the discourse.

Bethenny$_1$ calls a meeting to get everyone on the same page, but Jason$_2$ is hostile with the group, making things worse and forcing Bethenny$_1$ to play referee. Emotions are running high with *Bethenny*$_1$'s assistant, **Julie**$_3$, who breaks down at a lunch meeting when asked if **she**$_3$ is committed to the company for the long haul.



(b) Memory log of PeTra with 8 memory cells. The model correctly links "she" and "Julie" but fails at linking the three "Bethenny" mentions, and also fails at detecting "Jason".

Figure 5: Visualization of memory logs for different configurations of PeTra. The documents have their GAP annotations highlighted in red (italics) and blue (bold), with blue (bold) corresponding to the right answer. For illustration purposes only, we highlight all the spans corresponding to mentions of people and mark cluster indices as subscript. In the plot, X-axis corresponds to document tokens, and Y-axis corresponds to memory cells. Each memory cell has the OW=OVERWRITE and CR=COREF labels. Darker color implies higher value. We skip text, indicated via ellipsis, when the model doesn't detect people for extended lengths of text.

4.3 Human Evaluation for People Tracking

Model	Preference (in %)
PeTra	**74**
Ref. Reader	08
Neutral	18

Table 3: Human Evaluation results for people tracking.

Table 3 summarizes the results of the human evaluation for people tracking. The annotators prefer PeTra in 74% cases while the Referential Reader for only 8% instances (see Appendix A.4 for visualizations comparing the two). Thus, PeTra easily outperforms the Referential Reader on this task even though they are quite close on the GAP evaluation. The annotators agree on 68% of the documents, disagree between PeTra and Neutral for 24% of the documents, and disagree between PeTra and the Referential Reader for the remaining 8% documents. For more details, see Appendix A.2.

4.4 Model Runs

We visualize two runs of PeTra with different configurations in Figure 5. For both instances the model gets the right pronoun resolution, but clearly in Figure 5b the model fails at correctly tracking repeated mentions of "Bethenny". We believe these errors happen because (a) GAP supervision is limited to pronoun-proper name pairs, so the model is never explicitly supervised to link proper names, and (b) there is a lack of span-level features, which hurts the model when a name is split across multiple tokens.

5 Related Work

There are several strands of related work, including prior work in developing neural models with external memory as well as variants that focus on modeling entities and entity relations, and neural models for coreference resolution.

Memory-augmented models. Neural network architectures with external memory include memory networks (Weston et al., 2015; Sukhbaatar et al., 2015), neural Turing machines (Graves et al., 2014), and differentiable neural computers (Graves et al., 2016). This paper focuses on models with inductive biases that produce particular structures in the memory, specifically those related to entities.

Models for tracking and relating entities. A number of existing models have targeted entity tracking and coreference links for a variety of tasks. EntNet (Henaff et al., 2017) aims to track entities via a memory model. EntityNLM (Ji et al., 2017) represents entities dynamically within a neural language model. Hoang et al. (2018) augment a reading comprehension model to track entities, incorporating a set of auxiliary losses to encourage capturing of reference relations in the text. Dhingra et al. (2018) introduce a modified GRU layer designed to aggregate information across coreferent mentions.

Memory models for NLP tasks. Memory models have been applied to several other NLP tasks in addition to coreference resolution, including targeted aspect-based sentiment analysis (Liu et al., 2018b), machine translation (Maruf and Haffari, 2018), narrative modeling (Liu et al., 2018a), and dialog state tracking (Perez and Liu, 2017). Our study of architectural choices for memory may also be relevant to models for these tasks.

Neural models for coreference resolution. Several neural models have been developed for coreference resolution, most of them focused on modeling pairwise interactions among mentions or spans in a document (Wiseman et al., 2015; Clark and Manning, 2016a; Lee et al., 2017, 2018). These models use heuristics to avoid computing scores for all possible span pairs in a document, an operation which is quadratic in the document length T assuming a maximum span length. Memory models for coreference resolution, including our model, differ by seeking to store information about entities in memory cells and then modeling the relationship between a token and a memory cell. This reduces computation from $\mathcal{O}(T^2)$ to $\mathcal{O}(TN)$, where N is the number of memory cells, allowing memory models to be applied to longer texts by using the global entity information. Past work (Wiseman et al., 2016) have used global features, but in conjunction with other features to score span pairs.

Referential Reader. Most closely related to the present work is the Referential Reader (Liu et al., 2019a), which uses a memory model to perform coreference resolution incrementally. We significantly simplify this model to accomplish the same goal with far fewer parameters.

6 Conclusion and Future Work

We propose a new memory model for entity tracking, which is trained using sparse coreference resolution supervision. The proposed model outperforms a previous approach with far fewer parameters and a simpler architecture. We propose a new diagnostic evaluation and conduct a human evaluation to test the interpretability of the model, and find that our model again does better on this evaluation. In future work, we plan to extend this work to longer documents such as the recently released dataset of Bamman et al. (2019).

Acknowledgments

This material is based upon work supported by the National Science Foundation under Award Nos. 1941178 and 1941160. We thank the ACL reviewers, Sam Wiseman, and Mrinmaya Sachan for their valuable feedback. We thank Fei Liu and Jacob Eisenstein for answering questions regarding the Referential Reader. Finally, we want to thank all the annotators at TTIC who participated in the human evaluation study.

References

David Bamman, Olivia Lewke, and Anya Mansoor. 2019. An Annotated Dataset of Coreference in English Literature. *arXiv preprint arXiv:1912.01140*.

Hong Chen, Zhenhua Fan, Hao Lu, Alan Yuille, and Shu Rong. 2018. PreCo: A Large-scale Dataset in Preschool Vocabulary for Coreference Resolution. In *EMNLP*.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *EMNLP*.

Kevin Clark and Christopher D. Manning. 2016a. Deep Reinforcement Learning for Mention-Ranking Coreference Models. In *EMNLP*.

Kevin Clark and Christopher D Manning. 2016b. Improving Coreference Resolution by Learning Entity-Level Distributed Representations. In *ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.

Bhuwan Dhingra, Qiao Jin, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2018. Neural Models for Reasoning over Multiple Mentions Using Coreference. In *NAACL*.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural Turing machines. *arXiv preprint arXiv:1410.5401*.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, AdriàPuigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2017. Tracking the world state with recurrent entity networks. In *ICLR*.

Luong Hoang, Sam Wiseman, and Alexander Rush. 2018. Entity Tracking Improves Cloze-style Reading Comprehension. In *EMNLP*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *ICLR*.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A. Smith. 2017. Dynamic Entity Representations in Neural Language Models. In *EMNLP*.

Mandar Joshi, Omer Levy, Luke Zettlemoyer, and Daniel Weld. 2019. BERT for Coreference Resolution: Baselines and Analysis. In *EMNLP*.

Frank Keller. 2010. Cognitively Plausible Models of Human Language Processing. In *ACL*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end Neural Coreference Resolution. In *EMNLP*.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-Order Coreference Resolution with Coarse-to-Fine Inference. In *NAACL-HLT*.

Fei Liu, Trevor Cohn, and Timothy Baldwin. 2018a. Narrative Modeling with Memory Chains and Semantic Supervision. In *ACL*.

Fei Liu, Trevor Cohn, and Timothy Baldwin. 2018b. Recurrent Entity Networks with Delayed Memory Update for Targeted Aspect-Based Sentiment Analysis. In *NAACL-HLT*.

Fei Liu, Luke Zettlemoyer, and Jacob Eisenstein. 2019a. The Referential Reader: A recurrent entity network for anaphora resolution. In *ACL*.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019b. Linguistic Knowledge and Transferability of Contextual Representations. In *NAACL-HLT*.

Sameen Maruf and Gholamreza Haffari. 2018. Document Context Neural Machine Translation with Memory Networks. In *ACL*.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-Value Memory Networks for Directly Reading Documents. In *EMNLP*.

Julien Perez and Fei Liu. 2017. Dialog state tracking, a machine reading approach using Memory Network. In *EACL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 Shared Task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, CoNLL '12.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *NeurIPS*.

MK Tanenhaus, MJ Spivey-Knowlton, KM Eberhard, and JC Sedivy. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268(5217).

Kellie Webster, Marta Recasens, Vera Axelrod, and Jason Baldridge. 2018. Mind the GAP: A Balanced Corpus of Gendered Ambiguous Pronouns. In *TACL*, volume 6.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory Networks. In *ICLR*.

Sam Wiseman, Alexander M. Rush, Stuart Shieber, and Jason Weston. 2015. Learning Anaphoricity and Antecedent Ranking Features for Coreference Resolution. In *ACL-IJCNLP*.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning Global Features for Coreference Resolution. In *NAACL*.

Wei Wu, Fei Wang, Arianna Yuan, Fei Wu, and Jiwei Li. 2019. Coreference Resolution as Query-based Span Prediction. *arXiv preprint arXiv:1911.01746*.

A Appendix

A.1 Best Runs vs. Worst Runs

As Table 1 shows, there is significant variance in the performance of these memory models. To analyze how the best runs diverge from the worst runs, we analyze how the controller network is using the different memory cells in terms of overwrites. For this analysis, we choose the best and worst among the 5 runs for each configuration, as determined by GAP validation performance. For the selected runs, we calculate the KL-divergence of the average overwrite probability distribution from the uniform distribution and average it for each model type. Table 4 shows that for the memory variants *Learned Init* and *Fixed Key*, the worst runs overwrite more to some memory cells than others (high average KL-divergence). Note that both PeTra and Referential Reader are by design intended to have no preference for any particular memory cell (which the numbers support), hence the low KL-divergence.

	Avg KL-div	
	Best run	Worst run
PeTra	0.00	0.01
+ Learned Init.	0.3	**0.83**
+ Fixed Key	0.2	**0.8**
Ref. Reader	0.05	0.04

Table 4: A comparison of best runs vs. worst runs.

A.2 Human Evaluation Results

The agreement matrix for the human evaluation study described in Section 4.3 is shown in Figure 6. This agreement matrix is a result of the two annotations per document that we get as per the setup described in Section 3.4. Note that the annotations are coming from two sets of annotators rather than two individual annotators. This is also the reason why we don't report standard inter-annotator agreement coefficients.

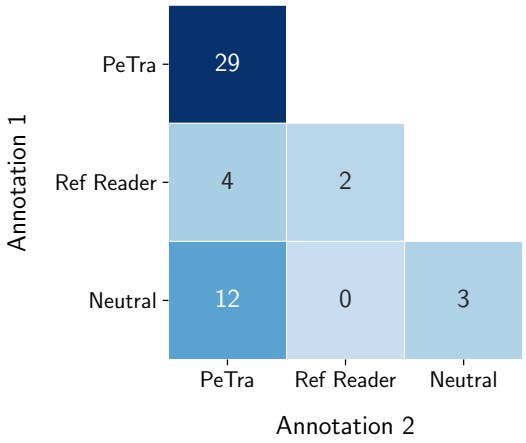


Figure 6: Agreement matrix for human evaluation study.

A.3 Instructions for Human Evaluation

The detailed instructions for the human evaluation study described in Section 4.3 are shown in Figure 7. We simplified certain memory model specific terms such as "overwrite" to "new person" since the study was really about people tracking.

A.4 Comparative visualization of memory logs of PeTra and the Referential Reader

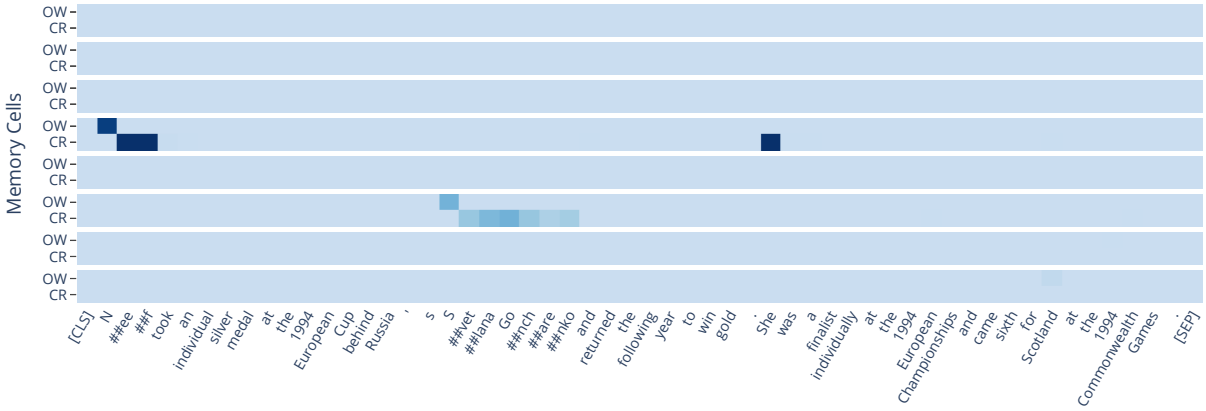
Figure 8 and 9 compare the memory logs of PeTra and the Referential Reader.

- In this user study we will be comparing memory models at tracking people.
- What are memory models? Memory models are neural networks coupled with an external memory which can be used for reading/writing.
- (IMPORTANT) What does it mean to track people for memory models?
  - Detect all references to people which includes pronouns.
  - A 1-to-1 correspondence between people and memory cells i.e. all references corresponding to a person should be associated with the same memory cell AND each memory cell should be associated with at most 1 person.
- The memory models use the following scores (which are visualized) to indicate the tracking decisions:
  - New Person Probability (Cell i): Probability that the token refers to a new person (not introduced in the text till now) and we start tracking it in cell i.
  - Coreference Probability (Cell i): Probability that the token refers to a person already being tracked in cell i.
- The objective of this study is to compare the models on the interpretability of their memory logs i.e. are the models actually tracking entities or not. You can choose how you weigh the different requirements for tracking people (from 3).
- For this study, you will compare two memory models with 8 memory cells (represented via 8 rows). The models are ordered randomly for each instance.
- For each document, you can choose model A or model B, or stay neutral in case both the models perform similarly.

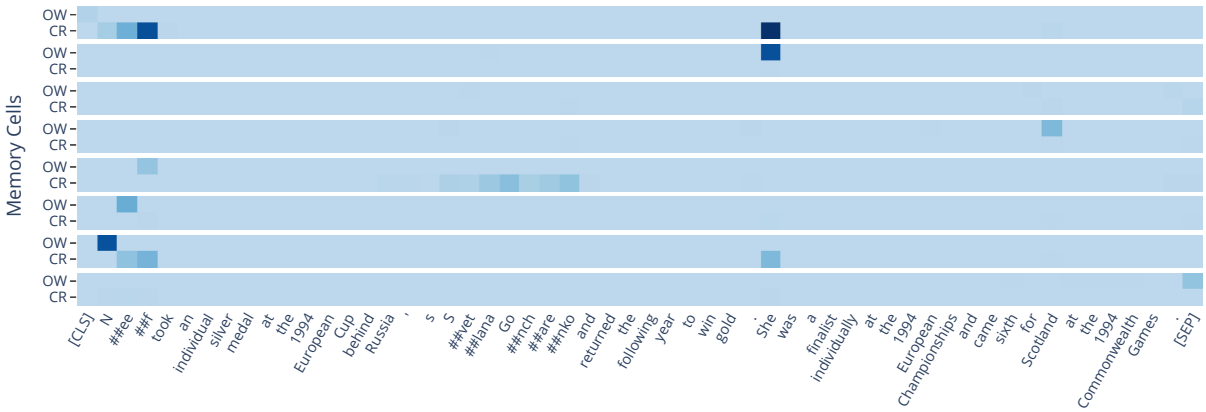
Figure 7: Instructions for the human evaluation study.

Neef$_1$ took an individual silver medal at the 1994 European Cup behind Russia's *Svetlana Goncharenko*$_2$ and returned the following year to win gold. **She**$_1$ was a finalist individually at the 1994 European Championships and came sixth for Scotland at the 1994 Commonwealth Games.

(a) GAP validation instance 293. The ground truth GAP annotation is indicated via colors.



(b) Memory log of PeTra with 8 memory cells. PeTra uses only 2 memory cells for the 2 unique people, namely Neef and Svetlana Goncharenko, and correctly resolves the pronoun.

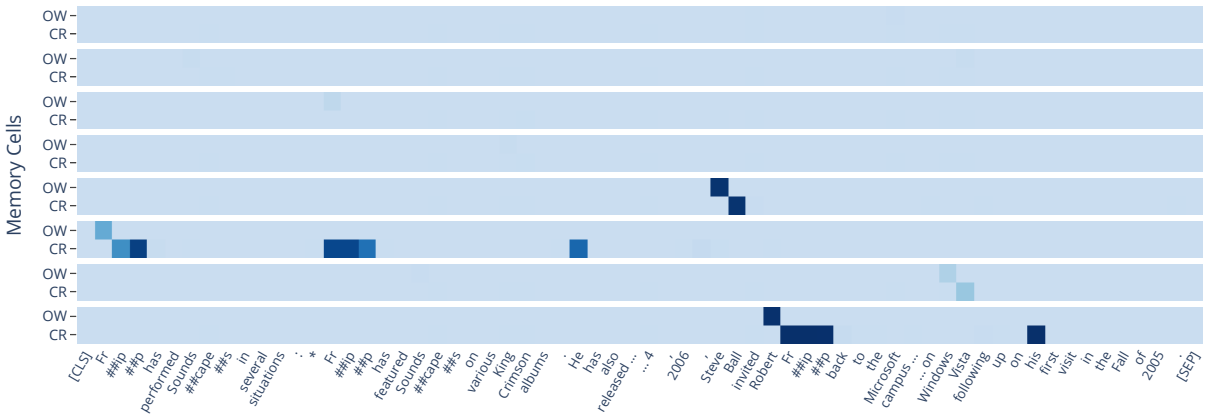


(c) Memory log of the Referential Reader with 8-memory cells. The Referential Reader does successfully resolve the pronoun in the topmost memory cell but it ends up tracking Neef in as many as 4 memory cells.

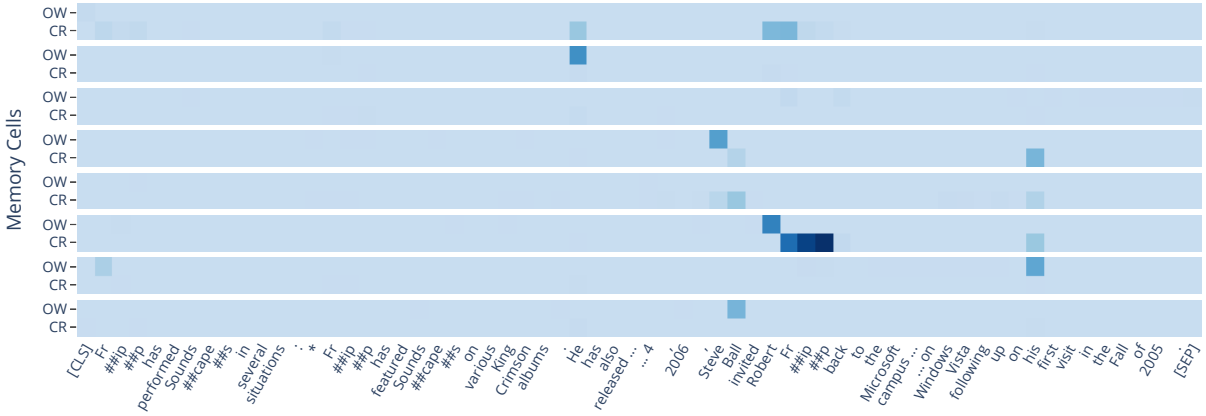
Figure 8: Both the models only weakly detect "Svetlana Goncharenko" which could be due to lack of span modeling.

Fripp$_1$ has performed Soundscapes in several situations: * Fripp$_1$ has featured Soundscapes on various King Crimson albums. He$_1$ has also released pure Soundscape recordings as well: * On May 4, 2006, *Steve Ball*$_2$ invited **Robert Fripp**$_1$ back to the Microsoft campus for a second full day of work on Windows Vista following up on **his**$_1$ first visit in the Fall of 2005.

(a) GAP validation instance 17. The ground truth GAP annotation is indicated via colors.



(b) Memory log of PeTra with 8-memory cells. PeTra is pretty accurate at tracking Robert Fripp but it misses out on connecting "Fripp" from the earlier part of the document to "Robert Fripp".



(c) Memory log of the Referential Reader with 8-memory cells. The Referential Reader completely misses out on all the mentions in the first half of the document (which is not penalized in GAP evaluations where the relevant annotations are typically towards the end of the document). Apart from this, the model ends up tracking Robert Fripp in as many as 6 memory cells, and Steve Ball in 3 memory cells.

Figure 9: PeTra clearly performs better than the Referential Reader at people tracking for this instance. PeTra's output is more sparse, detects more relevant mentions, and is better at maintaining a 1-to-1 correspondence between memory cells and people.