Learning Personas from Dialogue with Attentive Memory Networks

Eric Chu [*]
MIT Media Lab
echu@mit.edu

Prashanth Vijayaraghavan [*]
MIT Media Lab
pralav@mit.edu

Deb Roy
MIT Media Lab
dkroy@media.mit.edu

Abstract

The ability to infer persona from dialogue can have applications in areas ranging from computational narrative analysis to personalized dialogue generation. We introduce neural models to learn persona embeddings in a supervised character trope classification task. The models encode dialogue snippets from IMDB into representations that can capture the various categories of film characters. The best-performing models use a multi-level attention mechanism over a set of utterances. We also utilize prior knowledge in the form of textual descriptions of the different tropes. We apply the learned embeddings to find similar characters across different movies, and cluster movies according to the distribution of the embeddings. The use of short conversational text as input, and the ability to learn from prior knowledge using memory, suggests these methods could be applied to other domains.

1 Introduction

Individual personality plays a deep and pervasive role in shaping social life. Research indicates that it can relate to the professional and personal relationships we develop (Barrick and Mount, 1993), (Shaver and Brennan, 1992), the technological interfaces we prefer (Nass and Lee, 2000), the behavior we exhibit on social media networks (Selfhout et al., 2010), and the political stances we take (Jost et al., 2009).

With increasing advances in human-machine dialogue systems, and widespread use of social media in which people express themselves via short text messages, there is growing interest in systems that have an ability to understand different personality types. Automated personality analysis based on short text analysis could open up a range of potential applications, such as dialogue agents that

sense personality in order to generate more interesting and varied conversations.

We define persona as a person's social role, which can be categorized according to their conversations, beliefs, and actions. To learn personas, we start with the character tropes data provided in the CMU Movie Summary Corpus by (Bamman et al., 2014). It consists of 72 manually identified commonly occurring character archetypes and examples of each. In the character trope classification task, we predict the character trope based on a batch of dialogue snippets.

In their original work, the authors use Wikipedia plot summaries to learn latent variable models that provide a clustering from words to topics and topics to personas – their persona clusterings were then evaluated by measuring similarity to the ground-truth character trope clusters. We asked the question – could personas also be inferred through dialogue? Because we use quotes as a primary input and not plot summaries, we believe our model is extensible to areas such as dialogue generation and conversational analysis.

Our contributions are:

1. Data collection of IMDB quotes and character trope descriptions for characters from the CMU Movie Summary Corpus.

2. Models that greatly outperform the baseline model in the character trope classification task. Our experiments show the importance of multi-level attention over words in dialogue, and over a set of dialogue snippets.

3. We also examine how prior knowledge in the form of textual descriptions of the persona categories may be used. We find that a 'Knowledge-Store' memory initialized with descriptions of the tropes is particularly useful. This ability may allow these models to be used more flexibly in new domains and with

---

[*] The first two authors contributed equally to this work.

Character Trope	Character	Movie
Corrupt corporate executive	Les Grossman	Tropic Thunder
Retired outlaw	Butch Cassidy	Butch Cassidy and the Sundance Kid
Lovable rogue	Wolverine	X-Men

Table 1: Example tropes and characters

different persona categories.

2 Related Work

Prior to data-driven approaches, personalities were largely measured by asking people questions and assigning traits according to some fixed set of dimensions, such as the Big Five traits of openness, conscientiousness, extraversion, agreeability, and neuroticism (Tupes and Christal, 1992). Computational approaches have since advanced to infer these personalities based on observable behaviors such as the actions people take and the language they use (Golbeck et al., 2011).

Our work builds on recent advances in neural networks that have been used for natural language processing tasks such as reading comprehension (Sukhbaatar et al., 2015) and dialogue modeling and generation (Vinyals and Le, 2015; Li et al., 2016; Shang et al., 2015). This includes the growing literature in attention mechanisms and memory networks (Bahdanau et al., 2014; Sukhbaatar et al., 2015; Kumar et al., 2016).

The ability to infer and model personality has applications in storytelling agents, dialogue systems, and psychometric analysis. In particular, personality-infused agents can help "chit-chat" bots avoid repetitive and uninteresting utterances (Walker et al., 1997; Mairesse and Walker, 2007; Li et al., 2016; Zhang et al., 2018). The more recent neural models do so by conditioning on a 'persona' embedding – our model could help produce those embeddings.

Finally, in the field of literary analysis, graphical models have been proposed for learning character personas in novels (Flekova and Gurevych, 2015; Srivastava et al., 2016), folktales (Valls-Vargas et al., 2014), and movies (Bamman et al., 2014). However, these models often use more structured inputs than dialogue to learn personas.

3 Datasets

Characters in movies can often be categorized into archetypal roles and personalities. To understand the relationship between dialogue and personas,

we utilized three different datasets for our models: (a) the Movie Character Trope dataset, (b) the IMDB Dialogue Dataset, and (c) the Character Trope Description Dataset. We collected the IMDB Dialogue and Trope Description datasets, and these datasets are made publicly available [1].

3.1 Character Tropes Dataset

The CMU Movie Summary dataset provides tropes commonly occurring in stories and media (Bamman et al., 2014). There are a total of 72 tropes, which span 433 characters and 384 movies. Each trope contains between 1 and 25 characters, with a median of 6 characters per trope. Tropes and canonical examples are shown in Table 1.

3.2 IMDB Dialogue Snippet Dataset

To obtain the utterances spoken by the characters, we crawled the IMDB Quotes page for each movie. Though not every single utterance spoken by the character may be available, as the quotes are submitted by IMDB users, many quotes from most of the characters are typically found, especially for the famous characters found in the Character Tropes dataset. The distribution of quotes per trope is displayed in Figure 1. Our models were trained on 13,874 quotes and validated and tested on a set of 1,734 quotes each.

We refer to each IMDB quote as a (contextualized) dialogue snippet, as each quote can contain several lines between multiple characters, as well as italicized text giving context to what might be happening when the quote took place. Figure 2 show a typical dialogue snippet. 70.3% of the quotes are multi-turn exchanges, with a mean of 3.34 turns per multi-turn exchange. While the character's own lines alone can be highly indicative of the trope, our models show that accounting for context and the other characters' lines and context improves performance. The context, for instance, can give clues to typical scenes and actions that are associated with certain tropes, while the other characters' lines give further detail into

---

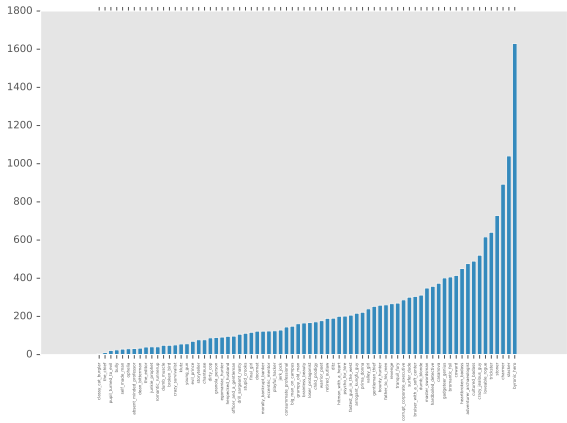[1] https://pralav.github.io/emnlp_personas/

2639

Figure 1: Number of IMDB dialogue snippets per trope

Stacks Edwards: What time is it?
Tommy DeVito: It's eleven thirty, we're supposed to be there by nine.
Stacks Edwards: Be ready in a minute.
Tommy DeVito: Yeah, you were always fuckin' late, you were late for your own fuckin' funeral.
[*shoots him*]

Figure 2: Example IMDB dialogue snippet containing multiple characters and context.

the relationship between the character and his or her environment.

3.3 Character Trope Description Dataset

We also incorporate descriptions of each of the character tropes by using the corresponding descriptions scraped from TVTropes[2]. Each description contains several paragraphs describing typical characteristics, actions, personalities, etc. about the trope. As we demonstrate in our experiments, the use of these descriptions improves classification performance. This could allow our model to be applied more flexibly beyond the movie character tropes – as one example, we could store descriptions of personalities based on the Big Five traits in our Knowledge-Store memory.

4 Problem Formulation

Our goal is to train a model that can take a batch of dialogue snippets from the IMDB dataset and predict the character trope.

Formally, let N_P be the total number of character tropes in the character tropes dataset. Each character C is associated with a corresponding ground-truth trope category P. Let $S = (D, E, O)$ be a dialog snippet associated with a character C, where $D = [w_{D_1}, w_{D_2}..., w_{D_T}]$ refers to the character's own lines, $E =$

---

[2]http://tvtropes.org

$[w_{E_1}, w_{E_2}..., w_{E_T}]$ is the contextual information and $O = [w_{O_1}, w_{O_2}..., w_{O_T}]$ denotes the other characters' lines. We define all three components of S to have fixed sequence length T and pad when necessary. Let N_S be the total number of dialogue snippets for a trope. We sample a set of N_{diag} (where $N_{diag} \ll N_S$) snippets from N_S snippets related to the trope as inputs to our model.

5 Attentive Memory Network

The Attentive Memory Network consists of two major components: (a) Attentive Encoders, and (b) a Knowledge-Store Memory Module. Figure 3 outlines the overall model. We describe the components in the following sections.

5.1 Attentive Encoders

Not every piece of dialogue may be reflective of a latent persona. In order to learn to ignore words and dialogue snippets that are not informative about the trope we use a multi-level attentive encoder that operates at (a) the individual snippet level, and (b) across multiple snippets.

Attentive Snippet Encoder

The snippet encoder extracts features from a single dialogue snippet S, with attention over the words in the snippet. A snippet $S = (D, E, O)$ is fed to the encoder to extract features from each of these textual inputs and encode them into an embedding space. We use a recurrent neural network as our encoder, explained in detail in Section 5.1.1. In order to capture the trope-reflective words from the input text, we augment our model with a self-attention layer which scores each word in the given text for its relevance. Section 5.1.2 explains how the attention weights are computed. The output of this encoder is an encoded snippet embedding $S^e = (D^e, E^e, O^e)$.

Attentive Inter-Snippet Encoder

As shown in Figure 3, the N_{diag} snippet embeddings S^e from the snippet encoder are fed to our inter-snippet encoder. This encoder captures inter-snippet relationship using recurrence over the N_{diag} snippet embeddings for a given trope and determines their importance. Some of the dialogue snippets may not be informative about the trope, and the model learns to assign low attention scores to such snippets. The resulting attended
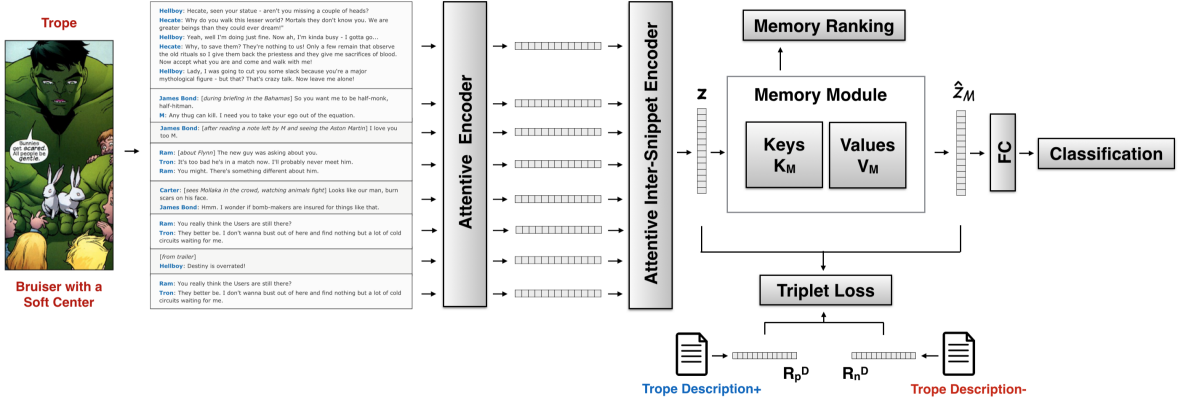
Figure 3: Illustration of the Attentive Memory Network. The network takes dialogue snippets as input and predicts its associated character trope. In this example, dialogue snippets associated with the character trope "Bruiser with a Soft Corner" is given as input to the model.

summary vector from this phase is the persona representation z, defined as:

$$z = \gamma_D^s D^s + \gamma_E^s E^s + \gamma_O^s O^s \qquad (1)$$
$$\gamma_D^s + \gamma_E^s + \gamma_O^s = 1$$

where $\gamma_D^s, \gamma_E^s, \gamma_O^s$ are learnable weight parameters. D^s, E^s, O^s refers to summary vectors of the N_{diag} character's lines, contextual information, and other characters' lines, respectively. In Section 7, we experiment with models that have γ_E^s and γ_O^s set to 0 to understand how the contextual information and other characters' lines contribute to the overall performance.

5.1.1 Encoder

Given an input sequence $(x_1, x_2, ..., x_T)$, we use a recurrent neural network to encode the sequence into hidden states $(h_1, h_2, ..., h_T)$. In our experiments, we use a gated recurrent network (GRU) (Chung et al., 2014) over LSTMs (Hochreiter and Schmidhuber, 1997) because the latter is more computationally expensive. We use bidirectional GRUs and concatenate our forward and backwards hidden states to get $\overleftrightarrow{h_t}$ for $t = 1, ..., T$.

5.1.2 Attention

We define an attention mechanism $Attn$ that computes s from the resultant hidden states $\overrightarrow{h_t}$ of a GRU by learning to generate weights α_t. This can be interpreted as the relative importance given to a hidden state h_t to form an overall summary vector for the sequence. Formally, we define it as:

$$a_t = f_{attn}(h_t) \qquad (2)$$
$$\alpha_t = softmax(a_t) \qquad (3)$$
$$s = \sum_{t=1}^{T} \alpha_t h_t \qquad (4)$$

where f_{attn} is a two layer fully connected network in which the first layer projects $h_t \in \mathbb{R}^{d_h}$ to an attention hidden space $g_t \in \mathbb{R}^{d_a}$, and the second layer produces a relevance score for every hidden state at timestep t.

5.2 Memory Modules

Our model consists of a read-only 'Knowledge-Store' memory, and we also test a recent read-write memory. External memories have been shown to help on natural language processing tasks (Sukhbaatar et al., 2015; Kumar et al., 2016; Kaiser and Nachum, 2017), and we find similar improvements in learning capability.

5.2.1 Knowledge-Store Memory

The main motivation behind the Knowledge-Store memory module is to incorporate prior domain knowledge. In our work, this knowledge refers to the trope descriptions described in Section 3.3.

Related works have initialized their memory networks with positional encoding using word embeddings (Sukhbaatar et al., 2015; Kumar et al., 2016; Miller et al., 2016). To incorporate the descriptions, we represent them with skip thought vectors (Kiros et al., 2015) and use them to initialize the memory keys $K_M \in \mathbb{R}^{N_P \times d_K}$, where N_P is the number of tropes, and d_K is set to the size of embedded trope description R^D, i.e. $d_K = ||R^D||$.

The values in the memory represent learnable embeddings of corresponding trope categories $V_M \in \mathbb{R}^{N_P \times d_V}$, where d_V is the size of the trope category embeddings. The network learns to use the persona representation z from the encoder phase to find relevant matches in the memory. This

2641

corresponds to calculating similarities between z and the keys K_M. Formally, this is calculated as:

$$z_M = f_z(z) \tag{5}$$

$$p_M^{(i)} = softmax(z_M \cdot K_M[i]) \tag{6}$$
$$\forall i \in \{1, .., N_P\}$$

where $f_z : \mathbb{R}^{d_h} \mapsto \mathbb{R}^{d_K}$ is a fully-connected layer that projects the persona representation in the space of memory keys K_M. Based on the match probabilities $p_M^{(i)}$, the values V_M are weighted and cumulatively added to the original persona representation as:

$$r_{out} = \sum_{i=1}^{N_P} p_M^{(i)} \cdot V_M[i] \tag{7}$$

We iteratively combine our mapped persona representation z_M with information from the memory r_{out}. The above process is repeated n_{hop} times. The memory mapped persona representation z_M is updated as follows:

$$z_M^{hop} = f_r(z_M^{hop-1}) + r_{out} \tag{8}$$

where $z_M^0 = z_M$, and $f_r : \mathbb{R}^{d_V} \mapsto \mathbb{R}^{d_K}$ is a fully-connected layer. Finally, we transform the resulting $z_M^{n_{hop}}$ using another fully-connected layer, $f_{out} \in \mathbb{R}^{d_K} \mapsto \mathbb{R}^{d_h}$, via:

$$\hat{z}_M = f_{out}(z_M^{n_{hop}}) \tag{9}$$

5.2.2 Read-Write Memory

We also tested a Read-Write Memory following Kaiser et. al (Kaiser and Nachum, 2017), which was originally designed to remember rare events. In our case, these 'rare' events might be key dialogue snippets that are particularly indicative of latent persona. It consists of keys, which are activations of a specific layer of model, i.e. the persona representation z, and values, which are the ground-truth labels, i.e. the trope categories. Over time, it is able to facilitate predictions based on past data with similar activations stored in the memory. For every new example, the network writes to memory for future look up. A memory with memory size N_M is defined as:

$$M = (K_{N_M \times d_H}, V_{N_M}, A_{N_M}) \tag{10}$$

Memory Read   We use the persona embedding z as a query to the memory. We calculate the cosine similarities between z and the keys in M, take the softmax on the top-k neighbors, and compute a weighted embedding $\hat{z}_M$ using those scores.

Memory Write   We update the memory in a similar fashion to the original work by (Kaiser and Nachum, 2017), which takes into account the maximum age of items as stored in A_{N_M}.

6   Objective Losses

To train our model, we utilize the different objective losses described below.

6.1   Classification Loss

We calculate the probability of a character belonging to a particular trope category P through Equation 11, where $f_P : \mathbb{R}^{d_h} \mapsto \mathbb{R}^{N_P}$ is a fully-connected layer, and z is the persona representation produced by the multi-level attentive encoders described in Equation 1. We then optimize the categorical cross-entropy loss between the predicted and true tropes as in Equation 12, where N_P is the total number of tropes, q_j is the predicted distribution that the input character fulls under trope j, and $p_j \in \{0, 1\}$ denotes the ground-truth of whether the input snippets come from characters from the jth trope.

$$q = softmax(f_P(z)) \tag{11}$$

$$J_{CE} = \sum_{j=1}^{N_P} -p_j \log(q_j) \tag{12}$$

6.2   Trope Description Triplet Loss

In addition to using trope descriptions to initialize the Knowledge-Store Memory, we also test learning from the trope descriptions through a triplet loss (Hoffer and Ailon, 2015). We again use the skip thought vectors to represent the descriptions. Specifically, we want to maximize the similarity of representations obtained from dialogue snippets with their corresponding description, and minimize their similarity with negative examples. We implement this as:

$$R^P = f_D(z) \tag{13}$$

$$J_T = max(0, s(R^P, R_n^D) - s(R^P, R_p^D) + \alpha_T) \tag{14}$$

where $f_D : \mathbb{R}^{d_h} \mapsto \mathbb{R}^{\|R^D\|}$ is a fully-connected layer. The triplet ranking loss is then Equation 14, where α_T is a learnable margin parameter and $s(\cdot, \cdot)$ denotes the similarity between trope embeddings (R^P), positive (R_p^D) and negative (R_n^D) trope descriptions.

Trope Description Triplet Loss with Memory Module

If a memory module is used, we compute a new triplet loss in place of the one described in Equation 14. Models that use a memory module should learn a representation $\hat{z}_M$, based on either the prior knowledge stored in the memory (as in Knowledge-Store memory) or the top-k key matches (as in Read-Write memory), that is similar to the representation of the trope descriptions.

This is achieved by replacing the persona embedding z in Equation 13 with the memory output $\hat{z}_M$ as shown in Equation 15, where $f_{D_M} : \mathbb{R}^{d_h} \mapsto \mathbb{R}^{||R^D||}$ is a fully-connected layer. To compute the new loss, we combine the representations obtained from Equations 13 and 15 through a learnable parameter γ that determines the importance of each representation. Finally, we utilize this combined representation $\hat{R}^P$ to calculate the loss as shown in Equation 17.

$$R_M^P = f_{D_M}(\hat{z}_M) \tag{15}$$

$$\hat{R}^P = \gamma R^P + (1-\gamma)R_M^P \tag{16}$$

$$J_{MT} = max(0, s(\hat{R}^P, R_n^D) - s(\hat{R}^P, R_p^D) + \alpha_{MT}) \tag{17}$$

6.3 Read-Write Memory Losses

When the Read-Write memory is used, we use two extra loss functions. The first is a Memory Ranking Loss J_{MR} as done in (Kaiser and Nachum, 2017), which learns based on whether a query with the persona embedding z returns nearest neighbors with the correct trope. The second is a Memory Classification Loss J_{MCE} that uses the values returned by the memory to predict the trope. The full details for both are found in Supplementary Section A.

6.4 Overall Loss

We combine the above losses through:

$$\begin{aligned} J = &\ \beta_{CE} \cdot J_{CE} \\ &+ \beta_T \cdot \hat{J}_T \\ &+ \beta_{MR} \cdot J_{MR} + \ \beta_{MCE} \cdot J_{MCE} \end{aligned} \tag{18}$$

$$\hat{J}_T = \begin{cases} J_{MT} & \text{if memory module is used} \\ J_T & \text{otherwise.} \end{cases}$$

where $\beta = [\beta_{CE}, \beta_{MCE}, , \beta_T, \beta_{MR}]$ are learnable weights such that $\sum_i \beta_i = 1$. Depending on which variant of the model is being used, the list β is modified to contain only relevant losses. For example, when the Knowledge-Store memory is used, we set $\beta_{MR} = \beta_{MCE} = 0$ and β is modified

to $\beta = [\beta_{CE}, \beta_T]$. We discuss different variants of our model in the next section.

7 Experiments

We experimented with combinations of our various modules and losses. The experimental results and ablation studies are described in the following sections, and the experimental details are described in Supplementary Section B. The different model permutation names in Table 2, e.g. "attn_3_tropetrip_ks-mem_ndialog16", are defined as follows:

- *baseline vs attn*: The 'baseline' model uses only one dialogue snippet S to predict the trope, i.e. $N_{diag} = 1$. Hence, the inter-snippet encoder is not used. The 'attn' model operates on N_{diag} dialogue snippets using the inter-snippet encoder to assign an attention score for each snippet S_i.

- *char vs. 3*: To measure the importance of context and other characters' lines, we have two variants – 'char' uses only the character's lines, while '3' uses the character's lines, other character's lines, and all context lines. Formally, in 'char' mode, we set γ_E^s and γ_O^s to 0 in Equation 1. In 'attn' mode, $(\gamma_E^s, \gamma_O^s, \gamma_D^s)$ are learned by the model.

- *tropetrip*: The presence of 'tropetrip' indicates that the triplet loss on the trope descriptions was used. If '-500' is appended to 'tropetrip', then the 4800-dimensional skip embeddings representing the descriptions in Equations 15 and 17 are projected to 500 dimensions using a fully connected layer.

- *ks-mem vs. rw-mem*: 'ks-mem' refers to the Knowledge-Store memory, and 'rw-mem' refers to the Read-Write memory.

- *ndialog*: The number of dialogue snippets N_{diag} used as input for the attention models. Any attention model without the explicit N_{diag} listed uses $N_{diag} = 8$.

7.1 Ablation Results

Baseline vs. Attention Model. The attention model shows a large improvement over the baseline models. This matches our intuition that not every quote is strongly indicative of character trope. Some may be largely expository or 'chit-chat' pieces of dialogue. Example attention scores are shown in Section 7.2.

Model	Accuracy	Precision	Recall	F1
baseline_char	0.286	0.538	0.286	0.339
baseline_3	0.287	0.552	0.288	0.349
attn_char	**0.630**	0.586	0.628	0.600
attn_3	0.630	**0.590**	**0.630**	**0.603**
attn_3_tropetrip	0.615	0.566	0.615	0.583
attn_3_tropetrip-500	**0.644**	**0.601**	**0.642**	**0.615**
attn_3_ks-mem	**0.678**	**0.648**	**0.676**	**0.657**
attn_3_rw-mem	0.635	0.598	0.635	0.611
attn_3_tropetrip_ks-mem	**0.663**	**0.628**	**0.662**	**0.639**
attn_3_tropetrip-500_ks-mem	0.654	0.618	0.652	0.629
attn_3_tropetrip_rw-mem	0.649	0.602	0.648	0.617
attn_3_tropetrip-500_rw-mem	0.644	0.608	0.644	0.619
attn_3_tropetrip-500_ks-mem_ndialog16	0.740	0.707	0.741	0.718
attn_3_tropetrip-500_ks-mem_ndialog32	**0.750**	**0.750**	**0.750**	**0.750**
attn_3_tropetrip_ks-mem_ndialog16	0.740	0.722	0.741	0.728
attn_3_tropetrip_ks-mem_ndialog32	0.731	0.712	0.731	0.718

Table 2: Experimental results. Details and analysis are given in Section 7.1. The best performing results in each block are bolded. The first block examines the baseline model vs. the attention model, as well as use of different inputs. The second block uses the triplet loss, and the third block uses our memory modules. The fourth block combines the triplet loss and memory module, which the fifth block extends to larger N_{diag}.
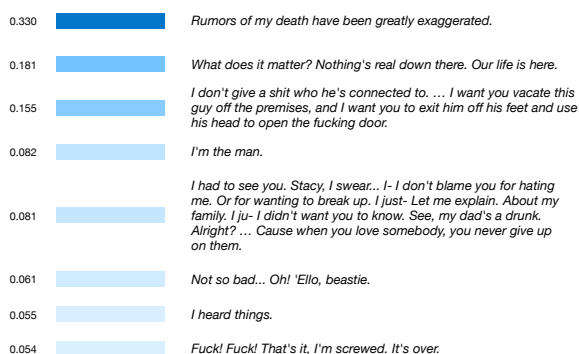
0.330	*Rumors of my death have been greatly exaggerated.*
0.181	*What does it matter? Nothing's real down there. Our life is here.*
0.155	*I don't give a shit who he's connected to. ... I want you vacate this guy off the premises, and I want you to exit him off his feet and use his head to open the fucking door.*
0.082	*I'm the man.*
0.081	*I had to see you. Stacy, I swear... I- I don't blame you for hating me. Or for wanting to break up. I just- Let me explain. About my family. I ju- I didn't want you to know. See, my dad's a drunk. Alright? ... Cause when you love somebody, you never give up on them.*
0.061	*Not so bad... Oh! 'Ello, beastie.*
0.055	*I heard things.*
0.054	*Fuck! Fuck! That's it, I'm screwed. It's over.*

Figure 4: Attention scores for a batch of dialogues for the "byronic hero" trope [3].

Though our experiments showed marginal improvement between using the 'char' data and the '3' data, we found that using all 3 inputs had greater performance for models with the triplet loss and read-only memory. This is likely because the others' lines and context capture more of the social dynamics and situations that are described in the trope descriptions. Subsequent results are shown only for the 'attn_3' models.

Trope Description Triplet Loss. Adding the trope description loss alone provided relatively small gains in performance, though we see greater gains when combined with memory. While both use the descriptions, perhaps the Knowledge Store memory matches an embedding against all the

---

[3]TVtropes.org defines a byronic hero as "Sometimes an Anti-Hero, others an Anti-Villain, or even Just a Villain, Byronic heroes are charismatic characters with strong passions and ideals, but who are nonetheless deeply flawed individuals...."

tropes, whereas the trope triplet loss is only provided information from one positive and one negative example.

Memory Modules. The Knowledge-Store memory in particular was helpful. Initialized with the trope descriptions, this memory can 'sharpen' queries toward one of the tropes. The Read-Write memory had smaller gains in performance. It may be that more data is required to take advantage of the write capabilities.

Combined Trope Description Triplet Loss and Memory Modules. Using the triplet loss with memory modules led to greater performance when compared to the $attn\_3$ model, but the performance sits around the use of either triplet only or memory only. However, when we increase the N_{diag} to 16 or 32, we find a jump in performance. This is likely the case because the model has both increased learning capacity and a larger sample of data at every batch, which means at least some of the N_{diag} quotes should be informative about the trope.

7.2 Attention Scores

Because the inter-snippet encoder provides such a large gain in performance compared to the baseline model, we provide an example illustrating the weights placed on a batch of N_{diag} snippets. Figure 4 shows the attention scores for the character's lines in the "byronic hero" trope. Matching what we might expect for an antihero personality, we find the top weighted line to be full of confidence and heroic bluster, while the middle lines

hint at the characters' personal turmoil. We also find the lowly weighted sixth and seventh lines to be largely uninformative (e.g. "I heard things."), and the last line to be perhaps too pessimistic and negative for a hero, even a byronic one.

7.3 Purity scores of character clusters

Finally, we measure our ability to recover the trope 'clusters' (with one trope being a cluster of its characters) with our embeddings through the purity score used in (Bamman et al., 2014). Equation 19 measures the amount of overlap between two clusterings, where N is the total number of characters, g_i is the i-ith ground truth cluster, and c_j is the j-th predicted cluster.

$$\text{Purity} = \frac{1}{N} \sum_i \max_j |g_i \cap c_j| \qquad (19)$$

We use a simple agglomerative clustering method on our embeddings with a parameter k for the number of clusters. The methods in (Bamman et al., 2014) contain a similar hyper-parameter for the number of persona clusters. We note that the metrics are not completely comparable because not every character in the original dataset was found on IMDB. The results are shown in Table 3. It might be expected that our model perform better because we use the character tropes themselves as training data. However, dialogue may be noisier than the movie summary data; their better performing Persona Regression (PR) model also uses useful metadata features such as the movie genre and character gender. We simply note that our scores are comparable or higher.

k	PR	DP	AMN
25	42.9	39.63	**48.4**
50	36.5	31.0	**48.1**
100	30.3	24.4	**45.2**

Table 3: Cluster purity scores. k is the number of clusters, PR and DP are the Persona Regression and Dirichlet Persona models from (Bamman et al., 2014), and AMN is our attention memory network.

8 Application: Narrative Analysis

We collected IMDB quotes for the top 250 movies on IMDB. For every character, we calculated a character embedding by taking the average embedding produced by passing all the dialogues through our model. We then calculated movie embeddings by taking the weighted sum of all the character embeddings in the movie, with the

weight as the percentage of quotes they had in the movie. By computing distances between pairs of character or movie embeddings, we could potentially unearth notable similarities. We note some of the interesting clusters below.

8.1 Clustering Characters

- Grumpy old men: Carl Fredricksen (Up); Walk Kowalski (Gran Torino)
- Shady tricksters, crooks, well versed in deceit: Ugarte (Casablanca); Eames (Inception)
- Intrepid heroes, adventurers: Indiana Jones (Indiana Jones and the Last Crusade); Nemo (Finding Nemo); Murph (Interstellar)

8.2 Clustering Movies

- Epics, historical tales: Amadeus, Ben-Hur
- Tortured individuals, dark, violent: Donnie Darko, Taxi Driver, Inception, The Prestige
- Gangsters, excess: Scarface, Goodfellas, Reservoir Dogs, The Departed, Wolf of Wall Street

9 Conclusion

We used the character trope classification task as a test bed for learning personas from dialogue. Our experiments demonstrate that the use of a multi-level attention mechanism greatly outperforms a baseline GRU model. We were also able to leverage prior knowledge in the form of textual descriptions of the trope. In particular, using these descriptions to initialize our Knowledge-Store memory helped improved performance. Because we use short text and can leverage domain knowledge, we believe future work could use our models for applications such as personalized dialogue systems.

References

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

David Bamman, Brendan O'Connor, and Noah A Smith. 2014. Learning latent personas of film characters. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, page 352.

Murray R Barrick and Michael K Mount. 1993. Autonomy as a moderator of the relationships between the big five personality dimensions and job performance. *Journal of applied Psychology*, 78(1):111.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Lucie Flekova and Iryna Gurevych. 2015. Personality profiling of fictional characters using sense-level links between lexical resources. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1805–1816.

Jennifer Golbeck, Cristina Robles, Michon Edmondson, and Karen Turner. 2011. Predicting personality from twitter. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 149–156. IEEE.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Elad Hoffer and Nir Ailon. 2015. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer.

John T Jost, Tessa V West, and Samuel D Gosling. 2009. Personality and ideology as determinants of candidate preferences and obama conversion in the 2008 us presidential election. *Du Bois Review: Social Science Research on Race*, 6(1):103–124.

Łukasz Kaiser and Ofir Nachum. 2017. Aurko roy, and samy bengio. *Learning to remember rare events. arXiv preprint*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

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 *International Conference on Machine Learning*, pages 1378–1387.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155*.

François Mairesse and Marilyn Walker. 2007. Personage: Personality generation for dialogue. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 496–503.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

Clifford Nass and Kwan Min Lee. 2000. Does computer-generated speech manifest personality? an experimental test of similarity-attraction. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 329–336. ACM.

Maarten Selfhout, William Burk, Susan Branje, Jaap Denissen, Marcel Van Aken, and Wim Meeus. 2010. Emerging late adolescent friendship networks and big five personality traits: A social network approach. *Journal of personality*, 78(2):509–538.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*.

Phillip R Shaver and Kelly A Brennan. 1992. Attachment styles and the" big five" personality traits: Their connections with each other and with romantic relationship outcomes. *Personality and Social Psychology Bulletin*, 18(5):536–545.

Shashank Srivastava, Snigdha Chaturvedi, and Tom M Mitchell. 2016. Inferring interpersonal relations in narrative summaries. In *AAAI*, pages 2807–2813.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Ernest C Tupes and Raymond E Christal. 1992. Recurrent personality factors based on trait ratings. *Journal of personality*, 60(2):225–251.

Josep Valls-Vargas, Jichen Zhu, and Santiago Ontanón. 2014. Toward automatic role identification in unannotated folk tales. In *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Marilyn A Walker, Janet E Cahn, and Stephen J Whittaker. 1997. Improvising linguistic style: Social and affective bases for agent personality. In *Proceedings of the first international conference on Autonomous agents*, pages 96–105. ACM.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*.