Towards Zero-Shot Frame Semantic Parsing for Domain Scaling

Ankur Bapna, Gokhan Tür, Dilek Hakkani-Tür, Larry Heck

Google Research, Mountain View

{ankurbpn,gokhant,dilekh,larryheck}@google.com

Abstract

State-of-the-art slot filling models for goal-oriented human/machine conversational language understanding systems rely on deep learning methods. While multi-task training of such models alleviates the need for large in-domain annotated datasets, bootstrapping a semantic parsing model for a new domain using only the semantic frame, such as the back-end API or knowledge graph schema, is still one of the holy grail tasks of language understanding for dialogue systems. This paper proposes a deep learning based approach that can utilize *only the slot description* in context without the need for any labeled or unlabeled in-domain examples, to quickly bootstrap a new domain. The main idea of this paper is to leverage the encoding of the slot names and descriptions within a multi-task deep learned slot filling model, to implicitly align slots across domains. The proposed approach is promising for solving the domain scaling problem and eliminating the need for any manually annotated data or explicit schema alignment. Furthermore, our experiments on multiple domains show that this approach results in significantly better slot-filling performance when compared to using only in-domain data, especially in the low data regime.

Index Terms: slot-filling, deep learning, multi-task RNNs, domain adaptation, dialogue systems

1. Introduction

In traditional goal-oriented dialogue systems, user utterances are typically understood in terms of hand-designed semantic frames comprised of domains, intents and slots [1]. Understanding the user utterance involves (i) detecting the domain of the utterance, (ii) classifying the intent of the utterance based on the semantic frame corresponding to the detected domain and (iii) identifying the values or sequence of tokens corresponding to each slot in the semantic frame. An example semantic frame is shown in Figure 1 for a flight related query: *find flights to new york tomorrow*.

Most modern approaches for conversational language understanding involve training machine learning models on annotated training data [2, 3, 4, among others]. Deep learning models typically outperform most other approaches in the domain of large scale supervised learning and this has been shown to be the case for spoken language understanding [5, 6, 7, 8, among others]. However, despite recent advancements and tremendous research activity in semi-supervised and unsupervised learning, these models still require massive amounts of labeled data to train.

In recent years, motivated by commercial applications like Apple Siri, Microsoft Cortana, Amazon Alexa, or Google Assistant, there is significant interest in enabling users to add more functionality and power to their respective assistants. However, although these assistants can handle queries corresponding to certain narrow domains with high reliability, the ability to understand user queries across a wide range of domains, in a robust manner, is still missing. This is a significant bottleneck that restricts the ability to crowd-source the addition of new actions or skills to these assistants.

W	find	flights	to	new	york	tomorrow
	↓	↓	↓	↓	↓	↓
S	O	O	O	B-Dest	I-Dest	B-Date
D	flight					
I	find_flight					

Figure 1: *An example semantic parse of an utterance (W) with slot (S), domain (D), intent (I) annotations, following the IOB (in-out-begin) representation for slot values.*

With recent advances in deep learning, there is renewed excitement around latent semantic representations which can be trained for a multitude of covered domains via transfer learning. An earlier work proposed training a single multi-task deep learning model covering all domains, providing implicit shared feature learning across domains [6]. The approach showed significantly better overall semantic template level performance. Similar experiments on using shared feature extraction layers for slot-filling across several domains have demonstrated significant performance improvements relative to single-domain baselines, especially in low data regimes [9].

In this study we explore semi-supervised slot-filling based on deep learning based approaches that can utilize natural language *slot label descriptions*. This alleviates the need for large amounts of labeled or unlabeled in-domain examples or explicit schema alignment, enabling developers to quickly bootstrap new domains. Similar ideas have previously been shown to work for domain classification, where domain names were leveraged to generate representations in a shared space with query representations [10]. Applying this idea to a full semantic frame is much more complex and requires building a general slot or concept tagger over a large set of domains.

The architecture for the general concept tagger is similar to those proposed in recent Question Answering (QA) and Machine Reading literature. We build upon the idea of using a learned question encoding and using it for extractive question answering from input passages [11, 12, 13]. However, our proposed use case involves using slot encodings learned from their natural language descriptions and training the model on a multitude of small in-domain datasets to generalize to new domains, instead of training and evaluating on larger open-domain QA datasets.

Through our experiments we demonstrate that our model learns to identify slots across domains, from small amounts of training data, without the need for any explicit schema alignments. Such an approach can significantly alleviate the domain scaling problem and reduce the need for additional manually annotated data when bringing up a new domain.

In Section 2 we describe the task and the dataset, followed by descriptions of the baseline model, the multi-task model and the concept tagger in Section 3. This is followed by experimental results in Section 4 and discussion in Section 5.

Table 1: *Sample utterances from each domain*

Domain	Sample	# Samples
bus_tickets	I need 2 adult and 6 senior bus tickets from St . Petersburg to Concord.	500
book_room	I need a hotel room for 5 guests to check-in next Friday	500
flights_1	book a flight to logan airport 3 / 23 to jan 2	10000
flights_2	Search for flights to Philly one - way with promo code 54ZFHK33	500
fare	How much is it on Lyft to go from Saratoga to Fremont	1000
find_restaurants	chinese places to eat that are not expensive	1000
appointments	set up a patient follow - up with ProHealth Chiropractic	2000
reserve_restaurant	I need a table at Sun Penang on December 24th	5000
book_cab	book a ride to 9192 johnson street with uber for 6 passengers	2000
book_hotel	book me a hotel room in Cincinnati that costs less than $300	1000

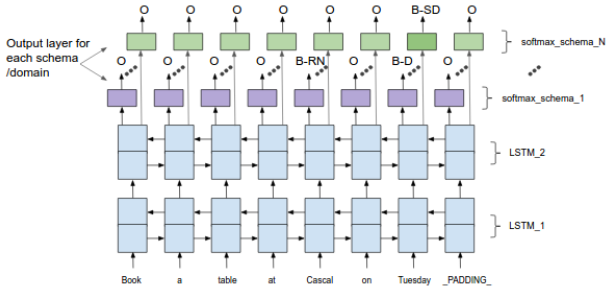

Figure 2: *Multi-task stacked LSTM architecture*

2. Slot Filling

In most spoken dialogue systems, the semantic structure of an application domain is defined in terms of *semantic frames*. Each semantic frame contains several typed components called "*slots*." For the example in Figure 1, the domain *Flights* may contain slots like *Departure_City*, *Arrival_City*, *Departure_Date*, *Airline_Name*, etc. The task of slot filling is then to instantiate slots in semantic frames from a given user query or utterance.

More formally, the task is to estimate the sequence of tags $Y = y_1, ..., y_n$ in the form of IOB labels as in [14] (with 3 outputs corresponding to 'B', 'I' and 'O'), and as shown in Figure 1 corresponding to an input sequence of tokens $X = x_1, ..., x_n$.

In literature, researchers usually employ known sequence labeling methods for filling frame slots of an application domain using a labeled training data set. With advances in deep learning, the best performing academic slot filling systems rely on recurrent neural network (RNN) or Long Short Term Memory (LSTM) based models. RNNs were first used for slot filling by Yao *et al.* [15] and Mesnil *et al.* [16]. A comprehensive compilation of RNN based slot filling approaches was described by Mesnil *et al.* [5]. State-of-the-art slot filling methods usually rely on bidirectional LSTM models [6, 5, 7, 17, 18, among others]. Extensions include encoder-decoder models[19, 20, among others] or memory networks [21].

For this study we crowd sourced natural language text data for 10 domains. The schema corresponding to each of these domains is described in Table 2. One noticeable feature of our datasets is a lack of fine-grained slot types as compared to the popular ATIS dataset [22] which contains over a hundred distinct slots.

For the collection, a list of possible slot-value combinations was generated from the Google knowledge graph manually, and used to prompt crowd-workers with slot-value pairs. The crowd-workers were instructed to ask their digital assistant to complete certain tasks with given slot-value based arguments. The collected utterances were then either automatically labeled if a verbatim match was found for the slot-values, or sent out to a second set of raters for labeling. For the labeling job the crowd workers were instructed to label spans corresponding to slot values in the instantiated samples.

Table 1 shows the list of these domains with representative example queries and the total number of training samples available. Test sets were constructed using the same framework. All the collected data was tokenized using a standard tokenizer and lower-cased before use since capitalization was seen to be indicative of slot values. All digits were replaced with special "#" tokens following [9].

3. Models

In this study we explore the idea of zero-shot slot-filling, by implicitly linking slot representations across domains by using the label descriptions of the slots. We compare the performance of three model architectures on varying amounts of training data:

- Single task bi-directional LSTM
- Multi-task bi-directional stacked LSTM model[6, 9]
- Concept tagging model using slot label descriptions

For all our experiments we use 200 dimensional word2vec embeddings trained on the GNews corpus [23]. Tokens not present in the pre-trained embeddings were replaced by a _OOV_ token. Each model was trained for 50000 steps using the RMSProp optimizer and tuned on the dev set performance before evaluation on the test set.

For evaluation, we compute the token F1 for each slot independently and report the weighted average over all slots for the target domain. We use token F1 instead of the traditional slot F1 since token level evaluation results in softer penalization for mistakes, to mitigate span inconsistencies in the crowd sourced labels.

3.1. Baseline single task model

We use a single domain bidirectional LSTM as our baseline model. The model consists of the embedding layer followed by a 128 dimensional (64 dimensions in each direction) bidirectional LSTM. This is followed by a softmax layer that acts on the LSTM state for every token to predict the IOB label corresponding to the token.

3.2. Multi-task model

The multi-task model consists of 2 stacked bidirectional LSTM layers with 256 dimensions each (128 dimensions in each direction). Both LSTM layers are shared across all domains, fol-

Table 2: *Slot schema / descriptions used for the concept tagger for each domain*

Domain	Slot descriptions
bus_tickets	departure time, number of adult passengers, arrival location, number of child passengers, number of senior passengers, departure location, promotion code, date of departure, trip type, discount type, date of return
book_room	features, property type, number of beds, number of guests, maximum price per day, location, check out date, room type, check in date
flights_1	flight class, date of second departure, number of passengers, second from location, flight type, from location, search type, departure date, second to location, to location, non stop, return date
flights_2	origin, # seniors, departure date, # adults, destination, return date, price type, promotion code, trip type
fare	origin, destination, transit operator
find_restaurants	amenities, hours, neighborhood, cuisine, price range
appointments	services, appointment time, appointment date, title
reserve_restaurant	number of people, restaurant name,reservation date, location, cuisine, restaurant distance, reservation time, meal, price range, rating
book_cab	pickup location, drop off location, # passengers, cab operator, type of cab, departure time
book_hotel	amenities, departure date, arrival date, price range, # rooms, hotel name, ratings, room type, location, duration of stay

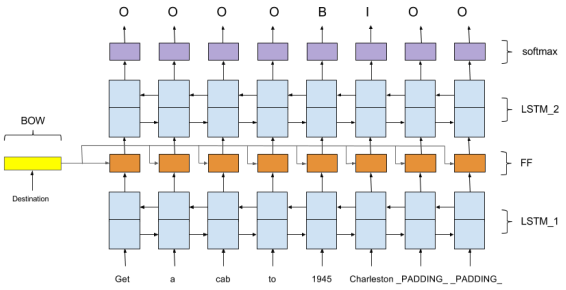

Figure 3: *Zero shot Concept Tagger architecture*

lowed by domain specific softmax layers, following [9]. The model was trained using a batch size of 100 with alternating batches from different domains. To avoid over-training the model on the larger domains, the number of batches chosen from each domain was proportional to the logarithm of the number of training samples from the domain. The conceptual model architecture is depicted in Figure 2.

3.3. Zero-Shot Concept Tagging Model

The main idea behind the zero-shot concept tagger is to leverage the slot names or descriptions in a domain-agnostic slot tagging model. Assuming that the slot description is semantically accurate, if one of the already covered domains contains a similar slot, a continuous representation of the slot obtained from shared pre-trained embeddings can be leveraged in a domain-agnostic model. An obvious example would be adding United Airlines when the multi-task model can already parse queries for American Airlines and Turkish Airlines. While the slot names may be different, the concept of *departure city* or *arrival city* should persist and can be transferred to the new task of United Airlines using their natural language descriptions. The very same idea can hold when the new domain is related but not *flights*, but, say, *ground transportation*. Similarly, domain independent slots such as location or date/time expressions can be implicitly shared for two domains like *hotel reservation* or *restaurant reservation*.

In order to incorporate this slot description knowledge into model training, we first generate an encoding of the slot by combining the token embeddings for the slot description. In principle this encoding can be obtained by passing description token embeddings through a RNN, but for the current experiments we just use their average. These slot representations are then combined within the multi-task architecture to obtain a domain-agnostic slot tagging model.

To elaborate, the zero-shot concept tagger consists of a single 256 dimensional bidirectional LSTM layer that acts on a sequence of tokens to produce contextual representations for each token in the utterance. This is followed by a feed forward layer where the contextual token representations are combined with the slot encoding to produce vectors of 128 dimensions. This feeds into another 128 dimensional bi-directional LSTM layer followed by a softmax layer that outputs the prediction for that slot. In our experiments these predictions are made independently for each slot by feeding a single slot description, but it is possible to slightly alter the architecture to make predictions for all slots. The input samples during training and evaluation for each slot included both positive (where the slot was present) and negative samples (where it was absent). The ratio of training samples from a particular domain in a batch was proportional to the logarithm of the number of training samples. The conceptual model architecture is depicted in Figure 3.

4. Experiments and Results

We compare the performances of the models on varying amounts of training data from each domain. This involves using all available out of domain data and varying the amount of training data for the target domain. To avoid performance variations due to the small sample sizes, the performance was averaged over 10 runs with training samples drawn from different parts of the domain dataset.

For every domain, 20% of the training examples were set aside for the dev set. Since we were evaluating with varying amounts of training data for each domain, the dev set was different for each data-point, corresponding to the bottom 20% of the training samples. For example, if the training set consisted of 100 samples from Domain A and 20 samples from Domain B, dev set A would consist of 20 samples from Domain A and dev set B would be comprised of 4 samples from Domain B. The performance on these dev sets was evaluated separately and averaged weighted by the log of the number of training samples. This weighted average was used to tune the model hyper-parameters. We used a logarithmic combination since it struck a good balance between noisy evaluations on domains with small dev sets

Table 3: *Weighted token F1 scores at various points on the learning curve for the compared models. ST corresponds to the single task baseline, MT corresponds to the multi task baseline and CT corresponds to the general concept tagging model.*

# target train samples	0	5			20			100			1000		
Domain	CT	ST	MT	CT	ST	MT	CT	ST	MT	CT	ST	MT	CT
book_room	**0.48**	0.09	0.40	**0.49**	0.28	0.53	**0.56**	0.50	0.61	**0.65**	-	-	-
bus_tickets	**0.45**	0.11	0.40	**0.63**	0.34	0.63	**0.72**	0.60	0.74	**0.78**	-	-	-
flights_1	**0.19**	0.19	0.41	**0.42**	0.46	0.59	**0.63**	0.66	0.71	**0.75**	0.77	0.79	**0.81**
flights_2	**0.45**	0.04	0.33	**0.56**	0.33	0.59	**0.69**	0.58	0.66	**0.73**	-	-	-
fare	**0.04**	0.53	**0.77**	0.71	0.84	0.90	**0.92**	0.93	0.94	**0.95**	**0.98**	**0.98**	**0.98**
book_hotel	**0.45**	0.08	0.31	**0.49**	0.38	0.61	**0.64**	0.69	**0.80**	0.78	0.88	**0.90**	**0.90**
find_restaurants	**0.35**	0.19	0.37	**0.52**	0.44	0.60	**0.62**	0.69	**0.75**	0.74	0.82	**0.84**	**0.84**
appointments	**0.56**	0.24	0.55	**0.63**	0.46	0.65	**0.67**	0.65	0.72	**0.73**	0.78	0.79	**0.80**
reserve_restaurant	**0.56**	0.20	0.50	**0.64**	0.50	0.68	**0.70**	0.72	**0.79**	0.78	0.82	0.85	**0.86**
book_cab	**0.18**	0.46	0.63	**0.65**	0.56	0.72	**0.74**	0.78	0.82	**0.84**	0.89	0.90	**0.92**

Table 4: *Comparison of slot-wise performances of the concept tagging model (CT) and the multi-task model (MT) on "appointment time" from appointments, "pickup location" from book_cab and "# seniors" from "flights_2".*

# train samples		0	5	100	500
appointment time	CT	**0.84**	**0.85**	**0.88**	**0.89**
	MT	-	0.66	0.87	**0.89**
pickup location	CT	0.05	0.08	0.39	0.51
	MT	-	**0.21**	**0.47**	**0.56**
# seniors	CT	**0.26**	**0.45**	**0.70**	**0.75**
	MT	-	0.11	0.38	0.53

and over-tuning to the domains with larger dev sets.

The performances along the learning curve for all the models on the 10 domains are described in Table 3. When no in-domain data is available, the concept tagging model is able to achieve reasonable bootstrap performance for most domains. Even when more data becomes available the model beats the single task model by significant margins and performs better than or on par with the multi-task baseline for most points on the learning curves.

5. Discussion

To better understand the strengths and weaknesses of the concept tagger we analyze its performance on individual slots. The model performs better than or on par with the multi-task model for most slots, with significant performance gains on slots that have shared semantics with slots in other domains. For slots that are specific to particular domains, like *discount type* from *bus_tickets*, the concept tagger usually needs a larger number of training samples to reach the same level of performance. This can be explained by a lack of slot-specific parameters within the model.

Table 4 compares the performance of the concept tagger and the multi-task model on three slots across different domains that illustrate the strengths and weaknesses of our approach. By leveraging shared features and semantics with *departure time*, *reservation time* and time related slots from other domains the concept tagger is able to reach within 10% of the peak performance on *appointment time* without the need for any in-domain training data. Similarly, the model is able to ramp up performance on *# seniors* with a small amount of in-domain data, despite the presence of a competing slot with similar semantics (*# adults*) within the same domain. This highlights the model's ability to generalize from slots with similar descriptions and semantics across domains.

On the other hand, the concept tagger's performance is worse than our multi-task baseline on *pickup location*. A lack of a good contextual representations for the description *pickup location* and the presence of a competing slot, *dropoff location*, might be responsible for the performance degradation observed for this slot. This highlights the concept tagger's susceptibility to descriptions that fail to produce a compatible slot representation, either due to an incomplete or misleading description of the slot semantics or a lack of good embedding representations for these descriptions. It might be possible to alleviate poor slot representations by fine tuning the slot representations on small amounts of in-domain training data after starting with representations derived from pre-trained word embeddings or using contextual word embeddings [24]. Enhancing utterance token representations with an entity linker or a knowledge base are possible extensions of this work that might enable better generalization to new entities. Exploring use of unlabeled training data from target domains with a domain adversarial loss [25] might be another interesting avenue for exploration.

The appeal for our approach derives from its simplicity and the minimal amount of supervision required to bring up slot-filling on a new domain. The proposed solution makes it possible to design a plug and play system with a reduced need for expensive labeled data for every additional domain.

6. Conclusions

In this paper we propose a novel approach to slot filling that leverages shared feature extraction and slot representations across domains by using the natural language descriptions of slots. We crowd-sourced slot filling datasets for ten domains to explore approaches that can easily scale across domains and demonstrate that our proposed concept tagging model performs significantly better than a strong multi-task baseline, especially in the low data regime. To further evaluate the strengths and weaknesses of the proposed approach, we analyze its performance on individual slots and demonstrate that our model is able to leverage shared features and semantic descriptions of slots defined in other domains, and shows potential for reasonable performance on slots in a new domain without the need for any in-domain training data or explicit schema alignment.

We hope that our proposed solution can provide a baseline approach for future research into scalable frame semantic parsing systems.

7. References

[1] G. Tur and R. D. Mori, Eds., *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*. New York, NY: John Wiley and Sons, 2011.

[2] S. Young, "Talking to machines (statistically speaking)," in *Proceedings of the ICSLP*, Denver, CO, September 2002.

[3] S. Hahn, M. Dinarelli, C. Raymond, F. Lefevre, P. Lehnen, R. D. Mori, A. Moschitti, H. Ney, and G. Riccardi, "Comparing stochastic approaches to spoken language understanding in multiple languages," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 6, pp. 1569–1583, 2011.

[4] Y.-Y. Wang, L. Deng, and A. Acero, "Spoken language understanding - an introduction to the statistical framework," *IEEE Signal Processing Magazine*, vol. 22, no. 5, pp. 16–31, September 2005.

[5] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tür, X. He, L. Heck, G. Tur, and D. Yu, "Using recurrent neural networks for slot filling in spoken language understanding," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 530–539, 2015.

[6] D. Hakkani-Tür, G. Tur, A. Celikyilmaz, Y.-N. Chen, J. Gao, L. Deng, and Y.-Y. Wang, "Multi-domain joint semantic frame parsing using bi-directional RNN-LSTM," in *Proceedings of the Interspeech*, San Francisco, CA, 2016.

[7] G. Kurata, B. Xiang, B. Zhou, and M. Yu, "Leveraging sentence-level information with encoder LSTM for semantic slot filling," in *Proceedings of the EMNLP*, Austin, TX, 2016.

[8] R. Sarikaya, G. E. Hinton, and B. Ramabhadran, "Deep belief nets for natural language call-routing," in *Proceedings of the ICASSP*, Prague, Czech Republic, 2011.

[9] A. Jaech, L. Heck, and M. Ostendorf, "Domain adaptation of recurrent neural networks for natural language understanding," in *Proceedings of the Interspeech*, San Francisco, CA, 2016.

[10] Y. Dauphin, G. Tur, D. Hakkani-Tür, and L. Heck, "Zero-shot learning and clustering for semantic utterance classification," in *Proceedings of the ICLR*, 2014.

[11] "Dataset and neural recurrent sequence labeling model for open-domain factoid question answering," *CoRR*, vol. abs/1607.06275, 2016, withdrawn. [Online]. Available: http://arxiv.org/abs/1607.06275

[12] K. Lee, T. Kwiatkowski, A. Parikh, and D. Das, "Learning recurrent span representations for extractive question answering," *arXiv preprint arXiv:1611.01436*, 2016.

[13] K. M. Hermann, T. Kociský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," *CoRR*, vol. abs/1506.03340, 2015. [Online]. Available: http://arxiv.org/abs/1506.03340

[14] C. Raymond and G. Riccardi, "Generative and discriminative algorithms for spoken language understanding," in *Proceedings of the Interspeech*, Antwerp, Belgium, 2007.

[15] K. Yao, G. Zweig, M.-Y. Hwang, Y. Shi, and D. Yu, "Recurrent neural networks for language understanding." in *Proceedings of the Interspeech*, Lyon, France, 2013.

[16] G. Mesnil, X. He, L. Deng, and Y. Bengio, "Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding," in *Proceedings of the Interspeech*, Lyon, France, 2013.

[17] N. T. Vu, P. Gupta, H. Adel, and H. Schütze, "Bi-directional recurrent neural network with ranking loss for spoken language understanding," in *Proceedings of the IEEE ICASSP*, Shanghai, China, 2016.

[18] V. Vukotic, C. Raymond, and G. Gravier, "A step beyond local observations with a dialog aware bidirectional gru network for spoken language understanding," in *Proceedings of the Interspeech*, San Francisco, CA, 2016.

[19] B. Liu and I. Lane, "Attention-based recurrent neural network models for joint intent detection and slot filling," in *Proceedings of the Interspeech*, San Francisco, CA, 2016.

[20] S. Zhu and K. Yu, "Encoder-decoder with focus-mechanism for sequence labelling based spoken language understanding," in *submission*, 2016.

[21] Y.-N. Chen, D. Hakkani-Tür, G. Tur, J. Gao, and L. Deng, "End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding," in *Proceedings of the Interspeech*, San Francisco, CA, 2016.

[22] P. Price, "Evaluation of spoken language systems: The atis domain."

[23] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the NIPS*, 2013.

[24] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng, "Improving word representations via global context and multiple word prototypes," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, 2012, pp. 873–882.

[25] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International Conference on Machine Learning*, 2015, pp. 1180–1189.