MULTITASK LEARNING FOR SPOKEN LANGUAGE UNDERSTANDING

Gokhan Tur

Speech Technology and Research Laboratory
SRI International
Menlo Park, CA 94025, USA
gokhan@speech.sri.com

ABSTRACT

In this paper, we present a multitask learning (MTL) method for intent classification in goal oriented human-machine spoken dialog systems. MTL aims at training tasks in parallel while using a shared representation. What is learned for each task can help other tasks be learned better. Our goal is to automatically re-use the existing labeled data from various applications, which are similar but may have different intents or intent distributions, in order to improve the performance. For this purpose, we propose an automated intent mapping algorithm across applications. We also propose employing active learning to selectively sample the data to be re-used. Our results indicate that we can achieve significant improvements in intent classification performance especially when the labeled data size is limited.

1. INTRODUCTION

When building spoken language processing systems, usually data are collected and a model is trained for each application individually. However this may be suboptimal in cases where a number of similar systems are going to be built. Our domain is a call routing system where the aim is to route the input calls in a customer care call center. In this spoken dialog system, callers are greeted by the open ended prompt "*How May I Help You?*" encouraging them to utter their requests in natural language. The system then tries to identify the customer's intent (call-type) using a spoken language understanding (SLU) component. In the event the system is unable to understand the caller with high enough confidence, the conversation usually proceeds with either a reprompt or a confirmation prompt.

The understanding step can be seen as a classification problem [1]. For this purpose, data-driven classifiers are trained using large amounts of task data which is usually transcribed and then labeled by humans, an expensive and laborious process. By "labeling", we mean assigning one or more of the predefined intents to each utterance. As an example, consider the utterance *I would like to know my account balance*, in a customer care application from a financial domain. Assuming that the utterance is recognized correctly, the corresponding intent would be *Request(Balance)* and the action would be telling the balance to the user after prompting for the account number or routing this call to the billing department.

Building many spoken dialog systems using similar intent classification models in a shorter time frame motivates us to re-use the existing labeled data from various applications to improve the performance. In our previous work, we have presented a model adaptation approach, where a better model is built by adapting an existing model from a similar application [2] with the same call-types. Furthermore, we have presented a library-based approach, where a human expert can bootstrap the new application model by manually selecting data from the library and augmenting them with rules [3].

In this paper, we present a multitask learning (MTL) method for natural language intent classification. MTL aims at training tasks (in our case, applications) in parallel while using a shared representation [4]. While typically learning algorithms learn one task at a time, what is learned for each task can help other tasks be learned better. Although the originally suggested MTL framework uses backpropagation neural networks, the idea is more general. MTL has been employed for many tasks during the last decade, even for speech and language processing: Parveen and Green have employed MTL for isolated-word connectionist speech recognition and obtained error reductions ranging from 20% to 50% [5]. Florian and Ngai have proposed a version of MTL for English and Chinese part of speech tagging and base noun phrase chunking with transformation based learning [6]. Sutton and McCallum have employed MTL under a more general framework called transfer learning for conditional random fields. They have shown improvements using MTL for named entity extraction using ACE and CoNLL data sets [7].

In this study, we propose reusing the already labeled

This work was done when the author was with AT&T Labs - Research, Florham Park, NJ 07932, USA.

data across applications while training. This is similar to model adaptation, however in this case we do not assume that call-types are the same. For multitask learning the main issue is the naming and granularity inconsistencies between call-type sets of different applications. We propose a method to find a mapping to resolve this issue using the already labeled data. Furthermore, instead of simply reusing all the labeled data, inspired by active learning, we present a method to only reuse the most informative labeled data.

In the following section, we briefly explain the MTL approach we have employed with the proposed mapping method. Then, in Section 3, we propose an active MTL method. We conclude after presenting the experiments and results.

2. MULTITASK LEARNING

MTL suggests data amplification to enable the learner to generalize better [4]. The idea is to simply concatenate the training data of various applications. In order to concatenate the training data, MTL requires a shared representation among the tasks, which are going to be learned in parallel. This requires the feature space and the set of classes to be the same. For our tasks, the input, hence the feature space, is always the same: the current utterance and its n-grams. The problem is the set of classes, since they differ across applications.

In the AT&T Spoken Language Understanding System, the call-types are designed to capture information that is sufficient to fulfill users' request [1]. Moreover, the call-types are not motivated by the action that needs to be taken, instead by the intent of the user. Still, it is not uncommon that the very same intents have been labeled differently across various applications due to various reasons:

- *Naming Inconsistencies*: One common reason for the mismatched call-types is due to considering different namings of two call-types, c_i and c_j ($c_i = c_j$). For example, in one application *Request(Bill)* call-type can be renamed as *Ask(Bill)*, indicating a vague request about a customer's bill, such as *I have a question about my bill.*
- *Different Call-type Granularities*: Due to specific application requirements, it is very common to have call-types with different granularities:

  $$c_i = c_{j1} \cup c_{j2} \cup ... \cup c_{jk}$$

  where c_i is one call-type from one application and $c_{j1}...c_{jk}$ are call-types from another application. For example, one application might label the utterance *At what time do you close?* with the call-type *Request(Store_Hours)* and the utterance *Where exactly is your Manhattan store?* with the call-type *Request(Store_Location)*, and another application might label the very same utterances with the single call-type *Request(Store_Info)*. Although while designing a new application the human designers make effort for consistency with the previous applications, there may be specific design requirements or these utterances may need to be treated differently.

Another reason for call-type mismatch might be due to overlaps among call-types, i.e. a given call-type, c_i, intersects with more than one call-types, c_j and c_k ($c_i \subset c_j$ and $c_i \subset c_k$ and $c_j \cup c_k \neq c_i$). Since they are not frequently seen in the AT&T SLU system, we assume these do not happen.

In order to make similar but not equivalent call-types usable across applications, we propose an automated mapping procedure. In this study, we assume that we have some amount of labeled data for these applications. First we train individual models, M_i and M_j using corresponding training data sets, D_i and D_j. Then the idea is using these labeled data and models to find out the call-type mappings. We converted this to an information (in this case call-type) retrieval problem. The goal is to retrieve the call-types that are merged in the other application. The recall is defined as the ratio of call-types that are selected among the actual, and the precision is defined as the ratio of call-types that are actually merged. First we automatically cross-label the data sets of two applications, D_i and D_j, using the existing models. We call an utterance, s_i, to be automatically labeled if the confidence score, $CS(s_i)$, is more than some threshold. In this work, we let

$$CS(s_i) \approx \max_{c_j} P(c_j|W)$$

where c_j is the call-type and W is the utterance. Using the call-types automatically assigned by M_i to D_j, $\hat{C}_j$, and actual call-types for D_j, C_j, it is possible to estimate the merged or renamed call-types. For example if all instances of call-type c_i is automatically labeled as $\hat{c}_j$, this indicates a renaming. In order to keep the precision high, we require that a call-type of the existing model to be labeled as a new call-type by more than a certain threshold precision. Then, the split call-types can be found vice-versa. This algorithm can be extended in case there are more than two applications, by trying binary combinations.

After the mapping is found, the last step of MTL is retraining the existing model using this information: The call-types which are found to be merged in D_j are also merged in D_i. The call-types which are found to be split in D_j are handled using the labels in $\hat{C}_j$. We get a new data set after these rename, merge, and split operations, $\hat{D}_i$ for application i. The new models are then built using the application's own training data, D_i and the mapped data from the other application, $\hat{D}_j$. More formally, the MTL procedure is as follows:

- Train M_i with D_i and M_j with D_j
- Form mapped data sets $\hat{D}_i$ and $\hat{D}_j$.
- Re-train M_i with $D_i \cup \hat{D}_j$.

3. MULTITASK ACTIVE LEARNING

Inspired by active learning, we propose an extension of the data amplification method of MTL. Active learning aims at reducing the number of training examples to be labeled by selectively sampling a subset of the unlabeled data. This is done by inspecting the unlabeled examples and selecting the most *informative* ones, with respect to a given cost function, for a human to label [8]. In our previous work, we have proposed using active learning for spoken language understanding [9]. In this study, the idea is instead of adding all the data from other applications, one can only add the most informative ones. Based on certainty-based active learning, we use the confidence scores of the utterances, $CS(s_i)$, as the criterion for informativeness, and used the utterances whose confidence scores are lower than some threshold. More formally, the active MTL procedure is as follows:

- Train M_i with D_i and M_j with D_j
- Form mapped data sets $\hat{D}_i$ and $\hat{D}_j$.
- Use M_i to classify $\hat{D}_j$ to get $CS(\hat{d}_j)$ for all $j = 1..n$ where $\hat{D}_j = \hat{d}_1, ..., \hat{d}_n$.
- $\hat{\hat{D}}_j = \{\hat{d}_j : CS(\hat{d}_j) < threshold\}$
- Re-train M_i with $D_i \cup \hat{\hat{D}}_j$.

Although this approach reduces the number of examples added to the training data, D_i, it implicitly gives more weight to the examples which are not seen before, hence got a lower score.

4. EXPERIMENTS AND RESULTS

We evaluated the proposed methods using the utterances from the database of the AT&T VoiceTone® spoken dialog system [10]. We performed our tests using the Boostexter classification tool [11], an implementation of the Boosting family of classifiers. Boosting is an iterative procedure; on each iteration a weak classifier is trained on a weighted training set, and at the end, the weak classifiers are combined into a single, combined classifier. For all experiments, we used word trigrams as features, and each weak classifier (e.g. "decision stump") checks the absence or presence of a feature.

We used two applications from the telecommunications domain and checked whether automatically selecting utterances with mapped call-types from one application, T_2, would help the other one, T_1. The data characteristics for the two applications used in the experiments are given in Table 1. In our experiments all of the utterances are transcribed in order not to deal with ASR errors.

While evaluating the classification performance, we used the *top class error rate* (TCER), which is the fraction of utterances in which the call-type with maximum probability was not one of the true call-types.

	T_1	T_2
Training Data Size	35,551 utt.	9,093 utt.
Test Data Size	5,000 utt.	5,172 utt.
Number of Calltypes	65	84
Call-type Perplexity	14.7	29.3
Average Utterance Length	12 words	13 words

Table 1. Data characteristics used in the experiments.

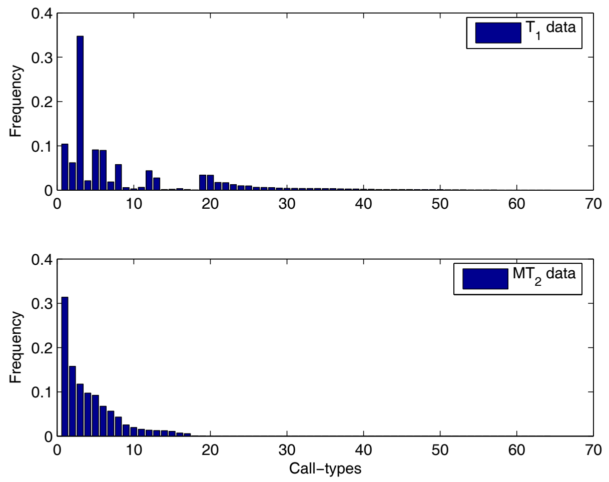

Fig. 1. Call-type frequencies for the T_1 data and the mapped T_2 ($\hat{D}_2$) data. The call-types are aligned.

Before reusing data from the application T_2, one needs to map the call-types into the other application. Using the method explained in Section 2, we have come up with 19 mappings. Some frequent examples include:

Make(Payment) $\rightarrow$ *Pay_Bill*
Request(Sales) $\rightarrow$ *New_Service*
Tellme(Balance) $\rightarrow$ *Account_Balance*
Verify(Payment) $\rightarrow$ *Account_Balance*

Note that the last two ones indicate two merged call-types. After these mappings, we have filtered out the utterances of T_2, whose call-types are unknown to Application T_1. This left us with about half of the all data of T_2, more specifically, with 4,666 utterances. The call-type frequencies of T_1 and mapped utterances of T_2 ($\hat{D}_2$) are given in Figure 1. As seen, the most frequent call-types of T_1 also exist in the $\hat{D}_2$ data and in total, only 11.5% of the utterances of T_1 has a call-type which is not seen in $\hat{D}_2$.

When we use the same 4,666 utterances ($\hat{D}_2$) as the sole training data, we get a TCER of 31.84% on the T_1 test set. Note that this figure is only 4.6% inferior to 27.26%, the performance when we use a random subset of the same size from T_1 training data. After getting these promising results, as the first experiment, we added $\hat{D}_2$ to the T_1 data, but got

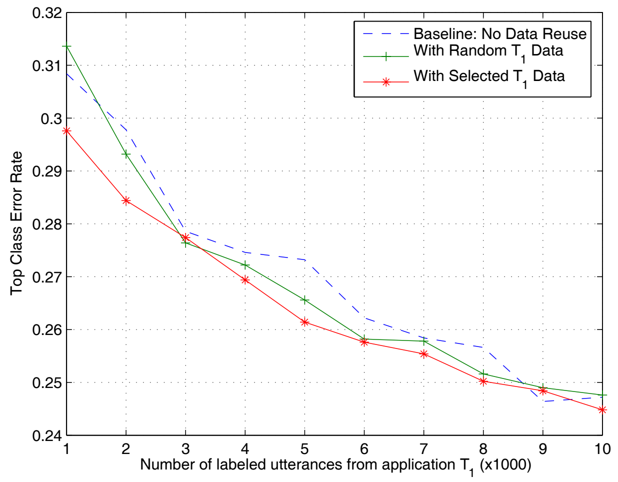

Fig. 2. Results using MTL. x-axis is the amount of labeled data from application T_1. Top learning curve is obtained using just T_1 data as a baseline. Below that lies the learning curves using the random and active MTL.

no improvement. Then in order to apply active MTL, we tested $\hat{D}_2$ with the T_1 model and selected 897 utterances, which have got low confidences, to obtain $\hat{\hat{D}}_2$, as explained Section 3. Figure 2 shows the learning curve by adding these 897 utterances to the training data of the T_1. The top most curve is the baseline and obtained without any data re-use. In order to check the effect of using selected data, we have randomly selected 897 utterances among 4,666 utterances, then added these to the T_1 training data. This is the curve lying below the baseline. As seen, MTL helped at all data points, until T_1 has about 9,000 labeled training utterances. Note that this is about 10 times the amount of data re-used. Furthermore, for the first 2 data points, the improvement is significant[1]. This figure also proves the effectiveness of the selective sampling of data to be re-used for MTL. At almost all data points, active MTL outperformed random MTL.

5. CONCLUSIONS

We have presented an application of MTL for natural language intent classification. We have shown that, for this task, using the proposed methods, it is possible to improve the performance of a spoken language understanding system significantly when there is not much training data available. We have also proposed combining MTL with active learning to selectively sample the data to re-use.

MTL is also applicable to many other speech and language processing tasks. For example, the well-known ATIS SLU task [12] requires the system to determine the departure and arrival cities in the utterances. One can use a named entity extraction task training data to determine the locations to improve the performance of these two sub-named entities in ATIS, and vice versa. This corresponds to merged classes in our case.

Note that in this paper, we have only employed the simplest method of MTL, i.e. data amplification. This is intuitively suboptimal in some cases. For example, since most classifiers are sensitive to the prior distribution of the classes, changing this distribution may harm the performance. This also leads to an important research area while employing MTL: selecting the utterances to be re-used.

Our future work includes using MTL with more than two applications. The ultimate goal is to collectively improve the performance of all these applications in parallel.

[1]Z-test with 0.95 confidence interval

6. REFERENCES

[1] N. Gupta, G. Tur, D. Hakkani-Tür, S. Bangalore, G. Riccardi, and M. Rahim, "The AT&T spoken language understanding system," *IEEE Transactions on Speech and Audio Processing*, To appear.

[2] G. Tur, "Model adaptation for spoken language understanding," in *Proceedings of the ICASSP*, Philadelphia, PA, May 2005.

[3] G. Di Fabbrizio, G. Tur, and D. Hakkani-Tür, "Bootstrapping spoken dialog systems with data reuse," in *Proceedings of the SigDial Workshop*, Boston, MA, May 2004.

[4] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, 1997.

[5] S. Parveen and P. Green, "Multitask learning in connectionist robust asr using recurrent neural networks," in *Proceedings of the EUROSPEECH*, Geneva, Switzerland, September 2003.

[6] R. Florian and G. Ngai, "Multidimensional transformation-based learning," in *Proceedings of the CoNLL*, Toulouse, France, July 2001.

[7] C. Sutton and A. McCallum, "Composition of conditional random fields for transfer learning," in *Proceedings of the HLT/EMNLP*, Vancouver, Canada, October 2005.

[8] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Machine Learning*, vol. 15, pp. 201–221, 1994.

[9] G. Tur, R. E. Schapire, and D. Hakkani-Tür, "Active learning for spoken language understanding," in *Proceedings of the ICASSP*, Hong Kong, May 2003.

[10] M. Gilbert, J. G. Wilpon, B. Stern, and G. Di Fabbrizio, "Virtual agents for contact center automation," *IEEE Speech Processing Magazine*, vol. 22, no. 5, September 2005.

[11] R. E. Schapire and Y. Singer, "Boostexter: A boosting-based system for text categorization," *Machine Learning*, vol. 39, no. 2/3, pp. 135–168, 2000.

[12] P. J. Price, "Evaluation of spoken language systems: The ATIS domain," in *Proceedings of the DARPA Workshop on Speech and Natural Language*, Hidden Valley, PA, June 1990.