Representing Movie Characters in Dialogues

Mahmoud Azab[1], **Noriyuki Kojima**[1], **Jia Deng**[2], **Rada Mihalcea**[1]

[1]Computer Science and Engineering
University of Michigan, Ann Arbor
{mazab,kojimano,mihalcea}@umich.edu

[2]Department of Computer Science
Princeton University
jiadeng@princeton.edu

Abstract

We introduce a new embedding model to represent movie characters and their interactions in a dialogue by encoding in the same representation the language used by these characters as well as information about the other participants in the dialogue. We evaluate the performance of these new character embeddings on two tasks: (1) character relatedness, using a dataset we introduce consisting of a dense character interaction matrix for 4,761 unique character pairs over 22 hours of dialogue from eighteen movies; and (2) character relation classification, for fine- and coarse-grained relations, as well as sentiment relations. Our experiments show that our model significantly outperforms the traditional Word2Vec continuous bag-of-words and skip-gram models, demonstrating the effectiveness of the character embeddings we introduce. We further show how these embeddings can be used in conjunction with a visual question answering system to improve over previous results.

1 Introduction

Understanding characters (or more broadly people) plays a critical role in the human-level interpretation of dialogues – be those in stories, movies, or day-to-day conversations. The verbal interaction between characters provides important information (Iyyer et al., 2016; Elson et al., 2010). In these contexts, the names of characters trigger reasoning at a much deeper level than other regular words, due to the character background, behaviors, social network, and so forth. Currently, the most commonly used word embedding models such as Word2Vec (Mikolov et al., 2013a,b) and Glove (Pennington et al., 2014) represent characters using the embeddings corresponding to the tokens used to name them. Using these models in a dialogue setting to represent the characters poses three main issues. First, name mentions in dialogues are sparse (Azab et al., 2018), which makes it difficult for these models to learn a good quality representation for these names (Barteld, 2017). Second, in dialogues or narratives, names often do not refer to the same person, and yet these embeddings have a single vector representation for each word in the vocabulary. For example, "Danny" in the dialogue of the "American History X" movie is different from "Danny" in the "Ocean's Eleven" movie. Finally, the learned embeddings of these names reflect the co-occurrences of these name mentions and other words uttered by these characters, but do not model how related these characters are. Thus, the resulting embeddings cannot be effectively used to further reason about the characters and their relations.

Henry:	I did not know you could fly a plane.
Indiana:	Fly yes. Land no. Dad, you have to use the machine gun. Get it ready. Eleven o'clock!
Henry:	What happens at eleven o'clock?
Indiana:	Twelve, eleven, ten. Eleven o'clock, fire! Dad, are we hit?
Henry:	More or less. Son, I am sorry. They got us.
Indiana:	Hang on, dad. We are going in.

Table 1: A snippet of conversation between two characters from the "Indiana Jones and the Last Crusade" movie with each dialogue turn annotated with its corresponding speaker name. We aim to generate embedding representations for "Indiana" and "Henry" in a way that captures their relation.

The representation of characters in dialogues has been an important task for social network extraction (Elson et al., 2010), character relation modeling (Chaturvedi et al., 2016), and persona-based conversation models (Li et al., 2016). However, most of the previous work relies upon the ex-

traction of linguistic features like explicit forms of address (Makazhanov et al., 2014), the length of the utterance, or the frequency of exchanges between the characters (Elson et al., 2010).

In this work, we address the task of representing characters in dialogues, specifically focusing on movies and plays. Given a set of dialogue turns, annotated with the corresponding speaker names, our goal is to generate a vector representation for each of these characters that captures the relation with other characters. We propose a new approach to embed characters in dialogues based not only on what a character is saying, but also to whom. This model allows the information from the words in a dialogue turn to propagate to the representation of the previous and following speakers.

Despite its simplicity, our model yields strong empirical performance. By evaluating our model on two different tasks – namely character relatedness and character relation classification (fine-grained, coarse-grained, and sentiment) – we find that the model exceeds by a large margin several strong baselines, which indicates that our model effectively captures the various characteristics of characters. Additionally, in the process of evaluating the model, we build a new dataset consisting of 4,761 character relation pairs obtained from eighteen movies, manually annotated with relatedness scores and relations of various granularities. We are making the dataset publicly available.

2 Related Work

Learning distributional representation of words plays an increasingly important role in representing text in many tasks (Bengio et al., 2013; Chen and Manning, 2014). The existence of huge datasets allowed learning high quality word embeddings in an unsupervised way by training a neural network on fake objectives (Mikolov et al., 2013a,b; Turney and Pantel, 2010). A major strength of these learned word embeddings is that they are able to capture useful semantic information that can be easily used in other tasks of interest such as semantic similarity and relatedness between pair of words (Mikolov et al., 2013a; Pennington et al., 2014; Wilson and Mihalcea, 2017) and dependency parsing (Chen and Manning, 2014; Dyer et al., 2015). However, these models treat names and entities no more than the tokens used to mention them. As a result, these models are unable to well represent names in narrative understanding task because the word "John" in a given story can be very different from the word "John" in another narrative. In this work, we only focus on representing character names and not the whole embedding space (Ji et al., 2017).

Recently, several approaches have been proposed to build dynamic representations for entities (Henaff et al., 2016; Ji et al., 2017; Kobayashi et al., 2016, 2017). One common approach is to rely on neural language models to encode the local context of an entity and use the resulting context vectors as the embedding for subsequent occurrences of that entity (Kobayashi et al., 2016, 2017). Another approach is to learn a generative model that generates the representation of an entity mention (Ji et al., 2017). Henaff et al. (2016) proposed an explicit entity tracking model by relying on an external memory to store information about entities as they appear in a given sentence. While these rich representations improve the performance on several tasks such as coreference and reading comprehension, they rely on explicit mentions of entities in text as available in toy datasets such as bAbi (Weston et al., 2015). Thus, it is difficult to apply these representations in a dialogue setting due to the sparseness of name mentions in dialogue, as well as the lack of explicit conversation connections between characters (as available in movies) (Azab et al., 2018). Most of the existing story understanding work feeds the model with the vector representations of names based on a global model such as Word2Vec or Glove, which hinders the ability of these models to understand dialogue (Tapaswi et al., 2016; Na et al., 2017; Lei et al., 2018). Recently, Li et al. (2016) relied on TV series scripts in order to learn speaker persona representations and used these representations to improve the performance of neural conversation models. Unlike (Ji et al., 2017; Li et al., 2016), we focus on representing character names in dialogue settings and learning different embeddings for characters from different story dialogues in a way that reflects the relatedness of story characters; more specifically, we propose the use of speaker prediction as an auxiliary supervision to improve the character representation.

Identifying and analyzing character relations in literary texts is a well studied problem (Agarwal et al., 2013; Makazhanov et al., 2014; Elson et al., 2010; Iyyer et al., 2016). Most of these models depend on analyzing the co-occurrence of the char-

acters and stylistic features used while characters address each other. These models are really important to summarize, understand, and generate stories (Elson et al., 2010). In this work, we use the task of character relation classification as an extrinsic evaluation task to evaluate the impact of character embeddings on this task.

3 Character Embeddings

Characters play an important role in any dialogue, including movies or plays. Yet, work to date has rarely considered specialized character representations. We hypothesize that a representation that leverages both the language uttered by the characters as well as information on the other characters in the dialogue could result in richer encodings. The intuition behind our hypothesis is explained by the example in table 1. Here, the word "Dad" should be associated not only with "Indiana" but also propagate its information to "Henry", conditioned by "Indiana". Our proposed model is well conveying this intuition to encode characters.

3.1 Setup

Our architecture builds on a pretrained embedding model generated by standard Word2Vec models (Mikolov et al., 2013a,b) or pre-trained contextualized word representations from neural language models (ELMo) (Peters et al., 2018). We start by collecting sets of (current speaker, previous speakers, next speakers, context words) as training examples. We split the four elements in the sets into target and context depending on our objectives. Figure 1 describes the input-output (target-context) pairs of our system. Additionally, our model works as an unsupervised post-training of existing embeddings, rather than starting the training from scratch. This is due to the fact that getting a good representation for characters is a separate task from getting a general representation of tokens. A good pre-trained embedding space is an essential component to map characters so that they will be distributed in a semantically meaningful embedding space. While a good pre-trained embedding is important, our models focus on "moving" the character embeddings without affecting any other word representations.

3.2 Architecture

We propose two post-training schemes, which we refer to as Character Embedding (SG) and Character Embedding (CBOW). The differences stand in the objective of post-training, given sets of (current speaker, previous speakers, next speakers, context words) as training examples. Formally, given the sequence of speakers at each turn $S = s_1, s_2, s_3, , , s_{T-1}, s_T$, we define context words C for turn t as the set of words found by a sliding context window in the utterance. We propose our post-training objectives as following:

$$L = \frac{1}{N} \sum_{s_i \in S} \sum_{w_i \in C(s_i)} \sum_{-sw \leq j \leq sw} \log(p(w_i|s_{i+j}))) \quad (1)$$

$$L = \frac{1}{N} \sum_{s_i \in S} \sum_{w_i \in C(s_i)} (\log(p(s_i|w_i) + \sum_{-sw \leq j \leq sw, j \neq 0} \log(p(s_i|s_{i+j}))) \quad (2)$$

Our Character Embedding (SG) model maximizes the objective on Equation 1, while Character Embedding (CBOW) maximizes the objective on Equation 2, where N indicates the number of training examples and sw indicates the size of the speaker window (speaker window of size one means we consider speakers of one preceding turn and one succeeding turn). Our formulation defines probabilities $p(s_i|w_i)$, $p(s_i|s_{i+j})$ and $p(w_i|s_{i+j})$ using the softmax equation. We also define two transformations of our network – lookup table (LUT) initialized by embedding of pre-trained embedding model and Linear Projection Layer W.

To examine the generality of our post-training schemes, we also apply them to another pre-trained word embedding model. Given a dialogue turn, we encode it using ELMo's pre-trained Bi-LSTM model (Peters et al., 2018) to generate a sequence of contextualized vectors for words. We add a linear projection layer on top that takes the generated embedding, in addition to the previous and following speakers, and train it to predict the speaker of the current turn. We refer to this model as Character Embedding (ELMo).

3.3 Training

We represent our contexts and targets as a one hot vector of length equal to the vocabulary size. The purpose of our model is to update the embedding

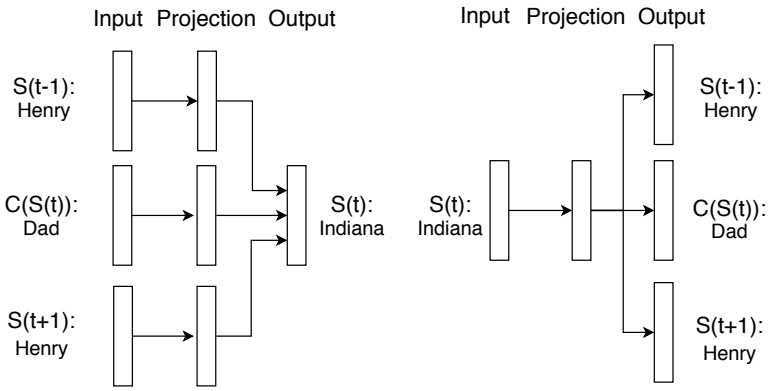

Figure 1: The conceptual figure describing input /output pairs of our character embedding model. The diagram describes when both the speaker window and the context window are size one. **Left**: Character Embedding(CBOW), **Right**: Character Embedding(SG).

of characters in LUT by propagating the gradient from our objectives. We use cross-entropy to calculate the loss, and we use gradient descent to update the parameters. The description of our Character Embedding (SG) model with a speaker window size of one is showed in Algorithm 1.

4 Evaluation Tasks and Datasets

We evaluate the quality of our speaker embedding model across two different tasks. Our goal is to evaluate how well each embedding model captures simple and complex character representations and interactions.

4.1 Character Relatedness

Measures of semantic relatedness between words indicate the degree to which words are associated with any kind of semantic relationship such as synonymy, antonymy, and so on. Semantic relatedness is commonly used as an absolute intrinsic evaluation task to assess and compare the quality of different word embeddings (Schnabel et al., 2015; Yih and Qazvinian, 2012; Upadhyay et al., 2016) and phrase embeddings (Wilson and Mihalcea, 2017).

Similarly, we define character relatedness as the degree to which a pair of characters in a given story are related to each other based on the story plot and their level of interaction throughout the dialogue. Given a pair of characters, we would like the relatedness score between their embedding representations to have a high correlation with their corresponding human-based relatedness score. Thus, the distance of the embeddings between closely related characters should be smaller than the distance between less related ones.

To measure the relatedness between characters in movies, we construct a new annotated dataset based on a publicly available dataset (Azab et al., 2018). That dataset includes 28K turns spoken by 396 different speakers in eighteen movies covering different genres, with the subtitles of each movie labeled with the character name of their corresponding speakers. On average, each character uttered 452 words.

For each movie in that dataset, two human annotators watched the movies and annotated a dense relatedness matrix of characters on a 1-5 scale. Table 2 shows the meaning of each score. These scores reflect the level of interaction or how closely related the characters are over the course of the movie. For example, given two characters X and Y, a high score for X and Y is assigned if e.g., X is the father of Y, regardless of the amount of interaction between the two characters. We also give a high score for the cases where X and Y are closely interacted, even if they are unrelated in terms of kinship. Due to the sparseness of the number of closely related characters, we asked the annotators to select the higher score when hesitating between two scores.

For three movies, the Pearson correlation between the two annotators is 0.8394, which reflects a very good agreement. We then average the scores assigned by the annotators and use the result as the human relatedness ground-truth score for each pair of characters.

In this dataset, we have 4,761 unique character pairs annotated with a relatedness score. Figure 2 shows the statistics over the relatedness scores. As shown in the table, only a small number of character pairs are closely related, while the majority

Algorithm 1: Character Embedding(SG)

E: The embedding from pre-trained model
W: Linear Projection Layer
α: Learning Rate
maxepoch: maximum epoch to run
LUT $\leftarrow$ E, epoch $\leftarrow$ 1;
while $epoch \leq maxepoch$ **do**
  for *t from 2 to* $T-1$ **do**
    $x_1 \leftarrow LUT[s_{t-1}]$;
    $x_2 \leftarrow LUT[s_t]$;
    $x_3 \leftarrow LUT[s_{t+1}]$;
    for w_0 *in* $C(s_t)$ **do**
      target $\leftarrow LUT[w_0]$;
      logits $= tanh(W^T(x_1+x_2+x_3))$;
      prediction $= softmax$(logits);
      loss $= -$target $+ \log$(prediction);
      $W := W - \alpha * \frac{\delta loss}{\delta W}$;
      $LUT[s_{t-1}] := x_1 - \alpha * \frac{\delta loss}{\delta x_1}$;
      $LUT[s_t] := x_2 - \alpha * \frac{\delta loss}{\delta x_2}$;
      $LUT[s_{t+1}] := x_3 - \alpha * \frac{\delta loss}{\delta x_3}$;
    end
  end
  epoch := epoch + 1
end

5	interacted frequently/closely related
4	interacted/related
3	moderately interacted/somewhat related
2	interacted few times/not related
1	did not interact/not related

Table 2: Relatedness annotation scores.

of the characters have either interacted very few times or did not interact at all. However, it is important to include these unrelated pairs while evaluating the quality of the character embeddings, as unrelated pairs might be closer than related ones especially for minor characters that do not speak much during the dialogue.

4.2 Character Relationships

Understanding the relationships between characters is a primary task in extracting and analyzing social relation networks from literary novels (Elson et al., 2010; Agarwal et al., 2013). It is also important for improving computational story summarization and generation methods (Elsner, 2012; Gorinski and Lapata, 2015).

Character relationship is a more complex task than character relatedness. In this task, given a pair of character embeddings, we would like to classify the type of their relationship on multiple dimensions. Specifically, we consider: fine-grained relations, such as sister/father/friend/enemy; coarse-grained relations, such as familial/social/professional; and relation sentiment, i.e., positive, negative or neural. The goal of this task is to evaluate the quality of our character embeddings and how well it captures such complex information in an unsupervised fashion. It also serves as an extrinsic evaluation for the impact of our character representations on downstream tasks.

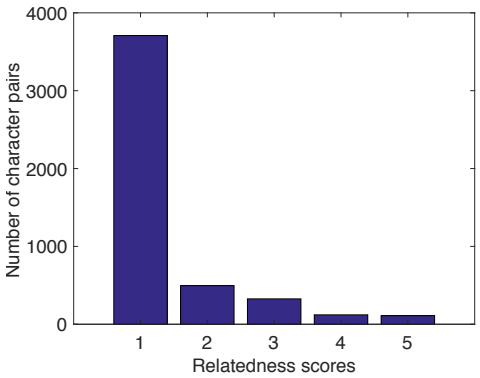

Figure 2: Statistics of the character relatedness dataset on movies of speaker naming dataset.

We use a subset of character relationships in a literary dataset (Massey et al., 2015). This dataset includes annotations for eighteen fine-grained relationship classes, four coarse-grained relationship classes, and three relation sentiment classes.[1] We use the 31 Shakespeare plays in this dataset, and obtain their corresponding text from project Gutenberg. We use the Shakespeare plays because they have the dialogue turns annotated with speakers names, which is necessary for training our character embedding models. The plays include a total of 605 character pair relationship annotations.

5 Experiments

5.1 Baselines

For each task, we compare our character embedding models against five baselines:

[1]Annotations on temporal change in the sentiment between each pair of characters is also included, but since our models do not have the ability to track such temporal information, we do not use these annotations.

Interaction Frequency. We count the number of exchanged dialogue turns between every pair of characters and normalize it by the total number of turns spoken by a given pair of characters.

TF-IDF. We treat all the utterances of a character as a document and calculate a tf-idf weight for each word. We then represent a character by its tf-idf vector of the words that they uttered.

Word2Vec (CBOW) model. We use the traditional Word2Vec architecture to train a word embedding space based on the continuous bag-of-words approach (Mikolov et al., 2013a). Given a sequence of words D, the context words that exist in a defined window size are considered as input to the network and the objective is to predict the target word by maximizing the average long probability:

$$L = \frac{1}{|D|} \sum_{w_i \in D} \log P(w_i | C(w_i)) \quad (3)$$

Word2Vec (SG) model. We use the skip-gram architecture of Word2Vec with negative sampling (Mikolov et al., 2013b). In this architecture, the objective is to learn a representation of the target word that would be good at predicting the words within a defined window by maximizing the average log probability:

$$L = \frac{1}{|D|} \sum_{w_i \in D} \sum_{w_0 \in C(w_i)} \log P(w_0 | w_i) \quad (4)$$

Character BOW. We represent each character as the mean-pooling of a 300-dimension pre-trained Word2Vec representation of all the words that this character has uttered through the entire dialogue.

Doc2Vec. We train a Doc2Vec model (Le and Mikolov, 2014) as tagged documents using the character names as the document tags. We then represent each character as the Doc2Vec representation of all the words that this character has uttered through the entire dialogue.

ELMo (Mean-Pooling). We use pre-trained contextualized word representations from neural language models (ELMo) (Peters et al., 2018) to generate character names representations based on the sentences that include their names.[2] To generate these representations, we feed the pre-trained ELMo model with a Glove representation for the words and ELMo augments their representation with the hidden states of its two layers bi-directional LSTM to represent the words with respect to their context. For each character name, we average their contextualized representations through the entire dialogue.

[2]We also tried training ELMo from scratch on our data but the pre-trained model produces better results.

5.2 Experimental Setting

To have these models trained on in-domain data, we use GenSim (Řehůřek and Sojka, 2010) to train the different architectures of Word2Vec on the almost 600K sentences / 4M words of subtitles and Shakespeare plays. For the target movies and plays, the speaker names are included in the training data so that we can have a vector representation for each character name. The names in our corpus have been manually normalized so that 'Joe' and 'Joseph' in a movie get the same representation, while 'Joseph' in a different movie gets a different representation. To achieve the first part of the name normalization, we utilize the name-clustering algorithm provided by Bamman (2014) to extract and cluster name tokens from the text and annotate the true representation of names for each cluster. We achieve the second part of the name normalization by adding the text title to the name tokens (e.g., 'Michael' becomes '$\text{Michael}_{Othello}$').

For GenSim (Řehůřek and Sojka, 2010), we set the learning rate to 0.1, the window size to 4 and the samples to 50 for negative sampling. We run 30 epochs to train our baselines. For post-training by our models, we use a gradient decent to update our parameters. For general experiments, we set the learning rate to 0.1 and the learning rate decays by the factor of 0.9 per 10 epochs. We run maximum 40 epochs for our post-training. For Character Embedding (CBOW), we use a context window of size two. We use a speaker window of size one for both the Character Embedding (CBOW) and the Character Embedding (SG).

5.3 Results

Character Relatedness. For each model, given a pair of characters we compute the cosine similarity score between the embeddings of these two characters, defined as:

$$similarity(\boldsymbol{C}1, \boldsymbol{C}2) = \frac{\boldsymbol{C}1 \cdot \boldsymbol{C}2}{||\boldsymbol{C}1|| \cdot ||\boldsymbol{C}2||} \quad (5)$$

and compute the similarity score between two characters in the embedding space similar to (Col-

Movie	Character	Methods	Closest	Second closest	Third Closest
The Devil's Advocate	Alice Lomax	Ground Truth	**Kevin Lomax**	**John Milton**	**Mary Lomax**
		Interaction Frequency	**Kevin Lomax**	Pam Garrety	John Milton
		TF-IDF	Mary Lomax	John Milton	Don King
		Character Average BOW	John Milton	Kevin Lomax	Barbara
		Word2Vec (CBOW)	Lloyd Gettys	Judge Poe	Alexander Cullen
		Word2Vec (SG)	Alfonse D'amato	Lloyd Gettys	Judge Poe
		ELMo (Mean-Pooling)	**Kevin Lomax**	Mary Lomax	Alexander Cullen
		Character Embedding(CBOW)	**Kevin Lomax**	Judge Poe	**Mary Lomax**
		Character Embedding(SG)	**Kevin Lomax**	**John Milton**	**Mary Lomax**
		Character Embedding(ELMo)	**Kevin Lomax**	Pam Garrety	**Mary Lomax**

Table 3: Example of character relatedness task. Given a character, we list the top three characters sorted in descending order from left to right according to their similarity scores.

lobert et al., 2011; Mikolov et al., 2013b). The list of the nearest characters of a given character C are all the other characters from the same movie sorted in descending order by their similarity score with respect to C.

	Pearson Coeff
Interaction Frequency	0.3632
TF-IDF	0.3129
Doc2Vec	0.1771
Word2Vec (CBOW)	0.2081
Word2Vec (SG)	0.1989
Character BOW	0.2256
ELMo (Mean-Pooling)	0.3212
Character Embedding(CBOW)	**0.4644**
Character Embedding(SG)	**0.4933**
Character Embedding(ELMo)	**0.3475**

Table 4: Comparison between the average Pearson correlation coefficient scores of the different models against average human relatedness scores.

Table 4 shows the Pearson correlation coefficients of the resulting similarity scores of each model against the average human annotation scores. These results suggest that having the context window over the utterance and adding the previous and next speakers to the input layer greatly improves the ability of the character embeddings to capture the relatedness between the different characters in a given story dialogue.

Table 3 shows an example of characters that are most related to "Alice Lomax" from the movie "The Devil's Advocate" as calculated based on each model sorted in descending order according to their cosine similarity scores. It is worth noting that Kevin Lomax is Alice's son, John Milton is Kevin's father and Mary Ann Lomax is Kevin's wife. On the other hand the characters suggested by both Word2Vec CBOW and SG models did not interact with Alice through the whole movie.

To further analyze the quality of the produced character embeddings, we evaluate the embeddings across different characters according to the their frequency of appearance in the movies. Figure 3 shows a comparison between the performance of the different models over minor and major characters based on the number of dialogue turns that each character uttered. These results show that our character embedding model consistently outperforms the traditional Word2Vec baseline models and reflect the robustness of our model in generating better character embeddings.

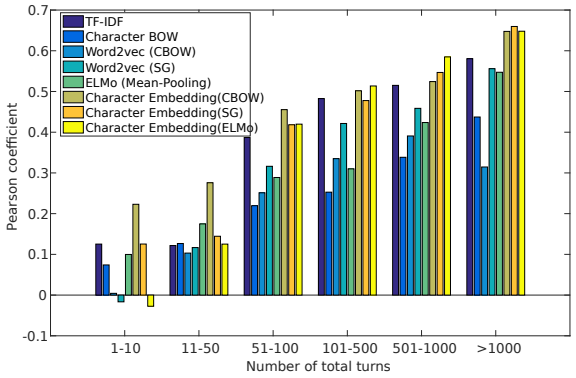

Figure 3: Comparison of the average Pearson correlation coefficient over characters who had different number of turns.

Character Relationship. We have three classification tasks for character relationships: 1) fine-grained relationship classification; 2) coarse-grained relationship classification; 3) relation sentiment classification. For each of these tasks, we train a logistic regression classifier using the Scikit-learn library (Pedregosa et al., 2011). These classifiers take a pair of character embeddings as a concatenation of their vectors and predict their

	Fine-grained Relation			Coarse-grained Relation			Sentiment		
	P	R	F	P	R	F	P	R	F
Interaction Frequency	0.04	0.16	0.06	0.30	0.44	0.33	0.33	0.58	0.42
TF-IDF	0.11	0.12	0.10	0.39	0.42	0.40	0.43	0.53	0.40
Character Average BOW	0.08	0.16	0.05	0.33	0.43	0.28	0.28	0.53	0.37
Word2Vec (CBOW)	0.11	0.13	0.12	0.37	0.38	0.38	0.39	0.40	0.39
Word2Vec (SG)	0.09	0.12	0.10	0.37	0.37	0.37	0.41	0.43	0.42
Doc2Vec	0.12	0.12	0.12	0.40	0.40	0.40	0.42	0.42	0.42
ELMo (Mean-Pooling)	0.14	0.18	0.14	0.39	0.41	0.40	0.44	0.50	0.46
Character Embedding(CBOW)	0.11	0.14	0.12	0.43	0.44	0.43	0.44	0.47	0.44
Character Embedding(SG)	0.11	0.17	0.12	0.43	0.46	0.42	0.40	0.51	0.42
Character Embedding (ELMo)	**0.18**	**0.19**	**0.19**	**0.48**	**0.48**	**0.48**	**0.48**	**0.48**	**0.48**

Table 5: Comparison between the average of the precision, recall and macro-weighted f-score of the baselines and our character embedding model on both fine-grained, coarse-grained character relation and sentiment classification.

Play	Char 1	. Char 2	Methods	Fine-grained	Coarse-grained	Senti-ment
			Ground Truth	**lovers**	**social**	**positive**
			Interaction Frequency	**lovers**	**social**	**positive**
The Two			TF-IDF	servant	**social**	negative
Gentlemen	Julia	Proteus	Character Average BOW	friend	**social**	**positive**
of Verona			Word2Vec (CBOW)	servant	familial	negative
			Word2Vec (SG)	servant	familial	**positive**
			ELMo (Mean-Pooling)	friend	**social**	**positive**
			Character Embedding(CBOW)	**lovers**	**social**	negative
			Character Embedding(SG)	**lovers**	**social**	**positive**
			Character Embedding(ELMo)	**lovers**	**social**	**positive**

Table 6: Example of classification task on Shakespeare's play, using different baselines and our character representation methods. The classification output consists of the relations of character 2 from character 1's perspective. A bold face indicates a correct relation classification.

relationship. We use a leave-one-play-out cross-validation in which character pairs from each play are used as a test set and character pairs from the other plays are used to train the models. Table 5 shows the classification average precision, recall and weighted F-score obtained by training the logistic regression classifiers using the character embeddings produced by the different models. Training classifiers using our character embedding models consistently outperforms the classifiers trained using the other models, which reflects the quality of the semantic information captured by our character embeddings when compared to other models. Table 6 shows examples of the three character relation classification tasks as classified by our character embedding models and the baselines.

Question Answering. As a final evaluation, we test the impact of our character embedding on dialogue understanding. TVQA (Lei et al., 2018) is a challenging dataset that includes 152.5K multiple choice question answers about 21.8K video clips from 6 TV shows such as the *Big Bang Theory*, *House*, and so on. These questions were created in a way that requires understanding of both the dialogue and the visual content of a given video. Each video clip includes the video frames and subtitles with speaker names aligned automatically with their corresponding show scripts (around 69% of the subtitle segments include speakers names). We follow the same dataset splits for training, validation, and test.

	Accuracy	
	Q+S	Q+S+V
MS (Glove) (Lei et al., 2018)	0.6515	0.6770
MS (Glove w/o names)	0.6177	0.6467
MS (CharEmbedding(CBOW))	**0.6590**	**0.6852**
MS (CharEmbedding(SG))	**0.6554**	**0.6884**

Table 7: Comparison on the TVQA validation dataset using the MS method with Glove and Glove fine-tuned using our proposed character embedding method.

To evaluate our embedding, we use the baseline implementation proposed with the TVQA dataset, namely Multi-Stream (MS). This model relies on bidirectional attention between context (represented by subtitles and/or visual content) and question answer pairs as queries to predict the correct answer (Lei et al., 2018). Visual features are included as textual labels of detected visual concepts in the frames of the video clip. To measure the effect of the person names on the model, we apply a named entity recognizer and replace the names with a fixed randomly generated embedding. Table 7 shows the results from the MS method using Glove, Glove with removing names from subtitles, and using a fine-tuned Glove using our character embedding model. The use of our character embeddings bring improvements over the pre-trained Glove embeddings, which demonstrates the usefulness of these character representations.

6 Conclusion

In this paper, we presented a novel unsupervised embedding model to represent characters and their interaction in a dialogue. Our embedding model produces character representations that reflect the language used by the characters as well as information about their relations with other characters. To evaluate the performance of our character embeddings, we experimented with two tasks on two datasets: (1) character relatedness, using a dataset we introduced consisting of a dense character interaction matrix for 4,761 unique character pairs over 22 hours of dialogue extracted from 18 movies; and (2) character relation classification, for fine- and coarse-grained relations, as well as relation sentiment. Our experiments show that our model significantly outperforms the traditional Word2Vec continuous bag-of-words and skip-gram models, thus demonstrating the effectiveness of the character embeddings we introduced. We further showed how the character embeddings can be used in conjunction with a visual question answering system to improve over previous results.

The dataset annotated with character relatedness scores introduced in the paper is publicly available from http://lit.eecs.umich.edu/downloads.html.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported by a Samsung research grant and by a DARPA grant HR001117S0026-AIDA-FP-045.

References

Apoorv Agarwal, Anup Kotalwar, and Owen Rambow. 2013. Automatic extraction of social networks from literary text: A case study on alice in wonderland. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*.

Mahmoud Azab, Mingzhe Wang, Max Smith, Noriyuku Kojima, Jia Deng, and Rada Mihalcea. 2018. Speaker naming in movies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

David Bamman. 2014. book-nlp: Natural language processing pipeline that scales to book-length documents. https://github.com/dbamman/book-nlp.

Fabian Barteld. 2017. Detecting spelling variants in non-standard texts. In *Proceedings of the Student Research Workshop at the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*.

Snigdha Chaturvedi, Shashank Srivastava, Hal Daumé III, and Chris Dyer. 2016. Modeling evolving relationships between characters in literary novels. In *AAAI*.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

Micha Elsner. 2012. Character-based kernels for novelistic plot structure. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

David K Elson, Nicholas Dames, and Kathleen R McKeown. 2010. Extracting social networks from literary fiction. In *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics.

Philip John Gorinski and Mirella Lapata. 2015. Movie script summarization as graph-based scene extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2016. Tracking the world state with recurrent entity networks. *arXiv preprint arXiv:1612.03969*.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A Smith. 2017. Dynamic entity representations in neural language models. *arXiv preprint arXiv:1708.00781*.

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. *arXiv preprint arXiv:1709.01679*.

Sosuke Kobayashi, Ran Tian, Naoaki Okazaki, and Kentaro Inui. 2016. Dynamic entity representation with max-pooling improves machine reading. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*.

Jie Lei, Licheng Yu, Mohit Bansal, and Tamara L Berg. 2018. Tvqa: Localized, compositional video question answering. *arXiv preprint arXiv:1809.01696*.

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*.

Aibek Makazhanov, Denilson Barbosa, and Grzegorz Kondrak. 2014. Extracting family relationship networks from novels. *arXiv preprint arXiv:1405.0603*.

Philip Massey, Patrick Xia, David Bamman, and Noah A Smith. 2015. Annotating character relationships in literary texts. *arXiv preprint arXiv:1512.00728*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*.

Seil Na, Sangho Lee, Jisung Kim, and Gunhee Kim. 2017. A read-write memory network for movie story understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*.

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)*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*.

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. Movieqa: Understanding stories in movies through question-answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. *arXiv preprint arXiv:1604.00425*.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Steven Wilson and Rada Mihalcea. 2017. Measuring semantic relations between human activities. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*.

Wen-tau Yih and Vahed Qazvinian. 2012. Measuring word relatedness using heterogeneous vector space models. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.