DETECTING ACTIONABLE ITEMS IN MEETINGS BY CONVOLUTIONAL DEEP STRUCTURED SEMANTIC MODELS

Yun-Nung Chen[⋆†] *Dilek Hakkani-Tür*[†] *Xiaodong He*[†]

[⋆]Carnegie Mellon University, Pittsburgh, PA
[†]Microsoft Research, Redmond, WA

yvchen@cs.cmu.edu, dilek@ieee.org, xiaohe@microsoft.com

ABSTRACT

The recent success of voice interaction with smart devices (human-machine genre) and improvements in speech recognition for conversational speech show the possibility of conversation-related applications. This paper investigates the task of actionable item detection in meetings (human-human genre), where the intelligent assistant dynamically provides the participants access to information (e.g. scheduling a meeting, taking notes) without interrupting the meetings. A convolutional deep structured semantic model (CDSSM) is applied to learn the latent semantics for human actions and utterances from human-machine (source genre) and human-human (target) interactions. Furthermore, considering the mismatch between source and target genre and scarcity of annotated data sets for the target genre, we develop adaptation techniques that adjust the learned embeddings to better fit the target genre. Experiments show that CDSSM performs better for actionable item detection compared to baselines using lexical features (27.5% relative) and other semantic features (15.9% relative) when the source genre and target genre match with each other. When the target genre mismatches with the source genre, our proposed adaptation techniques further improve the performance. The discussion and analysis of the experiments provide a reasonable direction for such an actionable item detection task[1].

***Index Terms*—** Actionable item, Convolotional Deep Structured Semantic Model (CDSSM), embeddings, adaptation.

1. INTRODUCTION

Meetings pose unique knowledge sharing opportunities, and have been a commonly accepted practice to coordinate work of multiple parties in organizations. With the surge of smart phones, computing devices have been easily accessible and real-time information search has been a common part of regular conversations [1]. Furthermore, recent improvements in conversational speech recognition suggest the possibility of automatic speech recognition and understanding on continual, in the background, audio recording of conversations [2]. In meetings, discussions could be a rich resource for identifying participants' next actions and helping them to accomplish those.

In this paper, we investigate a novel task of actionable item detection in meetings, with the goal of providing the participants easy access to information and performing actions that a personal assistant would handle without interrupting the meeting discussions. Actionable items in meetings would include discussions on scheduling, emails, action items, and search. Fig. 1 shows some meeting segments from the ICSI meeting corpus [3], where actionable items and their associated arguments are annotated. A meeting assistant would then take an appropriate action, such as opening the calendars of the involved participants for the dates being discussed, finding the emails and documents being discussed, or initiating a new one.

find_email
action: check emails of me011, search for any emails from them
me018: Have <from_contact_name>they</from_contact_name> ever responded to <contact_name>you</contact_name>?
me011: Nope.

send_email
action: email all participants, "link to An Anatomy of Spatial Description"
me015: Yeah it's - or - or just - Yeah. It's also all on my - my home page at E_M_L. It's called "An Anatomy of afind Spatial Description". But I'll send <email_content>that link</email_content>.

create_calendar_entry
action: open calendars of participants, marking times free for the three participants and schedule an event
mn015: I suggest w- to - for - to proceed with this in - in the sense that maybe, <date>throughout this week</date>, the <contact_name>three of us</contact_name> will - will talk some more about maybe segmenting off different regions, and we make up some - some toy a- observable "nodes" - is that what th-

Fig. 1. The ICSI meeting segments annotated with actionable items. The triggered intents are at the right part along with descriptions. The intent-associated arguments are labeled within texts.

Most of the previous work on language understanding of human-human conversations focus on analyzing task-oriented dialogues such as in customer care centers, and aim to infer semantic representations and bootstrap language understanding models [4, 5, 6, 7, 8, 9]. These would then be used in human-machine dialogue systems that automate the targeted task, such as travel arrangements. In this work, we assume presence of task-oriented dialogue systems (human-machine genre), such as personal assistants that can schedule meetings and send emails, and focus on adapting such systems to aid users in multi-party meetings (human-human genre).

Previous work on meeting understanding investigated detection of decisions [10, 11], action items [12], agreement and disagreements [13, 14], and summarization [15, 16, 17]. Our task is closest

[1]The data is available at http://research.microsoft.com/projects/meetingunderstanding/.

Human-Machine Genre

create_calendar_entry schedule a meeting with <contact_name>John</contact_name> <start_time>this afternoon</start_time>

Human-Human Genre

create_calendar_entry how about the <contact_name>three of us</contact_name> discuss this later <start_time>this afternoon</start_time>?

Fig. 2. The genre mismatched examples with the same action.

to detection of action items, where action items are considered as a subgroup of actionable items.

Utterances in the human-human genre are more casual and include conversational terms, but the terms related to the actionable item, such as dates, times, and participants are similar. Fig. 2 shows genre-mismatched examples (human-machine v.s. human-human), where both utterances have the same action create_calendar_entry. The similarity between two genres suggests that the data available from human-machine interactions (source genre) can be useful in recognizing actionable items in human-human interactions (target genre). Furthermore, due to the mentioned differences, the use of adaptation methods could be promising.

In this paper, we treat actionable item detection in meetings as a meeting utterance classification task, where each user utterance can trigger an actionable item. Recent studies used CDSSM to map questions into relation-entity triples for question answering [18, 19], which motivates us to use CDSSM for learning relations between actions and their triggering utterances. Also, several studies investigated embedding vectors as features for training task-specific models [20, 21, 22, 23, 24], which can incorporate more informative cues from large data. Hence, for utterance classification, this paper focuses on taking CDSSM features to help detect triggered actions. In addition, embedding adaptation has been studied using different languages and external knowledge [25, 26]. Considering the genre mismatch, embedding adaptation is proposed to fit the target genre and provide additional improvement.

In the following sections, we describe how to train CDSSM for action item detection task in Section 2. Then we propose adaptation techniques to overcome the mismatch between the source and target genre in Section 3. Section 4 describes how to use the trained embeddings for the task. Section 5 and Section 6 discuss the experiments, and Section 7 concludes.

2. CONVOLUTIONAL DEEP STRUCTURED SEMANTIC MODELS (CDSSM)

Here we describe how to train CDSSM for actionable item detection.

2.1. Architecture

The model is a deep neural network with convolutional structure, where the architecture is illustrated in Fig. 3 [21, 27, 28, 29]. The model contains: 1) a word hashing layer obtained by converting one-hot word representations into tri-letter vectors, 2) a convolutional layer that extracts contextual features for each word with its neighboring words defined by a window, 3) a max-pooling layer that discovers and combines salient features to form a fixed-length sentence-level feature vector, and 4) a semantic layer that further transform the max-pooling layer to a low-dimensional semantic vector for the input sentence.

Word Hashing Layer l_h. Each word from a word sequence (i.e. an utterance) is converted into a tri-letter vector [28]. For example, the tri-letter vector of the word "#email#" (# is a word boundary symbol) has non-zero elements for "#em", "ema", "mai", "ail", and "il#" via a word hashing matrix W_h. Then we build a high-dimensional vector l_h by concatenating all word tri-letter vectors. The advantages of tri-letter vectors include: 1) OOV words can be represented by tri-letter vectors, where the semantics can be captured based on the subwords such as prefix and suffix; 2) the tri-letter space is smaller, where the total number of tri-letters in our experiments is about 20.6K. Therefore, incorporating tri-letter vectors improves the representation power of word vectors and also reduces the OOV problem while keeping the size small.

Convolutional Layer l_c. A convolutional layer extracts contextual features c_i for each target word w_i, where c_i is the vector concatenating the word vector of w_i and its surrounding words within a window (the window size is set to 3). For each word, a local feature vector l_c is generated using a tanh activation function and a global linear projection matrix W_c:

$$l_{ci} = \tanh(W_c^T c_i), \text{ where } i = 1, ..., d, \tag{1}$$

where d is the total number of windows.

Max-Pooling Layer l_m. The max-pooling layer forces the network to only retain the most useful local features by applying the max operation over each dimension of l_{ci} across i in (1),

$$l_{mj} = \max_{i=1,\ldots,d} l_{ci}(j). \tag{2}$$

The convolutional and max-pooling layers are able to capture prominent words of the word sequences [21, 27]. As illustrated in Fig. 3, if we view the local feature vector $l_{c,i}$ as a topic distribution of the local context window, e.g., each element in the vector corresponds to a hidden topic and the value corresponds to the activation of that topic, then taking the max operation at each element keeps the max activation of that hidden topic across the whole sentence.

Semantic Layer y. The global feature vector l_m in (2) is fed to feed-forward neural network layers to output the final non-linear semantic features y as the output layer.

$$y = \tanh(W_s^T l_m), \tag{3}$$

where W_s is a learned linear projection matrix. The output semantic vector can be either utterance embeddings y_U or action embeddings y_A.

2.2. Training Procedure

The meeting data contains utterances and associated actions. The idea of this model is to learn the embeddings for utterances and actions such that the utterances with the same actions can be close to each other in the continuous space. Below we define the semantic score between an utterance U and an action A using the cosine similarity between their embeddings:

$$\text{CosSim}(U, A) = \frac{y_U \cdot y_A}{\|y_U\|\|y_A\|}. \tag{4}$$

2.2.1. *Predictive Model*

The posterior probability of a possible action given an utterance is computed based on the semantic score through a softmax function,

$$P(A \mid U) = \frac{\exp(\text{CosSim}(U, A))}{\sum_{A'} \exp(\text{CosSim}(U, A'))}, \tag{5}$$

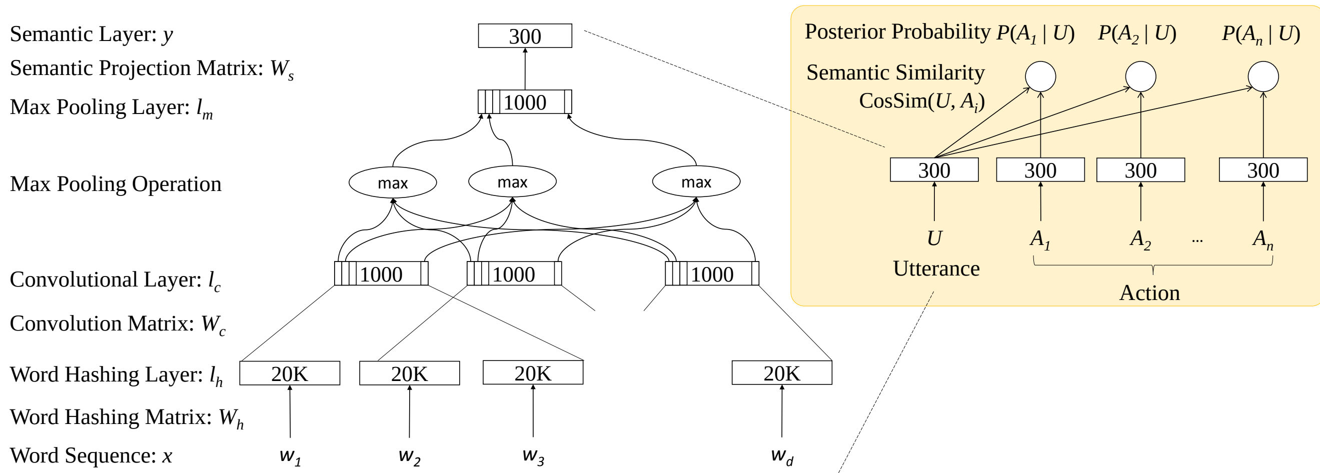

Fig. 3. Illustration of the CDSSM architecture for the predictive model.

where A' is an action candidate.

For training the model, we maximize the likelihood of the correctly associated actions given the utterances across the training set. The parameters of the model $\theta_1 = \{W_c, W_s\}$ is optimized by an objective:

$$\Lambda(\theta_1) = \log \prod_{(U,A^+)} P(A^+ \mid U). \tag{6}$$

The model is optimized using mini-batch stochastic gradient descent (SGD) [28]. Then we can transform the test utterances into the vector representations.

2.2.2. *Generative Model*

Similarly, we can estimate the posterior probability of an utterance given an action using the reversed setting,

$$P(U \mid A) = \frac{\exp(\mathrm{CosSim}(U, A))}{\sum_{U'} \exp(\mathrm{CosSim}(U', A))}, \tag{7}$$

which is the generative model that emits the utterances for each action. Also, the parameters of the model θ_2 is optimized by an objective:

$$\Lambda(\theta_2) = \log \prod_{(U^+,A)} P(U^+ \mid A). \tag{8}$$

The model can be obtained similarly and performs a reversed estimation for the relation between utterances and actions.

3. ADAPTATION

Practically the data for the target genre may be unavailable or insufficient to train CDSSM, so there may be a mismatch between the source and target genres. Based on the model trained on the source genre (θ_1 or θ_2), each utterance and action from the target genre can be transformed into a vector. Then it is possible that the embeddings of the target data cannot accurately estimate the score $\mathrm{CosSim}(U, A)$ due to the mismatch. Below we focus on adaptation approaches that adjust the embeddings generated by the source genre to fit the target genre, where two adaptation approaches are proposed.

3.1. Adapting CDSSM

Considering that the CDSSM is trained on the mismatched genre (human-machine genre), the CDSSM can be adapted by continually training the model using the data from the target genre (human-human genre) for several epochs (usually stop early before fully converged). Then the final CDSSM contains information about both genres, so it can be robust because of data from different genres and specific to the target genre.

3.2. Adapting Action Embeddings

Instead of adapting the whole CDSSM, this section applies an adaptation technique to directly learn adapted action embeddings that may be proper for the target genre. After converting actions and utterances from the target genre into vectors using CDSSM trained on the source genre, the idea here is to move the action embeddings based on the distribution of corresponding utterance embeddings, and then the adjusted action embeddings can fit to the target genre better. A similar idea was used to adapt embeddings based on the predefined ontology [30, 26].

Here we define Q as a set of action embeddings and R as a set of utterance embeddings obtained from the trained model (θ_1 or θ_2). Then we define two objectives, Φ_{act} and Φ_{utt}, to consider action and utterance embeddings respectively.

$$\Phi_{\text{act}}(\hat{Q}, \hat{R}) = \sum_{i=1}^{n} \left[\alpha_i \|\hat{q}_i - q_i\|^2 + \sum_{l(r_j)=i} \beta_{ij} \|\hat{q}_i - \hat{r_j}\|^2 \right],$$

$$\Phi_{\text{utt}}(\hat{R}) = \sum_{i:l(r_i)=1}^{n} \left[\alpha_i \|\hat{r_i} - r_i\|^2 + \sum_{l(r_j)=l(r_i)} \beta_{ij} \|\hat{r_i} - \hat{r_j}\|^2 \right],$$

where $q_i \in Q$ is the original action embeddings for the i-th action, $r_i \in R$ is the original utterance embeddings for the i-th utterance, and $l(\cdot)$ indicates the action label for an utterance. The idea here is to learn new action embeddings $\hat{q}_i$ that are close to q_i and the utterances labeled with the action i, $\hat{r_j}$. Also, Φ_{utt} suggests to learn new utterance embeddings $\hat{r_i}$ close to r_i and other utterances with the same action label. Here α and β control the relative strengths of associations. An objective $\Phi(\hat{Q}, \hat{R})$ combines them together:

$$\Phi(\hat{Q}, \hat{R}) = \Phi_{\text{act}}(\hat{Q}, \hat{R}) + \Phi_{\text{utt}}(\hat{R}). \tag{9}$$

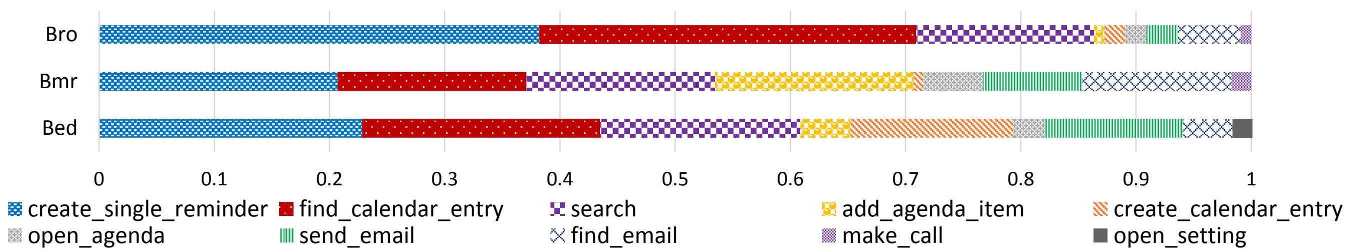

Fig. 4. Action distribution for different types of meetings.

With the integrated objective $\Phi(\hat{Q}, \hat{R})$, the sets of adapted action embeddings and adapted utterance embeddings ($\hat{Q}$ and $\hat{R}$ respectively) can be obtained simultaneously by an efficient iterative updating method [26, 31]. The updates for $\hat{q}_i$ and $\hat{r}_i$ are:

$$\Delta\hat{q}_i = \frac{\alpha q_i + \sum \beta_{ij}\hat{r_j}}{\alpha + \sum \beta}, \Delta\hat{r}_i = \frac{\alpha r_i + \sum \beta_{ij}\hat{r_j}}{\alpha + \sum \beta}. \tag{10}$$

Then the adapted action embeddings Q are obtained in order to estimate better scores for the target domain. Below we use the notation $\hat{y_A}$ to refer to the adapted action embeddings.

4. ACTIONABLE ITEM DETECTION

In order to predict the possible actions given utterances, for each utterance U, we transform it into a vector y_U, and then estimate the semantic similarity with vectors for all actions.

For the utterance U, the estimated semantic score of the k-th action is defined as:

$$\widehat{\text{CosSim}}(U, A_k) = \frac{y_U \cdot y_{\hat{A}_k}}{\|y_U\| \|y_{\hat{A}_k}\|}, \tag{11}$$

which is similar to (4), but replaces the original action embeddings y_A with the adapted embeddings $\hat{y_A}$. Note that the utterance embeddings are the original ones, so they can match the embeddings of test utterances.

The estimated semantic scores can be used in two ways [27]:

1. As final prediction scores: $\widehat{\text{CosSim}(U, A)}$ is directly treated as the prediction score of the actionable item detector.
2. As features of a classifier: $\widehat{\text{CosSim}(U, A)}$ is an input feature of a classifier and then a multi-class classifier can be trained as an actionable item detector. Then the trained classifier outputs the final prediction scores of actions given each test utterance for the detection task.

4.1. Unidirectional Estimation

With predictive and generative models from Section 2.2.1 and 2.2.2, here for the utterance U_i, we define the final prediction score of the action A_j using the predictive model as $S_P(i, j)$ and using the generative model as $S_G(i, j)$, where the prediction score can be obtained via above two ways.

4.2. Bidirectional Estimation

Considering that the estimation from two directions may model the similarity in different ways, we can incorporate the estimation from two directions by fusing the prediction scores, $S_P(i, j)$ and $S_G(i, j)$, to balance the effectiveness of predictive and generative models.

$$S_{\text{Bi}}(i, j) = \gamma \cdot S_P(i, j) + (1 - \gamma) \cdot S_G(i, j), \tag{12}$$

where γ is a weight to control the contributions from both sides.

5. EXPERIMENTS

5.1. Experimental Setup

The dataset is from the ICSI meeting corpus[2] [3], where 22 meetings previously used as test and dev sets are included for the actionable item detection task [32]. These include three types of meetings, Bed, Bmr, and Bro, which include regular project discussions between colleagues and conversations between students and their advisors[3]. The total numbers of utterances are 4544, 9227, and 7264 for Bed, Bmr, and Bro respectively.

Actionable items were manually annotated, where the annotation schema was designed based on the Microsoft Cortana conversational agent schema. There are in total 42 actions in Cortana data, and we identified 10 actions that are relevant to meeting scenarios: find_calendar_entry, create_calendar_entry, open_agenda, add_agenda_item, create_single_reminder, make_call, search, send_email, find_email, and open_setting[4]. There are total 318 utterances annotated with actionable items, which accounts for about 2% of all utterances. Fig. 4 shows actionable item distribution in the meeting corpus, where it can be found that different types of meetings contain slightly different distribution of actionable items, but some actions frequently occur in all meetings, such as create_single_reminder and find_calendar_entry.

Two meetings were annotated by two annotators, and we test the agreement for two settings using Cohen's Kappa coefficient [33]. First, the average agreement about whether an utterance includes an actionable item is 0.64; second, the average agreement about annotated actions (including others; total number of considered intents is 11) is 0.67, showing that the actionable items are consistent across persons.

5.2. Evaluation Metric

Due to imbalanced classes (number of non-actionable utterances is larger than number of actionable ones), the evaluation focuses on detection performance for each action. Here for each action, we use the area under the precision-recall curve (AUC) as the metric to evaluate whether the detector is able to effectively detect it for test

[2] `http://www.icsi.berkeley.edu/Speech/mr/`

[3] Bed (003, 006, 010, 012), Bmr (001, 005, 010, 014, 019, 022, 024, 028,030), Bro (004, 008, 011, 014, 018, 021, 024, 027)

[4] find_email and open_agenda do not occur in Cortana data.

Table 1. Actionable item detection performance on the average area of the precision-recall curve (AUC) (%).

Approach			#dim	Mismatch-CDSSM			Adapt-CDSSM			Match-CDSSM		
				$P(A\|U)$	$P(U\|A)$	Bidir	$P(A\|U)$	$P(U\|A)$	Bidir	$P(A\|U)$	$P(U\|A)$	Bidir
(a)	Sim ($\mathrm{CosSim}(U, A)$)			47.45	48.17	49.10	48.67	50.09	50.36	56.33	43.39	50.57
(b)	AdaptSim ($\widehat{\mathrm{CosSim}}(U, A)$)			54.00	53.89	**55.82**	59.46	56.96	**60.08**	64.19	60.36	**62.34**
(c)	SVM	Embeddings	300	53.07	48.07	55.71	60.06	59.03	63.95	64.33	65.58	**69.27**
(d)		(c) + Sim	311	52.80	54.95	59.09	60.78	60.29	65.08	64.52	64.81	68.86
(e)		(c) + AdaptSim	311	52.75	55.22	**59.23**	61.60	61.13	**65.71**	64.72	65.39	69.08

utterances. In the experiments, we report the average AUC scores over all classes (10 actions plus others).

5.3. CDSSM Training

To test the effect of CDSSM training data, we perform the experiments using the following models:

- Mismatch-CDSSM: a CDSSM trained on conversational agent data, which mismatches with the target genre.
- Adapt-CDSSM: a CDSSM pretrained on conversational agent data and then continually trained on meeting data.
- Match-CDSSM: a CDSSM trained on meeting data, which matches with the target genre.

The conversational agent data is collected by Microsoft Cortana, where there are about 1M utterances corresponding to more than 100 intents. For meeting data, we conduct the experiments on the manual transcripts. For all experiments, the total number of training iterations is set to 300, the dimension of the convolutional layer is 1000, and the dimension of the semantic layer is 300, where Adapt-CDSSM is trained on two datasets with 150 iterations for each.

5.4. Implementation Details

Considering that individuals may have consistent ways of referring to actionable items, to show the applicability of our approach to different speakers and meeting types, we take one of meeting types as training data and test on each of remaining two. Hence, we have 6 sets of experiments and report the average of AUC scores for evaluation, which is similar to 6-fold cross-validation. Note that the meeting data used in Match-CDSSM and Adapt-CDSSM is the training set of meeting data. The multi-class classifier we apply for actionable item detection in Section 4 is the SVM with RBF kernel using a default setting [34]. The parameters α and β in (9) are set to 1 to balance the effectiveness of original embeddings and the utterance embeddings with the same action. The parameter γ in (12) is set as 0.5 to allow predictive and generative models contribute equally.

6. EVALUATION RESULTS

Experimental results with different CDSSMs are shown in Table 1. Rows (a) and (b) use the semantic similarity as final prediction scores, where Sim (row (a)) uses $\mathrm{CosSim}(U_i, A_j)$ and AdaptSim (row (b)) uses $\widehat{\mathrm{CosSim}}(U_i, A_j)$ as $S_P(i, j)$ or $S_G(i, j)$. Rows (c)-(e) use the similarity as features and then train an SVM to estimate the final prediction scores, where row (c) takes utterance embedding vectors as features, and rows (d) and (e) include the semantic similarity as additional features for the classifier. Hence the dimension of features is 311, including 300 values of utterance embeddings and 11 similarity scores for all actions.

When we treat the semantic similarity as final prediction scores, adapted embeddings (AdaptSim) perform better, achieving 55.82%, 60.08%, and 62.34% for Mismatch-CDSSM, Adapt-CDSSM, and Match-CDSSM respectively. Considering that the learned embeddings do not fit the target genre well, the similarity treated as features of a classifier can be combined with other features to automatically adapt the reliability of the similarity features. Row (c) shows the performance using only utterance embeddings, and including the similarity scores as additional features can improve the performance for Mismatch-CDSSM (from 55.71% to 59.09%) and Adapt-CDSSM (from 63.95% to 65.08%). The action embedding adaptation further adjusts embeddings to the target genre based on Section 3.2, and row (c) shows that the performance can be further improved (59.23% and 65.71%). Below we discuss the results in different aspects.

6.1. Comparing Different CDSSM Training Data

Because the target genre is not always available or not enough for training CDSSM, we compare the results using CDSSM trained on different data. From Table 1, model adaptation (Adapt-CDSSM) improves the performance of Mismatch-CDSSM in all cases, showing that the embeddings pre-trained on the mismatched data are successfully adapted to the target genre and then resulting in better performance. Although Adapt-CDSSM takes more data than Match-CDSSM, Match-CDSSM performs better. However, for row (a), we can see that Match-CDSSM is not robust enough, because generative model ($P(U \mid A)$) performs 43.39% on AUC, even worse than Mismatch-CDSSM. It shows that the bidirectional model, the embedding adaptation, and additional classifier help improve the robustness so that Match-CDSSM achieve better performance compared to Adapt-CDSSM.

The best result from the matched features is one using only embeddings features (69.27% in row (c)), and the possible reason is that the embeddings fit well to the target genre, so adding similarity cannot provide additional information to improve the performance.

6.2. Effectiveness of Bidirectional Estimation

From Table 1, it is shown that all results from the bidirectional estimation significantly outperform the results using unidirectional estimation across all CDSSMs and all methods except for rows (a) and (b) from Match-CDSSM. Comparing between the predictive model ($P(A \mid U)$) and the generative model ($P(U \mid A)$), the performance is similar and does not show that a certain direction is better in most cases. The improvement of bidirectional estimation suggests that the predictive model and the generative model can compensate each other, and then provide more robust estimated scores.

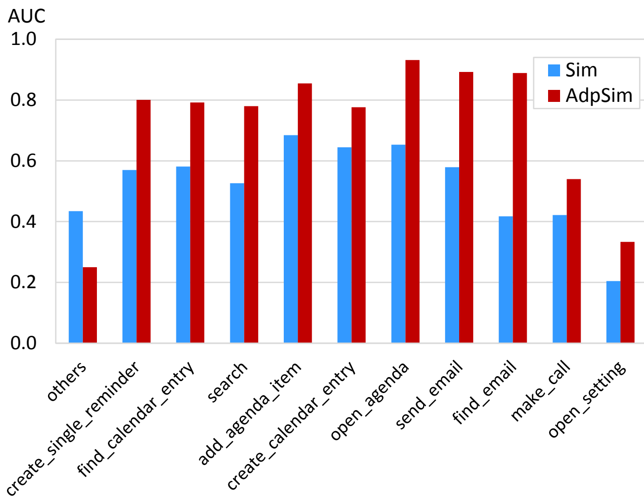

Fig. 5. The average AUC distribution over all actions in the training set before and after action embedding adaptation using Match-CDSSM.

6.3. Effectiveness of Adaptation Techniques

Two adaptation approaches, CDSSM adaptation and action embedding adaptation, are useful when the features do not perfectly fit to the target genre. When without SVM, model adaptation and action embedding adaptation improve the performance from 49.10% to 50.36% and to 55.82% respectively. Applying both adaptation techniques achieve 60.08% on average AUC. After we use the similarity scores as additional features of SVM, using individual adaptation improves the performance, and applying both techniques achieves further improvement. Therefore, it is shown that the proposed CDSSM and adaptation approaches can be applied when the data for the target genre is unavailable or scarce.

On the other hand, when using the matched data for CDSSM training (Match-CDSSM), action embedding adaptation still improves the performance before SVM (from 50.57% to 62.34%). Fig. 5 shows the performance distribution over all actions in the training set before and after action embedding adaptation, where we find that all AUC scores are increased except for others so that overall performance is improved. The reason why matched data cannot induce good enough embeddings is that there are much more utterances belonging to others in the meetings, so CDSSM is more sensitive to the action others due to data imbalance. However, the adaptation adjusts all action embeddings equally, forcing to increase the reliability of other action embeddings. Therefore, although the adapted result of others drops, the performance of all other actions is improved, resulting in better overall performance.

6.4. Effectiveness of CDSSM

To evaluate whether CDSSM provides better features for actionable item detection, we compare the performance with three baselines trained on the meeting corpus using the same setting:

- AdaBoost with ngram features
  A boosting classifier is trained using unigram, bigram and trigram features [35].
- SVM with ngram features
  An SVM classifier is trained using unigram, bigram and trigram features [34].
- SVM with doc2vec embeddings
  An SVM classifier is trained using paragraph vectors[5] [36], where the training set of paragraph vectors is the same as CDSSM takes, the vector dimension is set to 300, and the window size is 3.

Table 2. Actionable item detection performance on the area of the precision-recall curve (AUC) (%).

Approach			AUC
Baseline	AdaBoost	ngram	54.31
	SVM	ngram	52.84
	SVM	doc2vec	59.79
Proposed	SVM	CDSSM: $P(A\|U)$	64.33
	SVM	CDSSM: $P(U\|A)$	65.58
	SVM	CDSSM: Bidirectional	**69.27**

First two baselines use lexical features while the third one uses semantic features. Table 2 shows that two lexical baselines perform similarly, and AdaBoost is slightly better than SVM. Semantic embeddings trained on the meeting data as features perform better than lexical features, where doc2vec obtains 59.79% on AUC [36]. For the proposed approaches, both unidirectional CDSSMs outperform three baselines, achieving 64.33% for the predictive model and 65.58% for the generative model. In addition, bidirectional CDSSM improves the performance to 69.27%, showing a promising result and proving the effectiveness of CDSSM features.

6.5. Discussion

In addition to the power of CDSSM features, another advantage of CDSSM is the ability of generating more flexible action embeddings. For example, the actions open_agenda and find_email in the meeting data do not have the corresponding predefined intents in the Cortana data; however, CDSSM is still able to generate the action embeddings for find_email by incorporating the semantics from find_message and send_email. The flexibility may fill the gap between mismatched annotations. In the future work, we plan to investigate the ability of generating unseen action embeddings in order to remove the domain constraint for practical usage.

7. CONCLUSION

This paper focuses on the task of actionable item detection in meetings, where a convolutional deep structured semantic model (CDSSM) is applied to learn both utterance and action embeddings. Then the latent semantic features generated by CDSSM show the effectiveness of detecting actions in meetings compared to lexical features, and also outperform semantic paragraph vectors. The adaptation techniques are proposed to adjust the learned embeddings to fit the target genre when the source genre does not match well with target genre, showing significant improvements in detecting actionable items. The paper highlights a future research direction by releasing an annotated dataset and the trained embeddings for actionable item detection.

[5] https://radimrehurek.com/gensim/index.html

8. REFERENCES

[1] Barry Brown, Moira McGregor, and Donald McMillan, "Searchable objects: Search in everyday conversation," in *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. ACM, 2015, pp. 508–517.

[2] Donald McMillan, Antoine Loriette, and Barry Brown, "Repurposing conversation: Experiments with the continuous speech stream," in *Proceedings of the 33rd annual ACM conference on Human factors in computing systems*. ACM, 2015, pp. 3953–3962.

[3] Adam Janin, Don Baron, Jane Edwards, Dan Ellis, Davis Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, and Chuck Wooters, "The ICSI meeting corpus," in *Proceedings of 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2003.

[4] Gokhan Tur, Andreas Stolcke, L Lynn Voss, John Dowding, Benoît Favre, Raquel Fernández, Matthew Frampton, Michael W Frandsen, Clint Frederickson, Martin Graciarena, et al., "The CALO meeting speech recognition and understanding system.," in *SLT*, 2008, pp. 69–72.

[5] Srinivas Bangalore, Giuseppe Di Fabbrizio, and Amanda Stent, "Learning the structure of task-driven human–human dialogs," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 7, pp. 1249–1259, 2008.

[6] Yun-Nung Chen, William Yang Wang, and Alexander I. Rudnicky, "Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 2013, pp. 120–125.

[7] Yun-Nung Chen, William Yang Wang, Anatole Gershman, and Alexander I. Rudnicky, "Matrix factorization with knowledge graph propagation for unsupervised spoken language understanding," in *Proceedings of The 53rd Annual Meeting of the Association for Computational Linguistics and The 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*. ACL, 2015, pp. 483–494.

[8] Yun-Nung Chen, William Yang Wang, and Alexander I. Rudnicky, "Jointly modeling inter-slot relations by random walk on knowledge graphs for unsupervised spoken language understanding," in *Proceedings of 2015 Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies*. ACL, 2015, pp. 619–629.

[9] Yun-Nung Chen, Ming Sun, Alexander I. Rudnicky, and Anatole Gershman, "Leveraging behavioral patterns of mobile applications for personalized spoken language understanding," in *Proceedings of The 17th ACM International Conference on Multimodel Interaction*. ACM, 2015.

[10] Trung H Bui and Stanley Peters, "Decision detection using hierarchical graphical models," in *Proceedings of the ACL 2010 Conference Short Papers*. Association for Computational Linguistics, 2010, pp. 307–312.

[11] Raquel Fernández, Matthew Frampton, Patrick Ehlen, Matthew Purver, and Stanley Peters, "Modelling and detecting decisions in multi-party dialogue," in *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*. Association for Computational Linguistics, 2008, pp. 156–163.

[12] Fan Yang, Gokhan Tur, and Elizabeth Shriberg, "Exploiting dialogue act tagging and prosodic information for action item identification," in *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2008, pp. 4941–4944.

[13] Michel Galley, Kathleen McKeown, Julia Hirschberg, and Elizabeth Shriberg, "Identifying agreement and disagreement in conversational speech: Use of bayesian networks to model pragmatic dependencies," in *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2004, p. 669.

[14] Dustin Hillard, Mari Ostendorf, and Elizabeth Shriberg, "Detection of agreement vs. disagreement in meetings: Training with unlabeled data," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers-Volume 2*. Association for Computational Linguistics, 2003, pp. 34–36.

[15] Korbinian Riedhammer, Benoit Favre, and Dilek Hakkani-Tür, "Long story short–global unsupervised models for keyphrase based meeting summarization," *Speech Communication*, vol. 52, no. 10, pp. 801–815, 2010.

[16] Shasha Xie, Dilek Hakkani-Tür, Benoit Favre, and Yang Liu, "Integrating prosodic features in extractive meeting summarization," in *IEEE Workshop on Automatic Speech Recognition & Understanding*. IEEE, 2009, pp. 387–391.

[17] Yun-Nung Chen and Florian Metze, "Multi-layer mutually reinforced random walk with hidden parameters for improved multi-party meeting summarization," in *Proceedings of 14th Annual Conference of the International Speech Communication Association*. 2013, ISCA.

[18] Wen-tau Yih, Xiaodong He, and Christopher Meek, "Semantic parsing for single-relation question answering," in *Proceedings of ACL*, 2014.

[19] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao, "Semantic parsing via staged query graph generation: Question answering with knowledge base," in *Proceedings of the Joint Conference of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing of the AFNLP*. 2015, ACL Association for Computational Linguistics.

[20] Yonatan Belinkov, Mitra Mohtarami, Scott Cyphers, and James Glass, "Vectorslu: A continuous word vector approach to answer selection in community question answering systems," in *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval*, 2015, vol. 15.

[21] Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, Li Deng, and Yelong Shen, "Modeling interestingness with deep neural networks," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2014.

[22] Yun-Nung Chen, William Yang Wang, and Alexander I Rudnicky, "Leveraging frame semantics and distributional semantics for unsupervised semantic slot induction in spoken dialogue systems," in *2014 IEEE Spoken Language Technology Workshop*. IEEE, 2014, pp. 584–589.

[23] Yun-Nung Chen and Alexander Rudnicky, "Dynamically supporting unexplored domains in conversational interactions by enriching semantics with neural word embeddings," in *2014 IEEE Spoken Language Technology Workshop*. IEEE, 2014, pp. 590–595.

[24] Yun-Nung Chen, William Yang Wang, and Alexander I. Rudnicky, "Learning semantic hierarchy with distributional representations for unsupervised spoken language understanding," in *Proceedings of The 16th Annual Conference of the International Speech Communication Association*. ISCA, 2015, pp. 1869–1873.

[25] Manaal Faruqui and Chris Dyer, "Improving vector space word representations using multilingual correlation," in *Proceedings of EACL*, 2014.

[26] Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith, "Retrofitting word vectors to semantic lexicons," in *Proceedings of NAACL*, 2015.

[27] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Gregoire Mesnil, "A latent semantic model with convolutional-pooling structure for information retrieval," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 2014, pp. 101–110.

[28] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck, "Learning deep structured semantic models for web search using clickthrough data," in *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, 2013, pp. 2333–2338.

[29] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil, "Learning semantic representations using convolutional neural networks for web search," in *Proceedings of the companion publication of the 23rd international conference on World wide web companion*. International World Wide Web Conferences Steering Committee, 2014, pp. 373–374.

[30] Mo Yu and Mark Dredze, "Improving lexical embeddings with semantic knowledge," in *Proceedings of The 52nd Annual Meeting of the Association for Computational Linguistics*. ACL, 2014, pp. 545–550.

[31] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux, "Label propagation and quadratic criterion," *Semi-supervised learning*, vol. 10, 2006.

[32] Jeremy Ang, Yang Liu, and Elizabeth Shriberg, "Automatic dialog act segmentation and classification in multiparty meetings.," in *Proceedings of 2005 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2005, pp. 1061–1064.

[33] Jacob Cohen et al., "A coefficient of agreement for nominal scales," *Educational and psychological measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[34] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, pp. 27, 2011.

[35] Benoit Favre, Dilek Hakkani-Tür, and Sebastien Cuendet, "ICSIBoost," `http://code.google.come/p/icsiboost`, 2007.

[36] Quoc Le and Tomas Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning*, 2014, pp. 1188–1196.