Representation Learning Using Multi-Task Deep Neural Networks for Semantic Classification and Information Retrieval

Xiaodong Liu[†]*, **Jianfeng Gao**[‡], **Xiaodong He**[‡], **Li Deng**[‡], **Kevin Duh**[†] **and Ye-Yi Wang**[‡]
[†]Nara Institute of Science and Technology, 8916-5 Takayama, Ikoma, Nara 630-0192, Japan
[‡]Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA
xiaodong-l@is.naist.jp, {jfgao,xiaohe,deng}@microsoft.com
kevinduh@is.naist.jp, yeyiwang@microsoft.com

Abstract

Methods of deep neural networks (DNNs) have recently demonstrated superior performance on a number of natural language processing tasks. However, in most previous work, the models are learned based on either unsupervised objectives, which does not directly optimize the desired task, or single-task supervised objectives, which often suffer from insufficient training data. We develop a multi-task DNN for learning representations across multiple tasks, not only leveraging large amounts of cross-task data, but also benefiting from a regularization effect that leads to more general representations to help tasks in new domains. Our multi-task DNN approach combines tasks of multiple-domain classification (for query classification) and information retrieval (ranking for web search), and demonstrates significant gains over strong baselines in a comprehensive set of domain adaptation.

1 Introduction

Recent advances in deep neural networks (DNNs) have demonstrated the importance of learning vector-space representations of text, e.g., words and sentences, for a number of natural language processing tasks. For example, the study reported in (Collobert et al., 2011) demonstrated significant accuracy gains in tagging, named entity recognition, and semantic role labeling when using vector space word representations learned from large corpora. Further, since these representations are usually in a low-dimensional vector space, they result in more compact models than those built from surface-form features. A recent successful example is the parser by (Chen and Manning, 2014), which is not only accurate but also fast.

However, existing vector-space representation learning methods are far from optimal. Most previous methods are based on unsupervised objectives such as word prediction for training (Mikolov et al., 2013c; Pennington et al., 2014). Other methods use supervised training objectives on a single task, e.g. (Socher et al., 2013), and thus are often constrained by limited amounts of training data. Motivated by the success of multi-task learning (Caruana, 1997), we propose in this paper a multi-task DNN approach for representation learning that leverages supervised data from many tasks. In addition to the benefit of having more data for training, the use of multi-task also profits from a regularization effect, i.e., reducing overfitting to a specific task, thus making the learned representations universal across tasks.

Our contributions are of two-folds: First, we propose a multi-task deep neural network for representation learning, in particular focusing on semantic classification (query classification) and semantic information retrieval (ranking for web search) tasks. Our model learns to map arbitrary text queries and documents into semantic vector representations in a low dimensional latent space. While the general concept of multi-task neural nets is not new, our model is novel in that it successfully combines tasks as disparate as operations necessary for classifica-

*This research was conducted during the author's internship at Microsoft Research.

tion or ranking.

Second, we demonstrate strong results on query classification and web search. Our multi-task representation learning consistently outperforms state-of-the-art baselines. Meanwhile, we show that our model is not only compact but it also enables agile deployment into new domains. This is because the learned representations allow domain adaptation with substantially fewer in-domain labels.

2 Multi-Task Representation Learning

2.1 Preliminaries

Our multi-task model combines classification and ranking tasks. For concreteness, throughout this paper we will use query classification as the classification task and web search as the ranking task. These are important tasks in commercial search engines:

Query Classification: Given a search query Q, the model classifies in the binary fashion as to whether it belongs to one of the domains of interest. For example, if the query Q is "Denver sushi", the classifier should decide that it belongs to the "Restaurant" domain. Accurate query classification enables a richer personalized user experience, since the search engine can tailor the interface and results. It is however challenging because queries tend to be short (Shen et al., 2006). Surface-form word features that are common in traditional document classification problems tend to be too sparse for query classification, so representation learning is a promising solution. In this study, we classify queries into four domains of interest: ("Restaurant", "Hotel", "Flight", "Nightlife"). Note that one query can belong to multiple domains. Therefore, a set of binary classifiers are built, one for each domain, to perform the classification. We frame the problem as four binary classification tasks. Thus, for domain C_t, our goal is binary classification based on $P(C_t|\ Q)$ ($C_t = \{0, 1\}$). For each domain t, we assume supervised data $(Q, y_t = \{0, 1\}$ with y_t as binary labels.[1]

Web Search: Given a search query Q and a document list $\mathbf{L}$, the model ranks documents in the order of relevance. For example, if the query Q is "Denver sushi", model returns a list of documents that satisfies such information need. Formally, we estimate $P(D_1|Q), P(D_2|Q), \ldots$ for each document D_n and rank according to these probabilities. We assume that supervised data exist; I.e., there is at least one relevant document D_n for each query Q.

[1]One could frame the problem as a a single multi-class classification task, but our formulation is more practical as it allows adding new domains without retraining existing classifiers. This will be relevant in domain adaptation (§3.3).

2.2 The Proposed Multi-Task DNN Model

Briefly, our proposed model maps any arbitrary queries Q or documents D into fixed low-dimensional vector representations using DNNs. These vectors can then be used to perform query classification or web search. In contrast to existing representation learning methods which employ either unsupervised or single-task supervised objectives, our model learns these representations using multi-task objectives.

The architecture of our multi-task DNN model is shown in Figure 1. The lower layers are shared across different tasks, whereas the top layers represent task-specific outputs. Importantly, the input X (either a query or document), initially represented as a bag of words, is mapped to a vector (l_2) of dimension 300. This is the shared semantic representation that is trained by our multi-task objectives. In the following, we elaborate the model in detail:

Word Hash Layer (l_1): Traditionally, each word is represented by a one-hot word vector, where the dimensionality of the vector is the vocabulary size. However, due to the large size of vocabulary in real-world tasks, it is very expensive to learn such kind of models. To alleviate this problem, we adopt the *word hashing* method (Huang et al., 2013). We map a one-hot word vector, with an extremely high dimensionality, into a limited letter-trigram space (e.g., with the dimensionality as low as 50k). For example, word *cat* is hashed as the bag of letter trigram {*#-c-a, c-a-t, a-t-#*}, where # is a boundary symbol. Word hashing complements the one-hot vector representation in two aspects: 1) out of vocabulary words can be represented by letter-trigram vectors; 2) spelling variations of the same word can be mapped to the points that are close to each other in the letter-trigram space.

Semantic-Representation Layer (l_2): This is a shared representation learned across different tasks. this layer maps the letter-trigram inputs into a 300-

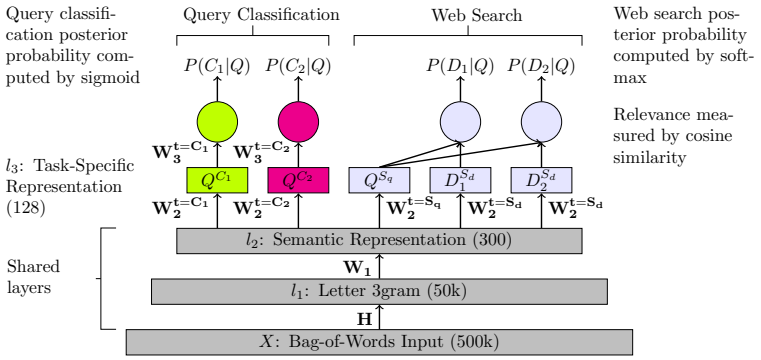


Figure 1: **Architecture of the Multi-task Deep Neural Network (DNN) for Representation Learning:** The lower layers are shared across all tasks, while top layers are task-specific. The input X (either a query or document, with vocabulary size 500k) is first represented as a bag of words, then hashed into letter 3-grams l_1. Non-linear projection W_1 generates the shared semantic representation, a vector l_2 (dimension 300) that is trained to capture the essential characteristics of queries and documents. Finally, for each task, additional non-linear projections W_2^t generate task-specific representations l_3 (dimension 128), followed by operations necessary for classification or ranking.

dimensional vector by

$$l_2 = f(\mathbf{W_1} \cdot l_1) \tag{1}$$

where $f(\cdot)$ is the tanh nonlinear activation $f(z) = \frac{1-e^{-2z}}{1+e^{-2z}}$. This 50k-by-300 matrix $\mathbf{W_1}$ is responsible for generating the cross-task semantic representation for arbitrary text inputs (e.g., Q or D).

Task-Specific Representation (l_3): For each task, a nonlinear transformation maps the 300-dimension semantic representation l_2 into the 128-dimension task-specific representation by

$$l_3 = f(\mathbf{W}_2^t \cdot l_2) \tag{2}$$

where, t denotes different tasks (query classification or web search).

Query Classification Output: Suppose $Q^{C_1} \equiv l_3 = f(\mathbf{W}_2^{t=C_1} \cdot l_2)$ is the 128-dimension task-specific representation for a query Q. The probability that Q belongs to class C_1 is predicted by a logistic regression, with sigmoid $g(z) = \frac{1}{1+e^{-z}}$:

$$P(C_1|Q) = g(\mathbf{W}_3^{t=C_1} \cdot Q^{C_1}) \tag{3}$$

Web Search Output: For the web search task, both the query Q and the document D are mapped into 128-dimension task-specific representations Q^{S_q} and D^{S_d}. Then, the relevance score is computed by cosine similarity as:

$$R(Q, D) = cos(Q^{S_q}, D^{S_d}) = \frac{Q^{S_q} \cdot D^{S_d}}{||Q^{S_q}||||D^{S_d}||} \tag{4}$$

Algorithm 1: Training a Multi-task DNN

Initialize model $\Theta : \{\mathbf{W_1}, \mathbf{W_2^t}, \mathbf{W_3^t}\}$ randomly
for *iteration in* $0...\infty$ **do**
1. Pick a task t randomly
2. Pick sample(s) from task t
   $(Q, y_t = \{0, 1\})$ for query classification
   $(Q, \mathbf{L})$ for web search
3. Compute loss: $L(\Theta)$
   $L(\Theta)$=Eq. 5 for query classification
   $L(\Theta)$=Eq. 6 for web search
4. Compute gradient: $\nabla(\Theta)$
5. Update model: $\Theta = \Theta - \epsilon\nabla(\Theta)$

end

The task t is one of the query classification tasks or web search task, as shown in Figure 1. For query classification, each training sample includes one query and its category label. For web search, each training sample includes query and document list.

2.3 The Training Procedure

In order to learn the parameters of our model, we use mini-batch-based stochastic gradient descent (SGD) as shown in Algorithm 1. In each iteration, a task t is selected randomly, and the model is updated ac-

cording to the task-specific objective. This approximately optimizes the sum of all multi-task objectives. For query classification of class C_t, we use the cross-entropy loss as the objective:

$$-\{y_t \ln P(C_t|Q) + (1-y_t)\ln(1-P(C_t|Q))\} \quad (5)$$

where $y_t = \{0,1\}$ is the label and the loss is summed over all samples in the mini-batch (1024 samples in experiments).

The objective for web search used in this paper follows the pair-wise learning-to-rank paradigm outlined in (Burges et al., 2005). Given a query Q, we obtain a list of documents $\mathbf{L}$ that includes a clicked document D^+ (positive sample), and J randomly-sampled non-clicked documents $\{D_j^-\}_{j=1,..,J}$. We then minimize the negative log likelihood of the clicked document (defined in Eq. 7) given queries across the training data

$$-\log \prod_{(Q,D^+)} P(D^+|Q) \quad (6)$$

where the probability of a given document D^+ is computed

$$P(D^+|Q) = \frac{\exp(\gamma R(Q,D^+))}{\sum_{D' \in \mathbf{L}} \exp(\gamma R(Q,D'))} \quad (7)$$

here, γ is a tuning factor determined on held-out data.

Additional training details: (1) Model parameters are initialized with uniform distribution in the range $(-\sqrt{6/(fan_{in}+fan_{out})}, \sqrt{6/(fan_{in}+fan_{out})})$ (Montavon et al., 2012). Empirically, we have not observed better performance by initialization with layer-wise pre-training. (2) Moment methods and AdaGrad training (Duchi et al., 2011) speed up the convergence speed but gave similar results as plain SGD. The SGD learning rate is fixed at $\epsilon = 0.1/1024$. (3) We run Algorithm 1 for 800K iterations, taking 13 hours on an NVidia K20 GPU.

2.4 An Alternative View of the Multi-Task Model

Our proposed multi-task DNN (Figure 1) can be viewed as a combination of a standard DNN for classification and a Deep Structured Semantic Model (DSSM) for ranking, shown in Figure 2. Other ways to merge the models are possible. Figure 3 shows an alternative multi-task architecture, where only the query part is shared among all tasks and the DSSM retains independent parameters for computing the document representations. This is more similar to the original DSSM. We have attempted training this model using Algorithm 1, but it achieves good results on query classification at the expense of web search. This is likely due to unbalanced updates (i.e. parameters for queries are updated more often than that of documents), and implying that the amount of sharing is an important design choice in multi-task models.

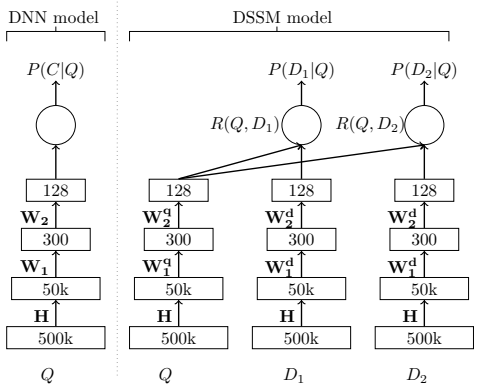


Figure 2: A DNN model for classification and a DSSM model (Huang et al., 2013) for ranking.

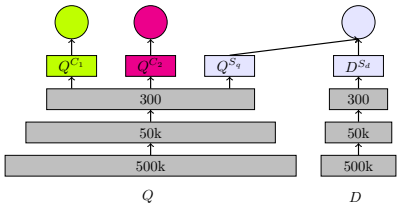


Figure 3: An alternative multi-task architecture. Compared with Figure 1, only the query part is shared across tasks here.

3 Experimental Evaluation

3.1 Data Sets and Evaluation Metrics

We employ large-scale, real data sets in our evaluation. See Table 1 for statistics. The test data for query classification were sampled from one-year log files of a commercial search engine with labels (yes or no) judged by humans. The test data for web search contains 12,071 English queries, where each query-document pair has a relevance label manually annotated on a 5-level relevance scale: *bad*, *fair*,

Task	Query Classification				Web Search
	Restaurant	Hotel	Flight	Nightlife	
Training	1,585K	2,131K	1,880K	1,214K	4,084K queries & click-through documents
Test	3,074	6,307	6,199	298	12,071 queries / 897,770 documents

Table 1: Statistics of the data sets used in the experiments.

good, *excellent* and *perfect*. The evaluation metric for query classification is the Area under of Receiver Operating Characteristic (ROC) curve (AUC) score (Bradley, 1997). For web search, we employ the Normalized Discounted Cumulative Gain (NDCG) (Järvelin and Kekäläinen, 2000).

3.2 Results on Accuracy

First, we evaluate whether our model can robustly improve performance, measured as accuracy across multiple tasks.

Table 2 summarizes the AUC scores for query classification, comparing the following classifiers:

- SVM-Word: a SVM model[2] with unigram, bigram and trigram surface-form word features.
- SVM-Letter: a SVM model with letter trigram features (i.e. l_1 in Figure 1 as input to SVM).
- DNN: single-task deep neural net (Figure 2).
- MT-DNN: our multi-task proposal (Figure 1).

The results show that the proposed MT-DNN performs best in all four domains. Further, we observe:

1. MT-DNN outperforms DNN, indicating the usefulness of the multi-task objective (that includes web search) over the single-task objective of query classification.
2. Both DNN and MT-DNN outperform SVM-Letter, which initially uses the same input features (l_1). This indicates the importance of learning a semantic representation l_2 on top of these letter trigrams.
3. Both DNN and MT-DNN outperform a strong SVM-Word baseline, which has a large feature set that consists of 3 billion features.

Table 3 summarizes the NDCG results on web search, comparing the following models:

[2]In this paper, we use the liblinear to build SVM classifiers and optimize the corresponding parameter C by using 5-fold cross-validation in training data. http://www.csie.ntu.edu.tw/ cjlin/liblinear/

System	Query Classification			
	Restaurant	Hotel	Flight	Nightlife
SVM-Word	90.91	75.82	91.17	91.27
SVM-Letter	88.75	69.65	85.51	87.71
DNN	97.38	76.81	95.58	93.24
MT-DNN	**97.57**	**78.56**	**96.21**	**94.20**

Table 2: Query Classification AUC results.

- Popular baselines in the web search literature, e.g. BM25, Language Model, PLSA
- DSSM: single-task ranking model (Figure 2)
- MT-DNN: our multi-task proposal (Figure 1)

Again, we observe that MT-DNN performs best. For example, MT-DNN achieves NDCG@1=0.334, outperforming the current state-of-the-art single-task DSSM (0.327) and the classic methods like PLSA (0.308) and BM25 (0.305). This is a statistically significant improvement ($p < 0.05$) over DSSM and other baselines.

To recap, our MT-DNN robustly outperforms strong baselines across all web search and query classification tasks. Further, due to the use of larger training data (from different domains) and the regularization effort as we discussed in Section 1, we confirm the advantage of multi-task models over than single-task ones.[3]

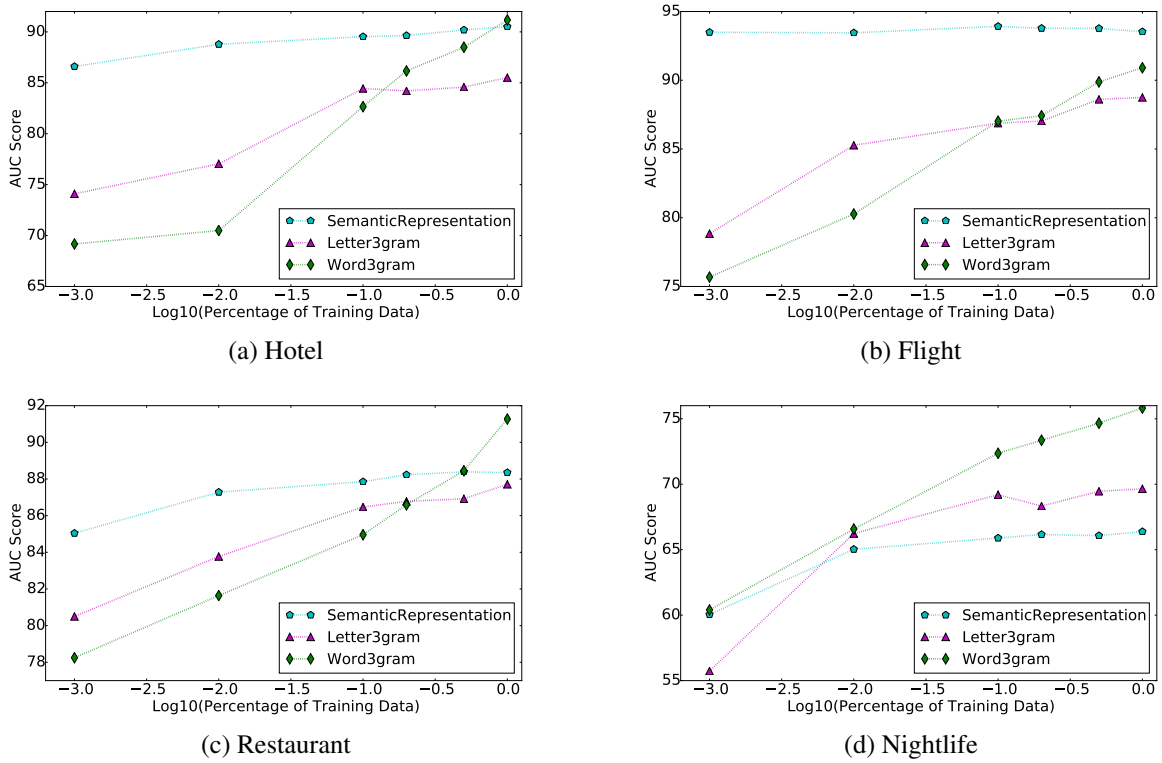
3.3 Results on Model Compactness and Domain Adaptation

Important criteria for building practical systems are agility of deployment and small memory footprint and fast run-time. Our model satisfies both with

[3]We have also trained SVM using Word2Vec (Mikolov et al., 2013b; Mikolov et al., 2013a) features. Unfortunately, the results are poor at 60-70 AUC, indicating the sub-optimality of unsupervised representation learning objectives for actual prediction tasks. We optimized the Word2Vec features in the SVM baseline by scaling and normalizing as well, but did not observe much improvement.

Models	NDCG@1	NDCG@3	NDCG@10
TF-IDF model (BM25)	0.305	0.328	0.388
Unigram Language Model (Zhai and Lafferty, 2001)	0.304	0.327	0.385
PLSA(Topic=100) (Hofmann, 1999; Gao et al., 2011)	0.305	0.335	0.402
PLSA(Topic=500) (Hofmann, 1999; Gao et al., 2011)	0.308	0.337	0.402
Latent Dirichlet Allocation (Topic=100) (Blei et al., 2003)	0.308	0.339	0.403
Latent Dirichlet Allocation (Topic=500) (Blei et al., 2003)	0.310	0.339	0.405
Bilingual Topic Model (Gao et al., 2011)	0.316	0.344	0.410
Word based Machine Translation model (Gao et al., 2010)	0.315	0.342	0.411
DSSM, J=50 (Figure 2, (Huang et al., 2013))	0.327	0.359	0.432
MT-DNN (Proposed, Figure 3)	**0.334***	**0.363**	**0.434**

Table 3: Web Search NDCG results. Here, * indicates statistical significance improvement compared to the best baseline (DSSM) measured by t-test at p-value of 0.05.



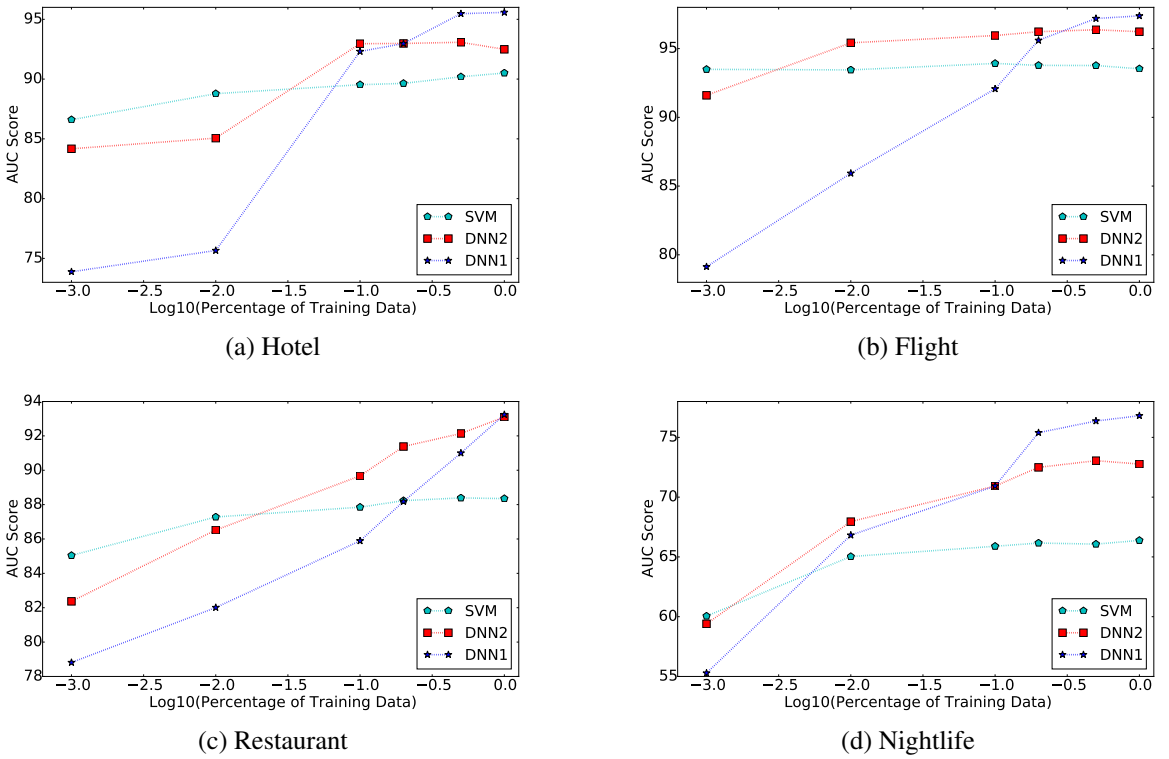
(a) Hotel

(b) Flight

(c) Restaurant

(d) Nightlife

Figure 4: Domain Adaption in Query Classification: Comparison of features using SVM classifiers. The X-axis indicates the amount of labeled samples used in training the SVM. Intuitively, the three feature representations correspond to different layers in Figure 1. **SemanticRepresentation** is the l_2 layer trained by MT-DNN. **Word3gram** is input X and **Letter3gram** is word hash layer (l_1), both not trained/adapted. Generally, **SemanticRepresentation** performs best for small training labels, indicating its usefulness in domain adaptation. Note that the numbers -3.0, -2.0, -1.0 and 0.0 in x-axis denote 0.1, 1, 10 and 100 percent training data in each domain, respectively.



(a) Hotel

(b) Flight

(c) Restaurant

(d) Nightlife

Figure 5: Domain Adaptation in Query Classification. Comparison of different DNNs.

high model compactness. The key to the compactness is the aggressive compression from the 500k-dimensional bag-of-words input to 300-dimensional semantic representation l_2. This significantly reduces the memory/run-time requirements compared to systems that rely on surface-form features. The most expensive portion of the model is storage of the 50k-by-300 W_1 and its matrix multiplication with l_1, which is sparse: this is trivial on modern hardware. Our multi-task DNN takes $<$ 150KB in memory whereas e.g. SVM-Word takes about 200MB.

Compactness is particularly important for query classification, since one may desire to add new domains after discovering new needs from the query logs of an operational system. On the other hand, it is prohibitively expensive to collect labeled training data for new domains. Very often, we only have very small training data or even no training data.

To evaluate the models using the above criteria, we perform domain adaptation experiments on query classification using the following procedure: (1) Select one query classification task t^*. Train MT-DNN on the remaining tasks (including Web Search task) to obtain a semantic representation (l_2); (2) Given a fixed l_2, train an SVM on the training data t^*, using varying amounts of labels; (3) Evaluate the AUC on the test data of t^*

We compare three SVM classifiers trained using different feature representations: (1) **SemanticRepresentation** uses the l_2 features generated according to the above procedure. (2) **Word3gram** uses unigram, bigram and trigram word features. (3) **Letter3gram** uses letter-trigrams. Note that **Word3gram** and **Letter3gram** correspond to SVM-Word and SVM-Letter respectively in Table 2.

The AUC results for different amounts of t^* training data are shown in Figure 4. In the Hotel, Flight and Restaurant domains, we observe that our semantic representation dominated the other two feature representations (**Word3gram** and **Letter3gram**) in all cases except the extremely large-data regime (more than 1 million training samples in domain t^*). Given sufficient labels, SVM is able to train well on **Word3gram** sparse features, but for most cases **Se-**

manticRepresentation is recommended.[4]

In a further experiment, we compare the following two DNNs using the same domain adaptation procedure: (1) **DNN1**: DNN where $\mathbf{W}_1$ is randomly initialized and parameters $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3^{t^*}$ are trained on varying amounts of data in t^*; (2) **DNN2**: DNN where $\mathbf{W}_1$ is obtained from other tasks (i.e. **SemanticRepresentation**) and fixed, while parameters $\mathbf{W}_2, \mathbf{W}_3^{t^*}$ are trained on varying amounts of data in t^*. The purpose is to see whether shared semantic representation is useful even under a DNN architecture. Figure 5 show the AUC results of DNN1 vs. DNN2 (the results **SVM** denotes the same system as **SemanticRepresentation** in Figure 4, plotted here for reference). We observe that when the training data is extremely large (millions of samples), one does best by training all parameters from scratch (DNN1). Otherwise, one is better off using a shared semantic representation trained by multi-task objectives. Comparing DNN2 and SVM with **SemanticRepresentation**, we note that SVM works best for training data of several thousand samples; DNN2 works best in the medium data range.

4 Related Work

There is a large body of work on representation learning for natural language processing, sometimes using different terminologies for similar concepts; e.g., feature generation, dimensionality reduction, and vector space models. The main motivation is similar: to abstract away from surface forms in words, sentences, or documents, in order to alleviate sparsity and approximate semantics. Traditional techniques include LSA (Deerwester et al., 1990), ESA (Gabrilovich and Markovitch, 2007), PCA (Karhunen, 1998), and non-linear kernel variants (Schölkopf et al., 1998). Recently, learning-based approaches inspired by neural networks, especially DNNs, have gained in prominence, due to their favorable performance (Huang et al., 2013; Baroni et al., 2014; Milajevs et al., 2014).

Popular methods for learning *word* representations include (Collobert et al., 2011; Mikolov et al., 2013c; Mnih and Kavukcuoglu, 2013; Pennington et al., 2014): all are based on unsupervised objectives of predicting words or word frequencies from raw text. End-to-end neural network models for specific tasks (e.g. parsing) often use these word representations as initialization, which are then iteratively improved by optimizing a supervised objective (e.g. parsing accuracy). A selection of successful applications of this approach include sequence labeling (Turian et al., 2010), parsing (Chen and Manning, 2014), sentiment (Socher et al., 2013), question answering (Iyyer et al., 2014) and translation modeling (Gao et al., 2014a).

Our model takes queries and documents as input, so it learns *sentence/document* representations. This is currently an open research question, the challenge being how to properly model semantic compositionality of words in vector space (Huang et al., 2013; M. Baroni and Zamparelli, 2013; Socher et al., 2013). While we adopt a bag-of-words approach for practical reasons (memory and run-time), our multi-task framework is extensible to other methods for sentence/document representations, such as those based on convolutional networks (Kalchbrenner et al., 2014; Shen et al., 2014; Gao et al., 2014b), parse tree structure (Irsoy and Cardie, 2014), and run-time inference (Le and Mikolov, 2014).

The synergy between multi-task learning and neural nets is quite natural; the general idea dates back to (Caruana, 1997). The main challenge is in designing the tasks and the network structure. For example, (Collobert et al., 2011) defined part-of-speech tagging, chunking, and named entity recognition as multiple tasks in a single sequence labeler; (Bordes et al., 2012) defined multiple data sources as tasks in their relation extraction system. While conceptually similar, our model is novel in that it combines tasks as disparate as classification and ranking. Further, considering that multi-task models often exhibit mixed results (i.e. gains in some tasks but degradation in others), our accuracy improvements across all tasks is a very satisfactory result.

5 Conclusion

In this work, we propose a robust and practical representation learning algorithm based on multi-task objectives. Our multi-task DNN model successfully combines tasks as disparate as classification and ranking, and the experimental results demon-

[4]The trends differ slightly in the Nightlife domain. We believe this may be due to data bias on test data (only 298 samples).

strate that the model consistently outperforms strong baselines in various query classification and web search tasks. Meanwhile, we demonstrated compactness of the model and the utility of the learned query/document representation for domain adaptation.

Our model can be viewed as a general method for learning semantic representations beyond the word level. Beyond query classification and web search, we believe there are many other knowledge sources (e.g. sentiment, paraphrase) that can be incorporated either as classification or ranking tasks. A comprehensive exploration will be pursued as future work.

Acknowledgments

We thank Xiaolong Li, Yelong Shen, Xinying Song, Jianshu Chen, Byungki Byun, Bin Cao and the anonymous reviewers for valuable discussions and comments.

References

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, Baltimore, Maryland, June. Association for Computational Linguistics.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2012. Joint learning of words and meaning representations for open-text semantic parsing. In *AISTATS*.

Andrew P Bradley. 1997. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28.

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)*, pages 740–750, Doha, Qatar, October. Association for Computational Linguistics.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6).

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

E. Gabrilovich and S. Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*.

Jianfeng Gao, Xiaodong He, and Jian-Yun Nie. 2010. Clickthrough-based translation models for web search: from word models to phrase models. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1139–1148. ACM.

Jianfeng Gao, Kristina Toutanova, and Wen-tau Yih. 2011. Clickthrough-based latent semantic models for web search. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 675–684. ACM.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2014a. Learning continuous phrase representations for translation modeling. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 699–709, Baltimore, Maryland, June. Association for Computational Linguistics.

Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, and Li Deng. 2014b. Modeling interestingness with deep neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2–13, Doha, Qatar, October. Association for Computational Linguistics.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. 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*, pages 2333–2338. ACM.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *NIPS*.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 633–644, Doha, Qatar, October. Association for Computational Linguistics.

Kalervo Järvelin and Jaana Kekäläinen. 2000. Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 41–48. ACM.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland, June. Association for Computational Linguistics.

Juha Karhunen. 1998. Principal component neural networkstheory and applications. *Pattern Analysis & Applications*, 1(1):74–75.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the International Conference on Machine Learning (ICML)*.

R. Bernardi M. Baroni and R. Zamparelli. 2013. Frege in space: A program for compositional distributional semantics. *Linguistic Issues in Language Technologies*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia, June. Association for Computational Linguistics.

Tomáš Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013c. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Matthew Purver. 2014. Evaluating neural word representations in tensor-based compositional settings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 708–719, Doha, Qatar, October. Association for Computational Linguistics.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*.

Gregoire Montavon, Genevieve Orr, and Klaus-Robert Muller. 2012. *Neural Networks: Tricks of the Trade 2nd ed*. springer.

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)*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.

B. Schölkopf, A. Smola, and K.-R. Müller. 1998. Nonlinear component analysis as kernel eigenvalue problem. *Neural Computation*, 10.

Dou Shen, Rong Pan, Jian-Tao Sun, Jeffrey Junfeng Pan, Kangheng Wu, Jie Yin, and Qiang Yang. 2006. Query enrichment for web-query classification. *ACM Trans. Inf. Syst.*, 24(3):320–352, July.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. 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*, pages 101–110. ACM.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October. Association for Computational Linguistics.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden, July.

Chengxiang Zhai and John Lafferty. 2001. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342. ACM.