Exploiting Shared Information for Multi-intent Natural Language Sentence Classification

Puyang Xu, Ruhi Sarikaya

Microsoft Corporation

{puyangxu, ruhi.sarikaya}@microsoft.com

意图共享槽位问题解决了吗？

Abstract

Multi-intent natural language sentence classification aims at identifying multiple user goals in a single natural language sentence (e.g., "find Beyonce's movie and music" → find_movie, find_music). The main motivation of this work is to exploit the shared intents across different intent combinations rather than treating the combination as an atomic label. We propose to achieve this by (1) adding class features, and (2) adding hidden variables to identify segments belonging to each intent. Experimental results demonstrate significant gains in classification accuracy over the baseline methods across a number of training conditions (3%-8% absolute on multi-intent sentences, 2%-3% absolute on single intent sentences).

Index Terms: spoken language understanding, semantic classification, hidden variable models

1. Introduction

In spoken dialogue systems, understanding the user's intent is a crucial step for the success of human-computer interaction. The natural language sentences are classified into predefined intent categories usually through the use of discriminative classifiers. The examples of such natural language sentences and their corresponding intents are shown below.

- search for 60s' music → find_music
- get movies like independence day → find_similar

In a typical human-to-human dialogue, a single sentence can often carry multiple intents. The ability of humans to understand and react to such multi-intent sentences allows for much more natural and smooth conversation than having to express every intent separately.

In contrast, in most of the existing human-computer dialogue systems [1, 2, 3, 4, 5], only one intent is assumed to be present in each sentence. Such an assumption may limit the information throughput and lead to unnatural dialogue experience slowing down the task completion because the user has to wait until the system has finished processing one intent before proceeding to the next. In order for the system to process multiple requests in a single dialogue turn, the intent classifier needs to be able to detect multiple intents for each input sentence as follows,

- check the weather in NY and book a flight for tomorrow morning → check_weather, book_flight,
- allow no interruptions while i send the following text → send_text, do_not_disturb.

It is important to point out that such multi-intent sentences usually contain cues associated with multiple intents. The ambiguous evidence can cause trouble for the classifiers trained only on single intents. We have found that the straightforward solution of taking top-k hypotheses generated by the existing single intent classifier yields poor accuracy. In order to detect multiple intents, it is preferable to have multi-intent data available. When training examples are associated with more than one labels, many of the multi-class algorithms can not handle them naturally. *Multi-label learning* [6] is usually the more appropriate framework for solving this problem.

Multi-label learning problems are usually tackled through some kind of transformation of the classification problem. A popular approach is to transform the original $|C|$-class classifier into $|C|$ binary classifiers, one for each $c \in C$ [7, 8]. Since each binary classifier is operated independently, arbitrary numbers of labels can be assigned to a single instance.

Another type of transformation is to directly treat the label combinations as a single label. Thus the classifier can be learned and used in the same fashion as the standard single-label classification. Such an approach can be susceptible to data sparsity problem – the number of label combinations can be too large and most of them may not have sufficient samples in the dataset. However, it was found in [6] that considering the label combination as an atomic label yields the best classification accuracy on various tasks among a number of multi-label learning methods. As we will demonstrate, we are able to corroborate such results in our experiments. Both of the aforementioned approaches will be investigated as baseline systems to compare with the techniques we will introduce in this paper.

Although directly predicting multiple labels as a single label is empirically proven to be an effective technique, there is clearly room for improvement. For all the instances that are assigned multiple labels, they usually bear the characteristics of each one of the classes to which they belong. Treating them as an atomic class different from all the others discards the valuable information that are shared by many classes, making it more likely to suffer from data sparsity problem. For example, the double intent *buy_game#play_game* is certainly not entirely different from *buy_game#play_music* or the single intent *buy_game*. The words and phrases that people use to express the intent *buy_game* usually appear in all these classes, making it possible for us to improve the performance of the classifier by leveraging such commonality across different classes.

Therefore, the goal of this work is to investigate whether the information overlap among different intent combinations can be exploited to improve multi-intent detection upon the standard multi-label learning techniques. Specifically, we introduce two novel approaches, namely (1) the class-based models, where we add features shared by a set of intent combinations, and (2) the hidden variable models, where the input sentences are implicitly segmented to indicate the word sequences belonging to each single intent.

Although the number of intents to handle is often considered to be a design choice by many systems, to the best of our knowledge, there has been no investigation on the classification techniques that can be used to improve the detection of multiple intents over basic approaches.

The rest of the paper is organized as follows: We first briefly describe the feature-based log-linear model and Perceptron training in Section 2, as they will be used as the primary framework throughout our experiments. Adding class features is the topic of Section 3. The hidden variable model is introduced in Section 4. We compare various techniques in Section 5, where the experimental results are presented, followed by conclusions in Section 6.

2. Log-linear models and Perceptron training

In this work, an important baseline system is the previously described multi-label learning technique in which the combination of classes are treated as an atomic label. We will adopt log-linear models together with Perceptron training to implement this baseline system. The class models and the hidden variable models are all extensions to this baseline approach. Before we introduce our methods, we first briefly describe log-linear models and Perceptron training.

Log-linear models are widely used in the machine learning community. It allows a flexible framework to incorporate different knowledge sources as feature constraints. Specifically, the log-linear model defines a probability distribution over C, namely the set of class labels, as illustrated in equation (1),

$$P(c|W) = \frac{\exp\left(f(W,c)\cdot\theta\right)}{\sum_{c'\in C}\exp\left(f(W,c')\cdot\theta\right)}, \quad (1)$$

where f is the vector of feature functions defined jointly over the input W and the class label. θ is the corresponding vector of feature weights. The denominator is a normalizer summing over all $|C|$ classes.

Perceptron training is an effective technique that dates back to the 1950s [9]. Compared with other training methods, it has a much simpler learning procedure. Specifically, for each training instance (W, c^*), we update the parameters following equation (2) if the prediction $\hat{c}$ by the current model differs from the correct label c^*,

$$\theta += \eta\left(f(W,c^*) - f(W,\hat{c})\right). \quad (2)$$

3. Adding class features

We will mainly look at the n-gram patterns in the query text to determine which class of intent combination that the query belongs to. Therefore, for our purpose, the feature function can be represented as a string which takes the form of *n-gram_target*, where *target* can be a single intent label or a multi-intent label such as *buy_game#play_game*.

As we have discussed, for target labels consisting of multiple intents, it can be advantageous to also model the embedded intents rather than treating it as an atomic label different from all others. To enable this, for target label *buy_game#play_game*, besides the standard *n-gram_buy_game#play_game* features, we also define n-grams features dependent on each embedded intent, namely *n-gram_buy_game*, *n-gram_play_game*, ...

Note that these features directly predict *buy_game* and *play_game* individually as target classes. Thus by having these features learned also on intent combinations containing each single intent, we allow the information to be shared across different target labels. Intuitively, if we do not have sufficient examples for certain combinations, we should still be able to predict them because the embedded classes in them are modeled utilizing more data.

Adding such class-level information to improve generalization has found success for other tasks in the community. In statistical language modeling, where the goal is to predict the next word based on the history, introducing grouping of the predicted word has been shown to improve the quality of the language model [10]. The word groups are usually derived according to word statistics in a text corpus. For our purpose, the grouping of target labels can be derived in an easier fashion – each embedded class in the target label forms a group.

In fact, by inspecting the names of the intents, in addition to the label-level sharing, a deeper level of sharing can be exploited. In our dataset, among the 58 intent names, 16 of them have *find* as the first word (find_movie, find_game, etc), 8 of them have *get*, 3 of them have *buy*... Since the first word of the intent name usually describe the intended action by the user, this can be seen as the sharing at the action level. We can thus define new features in the form of *ngram_action*, allowing the frequent actions to be modeled leveraging data across multiple target classes.

4. Adding hidden variables

Instead of assuming the entire sentence belongs to multiple classes, the approach described in this section aims at inducing a segmentation of the sentence, and assigning class labels using information at the segment level. Such a more refined modeling strategy enables us to specifically carve out the word sequences representing each single intent, and identify the parts of the sentence that are shared by samples belonging to different intent combinations.

One can certainly think of using existing sentence segmentation algorithms before passing each segment to the standard intent classifier. However, natural language sentences often do not have clear boundaries. For sentences like "find avatar and play it", segmenting it into two parts and processing them separately may cause trouble for the semantic understanding – "play it" as an isolate unit is not too informative about what the user intends to do. Instead of taking such a cascade approach, we introduce hidden variables into the classification model, so that the segmentation model is optimized jointly for the purpose of semantic classification.

4.1. Previous work

Adding hidden variables has been investigated before for the purpose of text classification [11]. However, it was used for generative modeling (mixture model) as opposed to discriminative modeling and the task was different from ours. In the discriminative setting, adding hidden variables has been shown to be an effective technique for tasks such as phone classification [12] and gesture recognition [13]. These models are generally described as *hidden conditional random fields*.

4.2. Model description

For better illustration of the proposed hidden variable model, we first represent the baseline log-linear classification model described in Section 2 as an *undirected graphical model*, as depicted in Figure 1. Without loss of generality, each node in the figure can represent the entire n-gram history rather than

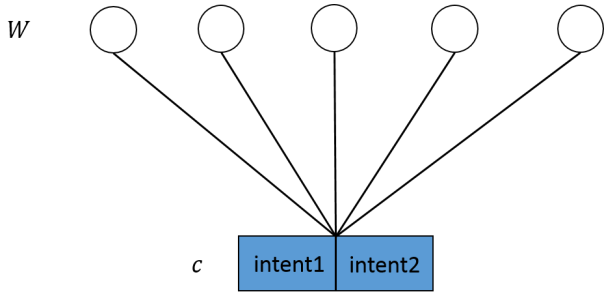

Figure 1: *Graphical model representation of the baseline log-linear classification model.*

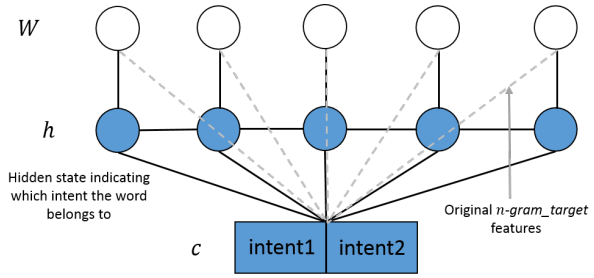

Figure 2: *Graphical model representation of the hidden variable model.*

a single word. Each undirected edge connecting a node in the word sequence W and the target class c represents a *potential function*, which is essentially the exponential sum of a set of weighted n-gram feature functions (unigram, bigram, up to n-gram).

The proposed hidden variable model is depicted in Figure 2. Compared with Figure 1, a hidden variable layer is attached to the word sequence, indicating the type of intent each word belongs to ($h \in \{1, 2, ..., |I|\}$, where $|I|$ is the number of single intents). For a particular intent combination c, the hidden variable can only take values corresponding to the types of intents embedded in c. Thus the sequence of hidden variables introduces a segmentation of the sentence by specifying the words belonging to each single intent.

Formally, the probability of c given W can be written as

$$P(c|W) = \sum_h P(c, h|W)$$
$$= \sum_h \frac{e^{\sum_{i=1}^{i=|W|} \theta \cdot \left[g(W_i,c),\, e(W_i,h_i,c),\, t(h_i,h_{i-1},c)\right]}}{Z}, \quad (3)$$

where Z sums over both c and h. As we can see, aside from the original n-*gram_target* features $g()$, which are indicated by the dashed line, the feature vector is augmented with the hidden state transition features $t()$ (edges between hidden nodes), and the hidden state emission features $e()$ (edges between hidden nodes and word nodes), which essentially measure how likely the word is generated by the intent type it is assigned to.

4.3. Latent variable Perceptron training

Likelihood-based training can be applied to the model described in equation (3). However, computing the gradient of the likelihood function with respect to the parameters usually involves forward-backward passes over the hidden states trellis, which can be computationally expensive. In [14, 15], it was shown that Perceptron training can also be used to estimate models with hidden dependencies. Not only does it reduce the training time significantly, it also yields competitive results on a number of tasks.

Under latent variable Perceptron training, the hidden variables are not marginalized out. It is assumed that $P(c|W) \approx \max_h P(c, h|W)$. Therefore, the correct class c^* along with the 1-best sequence of hidden variables h^{c^*} becomes the golden target of Perceptron training. Similarly, the best-scoring class predicted by the model ($\hat{c} = \arg\max_c P(c|W)$) is also associated with its 1-best segmentation $h^{\hat{c}}$. Whenever c^* differs from $\hat{c}$, parameters are updated in a similar fashion as equation (2),

$$\theta \quad += \quad \eta \sum_{i=1}^{|W|} \left[g(W_i, c^*),\, e(W_i, h_i^{c^*}, c^*),\, t(h_i^{c^*}, h_{i-1}^{c^*}, c^*) \right]$$
$$- \quad \left[g(W_i, \hat{c}),\, e(W_i, h_i^{\hat{c}}, \hat{c}),\, t(h_i^{\hat{c}}, h_{i-1}^{\hat{c}}, \hat{c}) \right] \quad (4)$$

5. Experimental Results

5.1. Data & Setup

The dataset consists of 31178 natural language queries in the domains of movie, music, games and generic commands. 14784 of them are single intent sentences, while 16394 of them contain double intents. The dataset is divided into 70%, 10% and 20% partitions, used for training, tuning and testing respectively.

It is worth pointing out that although the described techniques are not restricted to handling only double intent sentences, it is often rare for a single sentence to carry more than two intents in a natural dialogue. Therefore, we did not attempt to collect query data containing more than two intents.

When measuring the classification accuracy for double intent sentences, a prediction is considered to be correct only when both intents are predicted correctly. The order of intents are not considered in this work.

5.2. Two baseline approaches

In *Boostexter* [16], the well-known *Adaboost* algorithm was adapted to handle multi-label classification. Internally, it replaces each training example with a set of $\{0, 1\}$-labeled examples (one for each output class), so that the standard binary *Adaboost* can be used. As we discussed in Section 1, such reduction to binary problems naturally extends itself to multi-label scenarios. In our experiments, one of the baseline systems is *Boostexter*. We use its publicly available implementation *icsiboost* [17].

Another baseline approach we will compare with is the classification model treating combinations of intents as an atomic label. As we mentioned, this is also one of the commonly used technique for multi-label learning. The class models and the hidden variable models are both extension within this framework (directly predicting intent combinations). We will call this method the atomic model for the rest of the paper.

5.3. Model features

We use label dependent n-gram features up to trigrams for *icsiboost* and the atomic model. In addition, the class-based models also use various sub-label (single intent, action) dependent n-gram features as we have described.

For the hidden variable model, the order of state dependent n-grams (emission features) can impact the complexity of training and inference. Constrained by the Markov assumption of the underlying model, using trigram emission features in addition to bigrams for double intent sentences would increase the number of hidden states from 2 to 4, which leads to a 4 times larger complexity of inference. Therefore, we constrain ourselves to using only bigrams for the state-level emission features.

5.4. Results

The first set of experiments are conducted making use of only double intent data (DI). Table 1 demonstrates the classification accuracy of using the various approaches. The class-based

Table 1: *Intent detection accuracy (%) on DI test set using different models trained on DI data*

Model	Accu
multi-label icsiboost	78.7
atomic model	81.7
class model (label)	83.6
class model (label+act)	83.0
hidden variable model	82.8

model and the hidden variable model, both outperform the two baseline approaches by noticeable margins. The class model with label-level sharing appears to be the best approach in this setup.

In the second setup, we also add single intent sentences (SI) in training and testing. This is often necessary because the intent classifier should be able to handle both SI and DI sentences simultaneously. We also vary the amount of DI data to simulate more data-scarce training conditions. This is also a realistic situation as it is usually more difficult to collect DI data than SI data. If we scale up to support more intent combinations, data sparsity can become a more serious problem – most of the intent combinations may not have sufficient examples in our training set.

In Figure 3 and 4, the detection accuracy on DI and SI are plotted with varying amounts of DI data (10%, 25%, 50% and 100%). Comparing Table 1 and the right-most column of Figure 3, adding SI data into training degrades the accuracy on DI data noticeably for all of the methods. As we reduce the amount the DI data, the performance drops significantly, indicating the importance of DI training data for classifying DI sentences.

Meanwhile, both of the proposed techniques are able to achieve higher classification accuracy over the baseline approaches. Such an advantage is particularly obvious on the DI test set with small amounts of DI training data. For class-based models, adding action-level sharing helps in some cases but does not seem to be consistent. The hidden variable model appears to be the best method under all training conditions, outperforming also the class-based techniques substantially.

Another important observation to note here is that multi-label learning using *icsiboost* (10000 iterations are run) yields significantly lower accuracy than the others, the difference is much more conspicuous than in the previous scenario, where only DI data were used. As opposed to methods that directly predict the label combinations, binarization-based techniques such as *icsiboost*, which predict each label separately, usually rely on a threshold (0 by default) to decide the number of labels

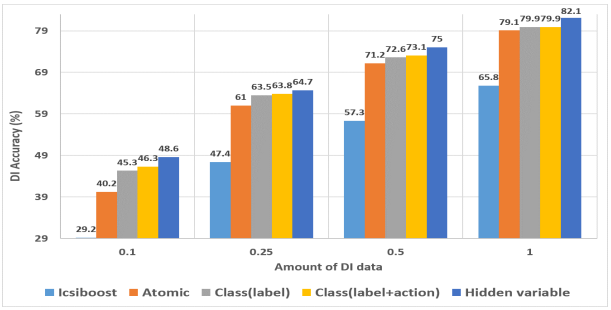

Figure 3: *Intent detection accuracy (%) on DI test set with models trained on SI data plus various amounts of DI data.*

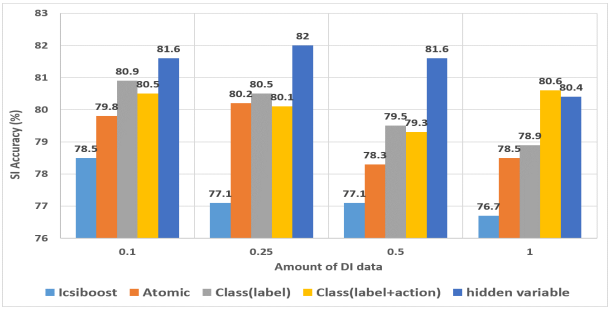

Figure 4: *Intent detection accuracy (%) on SI test set with models trained on SI data plus various amounts of DI data.*

to output – only labels scored higher than the threshold are predicted. Such a threshold can be optimized on the development set. In the previous experimental setup, since all sentences contain two intents, the best strategy is therefore always outputting two intents. Consequently, the difference in accuracy was not too obvious. However, when the classifier needs to handle both types of data, it is difficult to have a common threshold, resulting in much degraded performance. If we always predict one intent for SI sentences and two intents for DI sentences (*perfect* threshold), we have observed that the accuracy numbers could reach the same level of the atomic models, which is still worse than the proposed approaches.

It is worth pointing out that in addition to the superior classification accuracy, our hidden variable model induces quite reasonable segmentation (shown below, 0 and 1 indicate the intent the word belongs to).

- find_music#play_music: find(0) music(0) and(1) listen(1) to(1) it(1)
- find_movie#find_duration: display(0) foreign(0) films(0) how(1) long(1) is(1) the(1) longest(1)

6. Conclusions

In this paper, we proposed two types of techniques to improve the semantic classification of natural language sentences with multiple intents. Central to our techniques is to exploit shared information across different intent combinations. We showed that the class-based, and the hidden variable-based approaches both lead to substantially higher classification accuracy than the baseline approach in which the combinations are treated as atomic labels, as well as the binarization-based technique commonly used for multi-label learning.

7. References

[1] Gorin, A.L., Riccardi, G. and Wright, J.H., "How may I help you?", Speech Communication, 23(1):113–127, 1997.

[2] Gorin, A.L., Abella, A., Riccardi, G. and Wright, J.H., "Automated natural spoken dialogue", Computer, 35(4):51–56, 2002.

[3] Gupta, N., Tur, G., Hakkani-Tur, D., Bangalore, S., Riccardi, G. and Gilbert, M., "The AT&T spoken language understanding system", IEEE Transaction on Audio, Speech and Language Processing, 14(1):213–222, 2006.

[4] Young, S., Gasic, M., Keizer. S., Mairesse, F., Thomson, B. and Yu, K., "The Hidden Information State model: a practical framework for POMDP based spoken dialogue management", Computer Speech and Language, 24:150-174, 2010.

[5] Sarikaya, R., "Rapid bootstrapping of statistical spoken dialogoue systems", Speech Communication, 50(7):580–593, 2008.

[6] Tsoumakas, G. and Katakis, I., "Multi-label classification: An overview", International Journal of Data Warehousing & Mining, 3(3):1–13, 2007.

[7] Li, T. and Ogihara, M., "Detecting emotion in music", Proceedings of the International Symposium on Music Information Retrieval, 2003.

[8] Boutell, M., Luo, J., Shen, X. and Brown, C.M., "Learning multi-label scene classification", Pattern Recognition, 37(9):1757–1771, 2004.

[9] Rosenblatt, F., "The Perceptron – a perceiving and recognizing automaton", Tech. Rep., 1957.

[10] Goodman, J., "A bit of progress in language modeling", Computer Speech and Language., 1957.

[11] McCallum, A.K., "Multi-label text classification with a mixture model trained by EM", Proceedings of AAAI Workshop on Text Learning, 1999.

[12] Gunawardana, A., Mahajan, M., Acero, A. and Platt. J.C., "Hidden conditional random field for phone classification", Proceedings of International Conference on Speech Communication and Technology, 2005.

[13] Quattoni, A., Wang, S., Morency, L., Collins, M. and Darrel, T., "Hidden-state conditional random fields", IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007.

[14] Sun, X., Matsuzaki, T., Okahohara, D. and Tsuijii, J., "Latent variable Perceptron algorithm for structured classification", Proceedings of International Joint Conference on Artificial Intelligence, 1236–1242, 2009.

[15] Van der Maaten, L., Welling, M. and Saul, L., "Hidden-unit conditional random fields", Proceedings of International Conference on Artificial Intelligence & Statistics, 15:479–488, 2011.

[16] Schapire, R.E. and Singer, Y., "Boostexter: A boosting-based system for text categorization", Machine Learning, 39(2/3):135–168, 2000.

[17] Favre, B., and Hakkani-Tur, D. and Cuendet, S., "Icsiboost", Online:http://code.google.come/p/icsiboost, 2007.