Targeted Feature Dropout for Robust Slot Filling in Natural Language Understanding

Puyang Xu, Ruhi Sarikaya

Microsoft Corporation, Redmond WA 98052, USA

{puyangxu, ruhi.sarikaya}@microsoft.com

Abstract

In slot filling with conditional random field (CRF), the strong current word and dictionary features tend to swamp the effect of contextual features, a phenomenon also known as *feature undertraining*. This is a dangerous tradeoff especially when training data is small and dictionaries are limited in their coverage of the entities observed during testing. In this paper, we propose a simple and effective solution that extends the feature dropout algorithm, directly aiming at boosting the contribution from entity context. We show with extensive experiments that the proposed technique can significantly improve the robustness against unseen entities, without degrading performance on entities that are either seen or exist in the dictionary.

Index Terms: slot filling, feature dropout, spoken language understanding

1. Introduction

Slot filling plays an important role in spoken language understanding (SLU) systems. It is the task of extracting key bits of information (e.g. entities) from natural language sentences in order to formulate a query and fetch information from the knowledge back-end. In order to provide the correct response to the user request, the system relies heavily on accurate extraction of such semantic components (e.g. identifying the location entity for a weather inquiry, determining the intended recipient of a text message).

State-of-the-art slot filling relies on statistical machine learning techniques, in particular discriminative sequence models such as the conditional random field (CRF) [1]. The CRF model often outperforms generative models such as hidden markov models, partly due to its flexibility to deal with overlapping features. It is also believed to overcome the *label bias* problem [1] facing locally normalized models.

For discriminative models, rich feature sets are usually helpful given sufficient amount of training data. However, when training data is limited, there could be undesirable interactions among features – highly indicative features tend to diminish the contribution from other features which may as well provide valuable information. This problem has been described as *undertraining* [2], or *co-adaptation* [3].

For the task of slot filling, as we will demonstrate empirically later in the paper, such undertraining can be particularly harmful. To illustrate this point, suppose we want to learn the semantic pattern "tell `contact_name` that `message`" for a calling/texting application. While multiple types of features can help extracting the slots, the most consistent signals actually come from the context (e.g. the previous word is "tell", the next word is "that", etc.). However, such contextual features may get undertrained due to the following two reasons:

- Frequent repetition of same entities. Most entities appear multiple times, leading to insufficient tail representation. This will inflate the weight of current word features.
- Use of large entity dictionaries that cover significant proportions of the entities in the training data. This will inflate the weight of dictionary features.

As a result, the model tends to rely too much on (lexical) features extracted from the slot values and dictionary features to assign a tag rather than the context. Consequently, we often observe noticeable performance degradation at test time when the slot value does not exist in the training set or the entity dictionary.

The solutions to undertraining problems in general tend to have similar flavors. It often involves inducing variance in the feature representation. One simple approach is to generate corrupted versions of the existing training data [4]. However, this approach usually slows down the training process. Feature bagging is another way to deal with weight undertraining [5, 2]. Instead of training one single model with the complete feature set, multiple models are trained each with a subset of the features, forcing each model to rely only on partial information. The resulting models are combined at the end. Despite some success, such bagging techniques are often believed to be computationally expensive since multiple models are built. An alternative approach to inject feature noise is to formulate this as some form of feature regularization [6, 7, 8, 9, 10], in which only one model is trained with a regularized objective function, taking into account the variability of noisy features.

Feature dropout [3] is a recently proposed algorithm for improving the accuracy of the neural network based models. The dropout procedure corresponds to inducing *blackout* noise to the feature vector – for each update iteration, randomly disabling a subset of features by setting them to zero. This will force the rest of the features to rely on their own and maximize their contribution to the model, preventing harmful co-adaption and undertraining. Although the technique was initially proposed for neural networks, it may as well be utilized for other feature based models.

The dropout noise injected randomly can also be formulated as an explicit regularization term without actually corrupting the training data, leading to nice theoretic interpretations [8]. However, the regularized training objectives are usually more difficult to optimize, particularly for structured models [10], such as CRF.

In this work, we propose a low-overhead extension to the feature dropout technique, tailored for the slot filling task. The goal is to address the undertraining problem of contextual features in learning slot patterns. Our algorithm is simple yet effective, requiring no change of the objective function or the optimization algorithm.

The rest of the paper is organized as follows: We briefly describe the CRF model and the features used for slot filling in section 2. The dropout algorithm as well as our modification to it are described in section 3. In section 4, we present extensive experimental study of the undertraining problem for slot filling. We demonstrate how the proposed targeted feature dropout technique can alleviate undertraining and improve the robustness of the model against unseen slot entities.

2. CRF for Slot Filling

Slot filling can be formulated as a sequence labeling task, which can be modeled effectively using CRF. CRF is a globally normalized discriminative model, specifying the conditional distribution of the hidden label sequence y, given the observation x,

$$P(y|x) = \frac{\exp \sum_i s(y_i, y_{i-1}, x)}{Z}. \quad (1)$$

As shown in (1), the probability of the label sequence is proportional to the exponential sum of the feature scores accumulated at each position (indexed by i).

In the context of slot filling, the hidden label is essentially the slot label, and the observation is the word sequence. The task then becomes assigning the correct slot label to each word in the sentence. There are some standard features that are conventionally used for slot filling, such as slot tag transition features, current word features and left/right n-grams. Entity dictionary is another important source of information, it is also frequently used for slot tagging models – whenever a word or phrase matches an entity in the dictionary, the dictionary feature will be activated for all words within the matched segment.

As we described, multiple sources of information contribute to the feature score $s(y_i, y_{i-1}, x)$ at each word position. When the current word feature or the dictionary feature are highly frequent and indicative, the model will not be able to generate sufficiently large gradient to update other features such as contextual words. Consequently, if a slot entity at test time has not been seen in training and also does not exist in the dictionary, it is very likely that the model will miss the slot.

3. Targeted Feature Dropout

The original dropout technique removes features randomly – for each training instance, only a random subset of the features will be activated. While it can be perceived as a general treatment to the undertraining problem, it is not specifically directed at the problem we are facing in slot filling.

In slot filling, the undertraining problem is much more specific – the contextual features may not get large enough weights due to the strong influence of the current word feature and the dictionary feature. The negative impact of such undertraining is also much more measurable – if a slot entity is unseen, the chance of a detection error becomes much higher.

To address this issue, we propose the following modified version of the dropout procedure.

1. Identify slot types that are potentially vulnerable to unseen entities. They are generally slot types whose values are hard to enumerate (e.g. person names, movie names). In other words, it is difficult to be able to observe all possible slot values with finite amount of training data and finite size entity dictionaries. Moreover, as we will later demonstrate, the slot types are under the most risk if (1) There is not sufficient "tail" representation in the training data. (2) The dictionary covers a significant proportion of the entities in training, but high number of out-of-dictionary entities are expected at test time.

2. For each training instance, extract all the active features. If the feature function depends on the word inside of the slot types identified in step 1, disable it with probability p. The disabled feature is not used for model inference and its weight is not updated.

3. At test time, no feature is dropped. All active features contribute to the model score.

We now give a concrete example to further illustrate this process: Suppose we identified `contact_name` to be a susceptible slot. During training, if we see a sentence "Text John I'll be late" which contains a `contact_name` slot (John), we will drop all features that depend on "John" (unigram feature "john", bigram feature "text john", "john i'll", etc) with a probability. The intuition is that, while "John" is indeed a person's name and very likely to exist in the dictionary, it could well be another name which we have not seen in training or in the dictionary. So occasionally we will ask the model to ignore these obvious cues from the entity itself and learn from the surrounding words, which arguably are more stable cues especially for natural language sentences, where entities often appear in certain contexts.

The standard feature dropout is unlikely to help us reach our goal, since it treats all features the same and drops them with equal probability. Feature bagging [5, 2] is an alternative to try, but the complexity of building multiple models and combining them can be nontrivial.

4. Experimental Results

For all the experiments presented in this section, the CRF models are trained using the perceptron algorithm. No explicit regularization is performed, but this is compensated by parameter averaging which is believed to have some regularization effect [11]. The feature set includes tag transition features, current word features, surrounding word features within the ± 2 context, and entity dictionary features.

4.1. On ATIS Task

The ATIS dataset has been used extensively in the SLU research community as a benchmark dataset. It consists of 4978 spoken language queries for training and 893 queries for testing from the air travel reservation domain. Another 491 queries are set aside as development data. While the entire dataset contains more than 80 types of slots, we mainly focus on the two slots related to city names, namely `fromloc.city_name` (departure city) and `toloc.city_name` (destination city), which are the two most frequent slot types on the dataset, covering 50% of all entities.

For these two location slot types, it is very likely that at test time we will observe entities not seen in training. Given the massive number of location names in the world, it is hardly feasible to even list all of them, not to mention using them in training (either as dictionary or specific training instance). On the ATIS dataset, this problem is however nonexistent – there is no new city name on the test data. In other words, everything has been observed in training.

While most slot filling work tends to focus on reporting numbers on the original ATIS test set, we would like to investigate how the tagging performance may change if entities are unseen at test time. To enable such analysis, we create an ad-

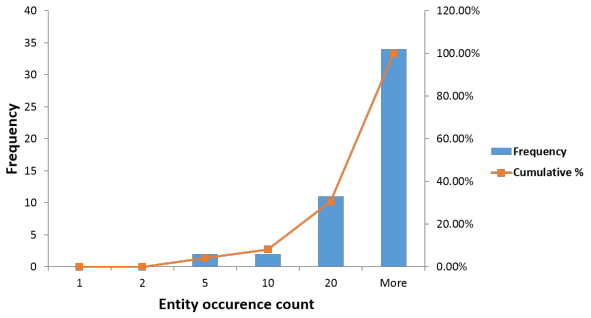

Figure 1: *Histogram of the city name entities on ATIS data*

ditional test set based on the original test set by replacing each word in city name entities with a special token indicating unseen words – we call it the unseen test set.

Table 1: *Slot F1 scores on original test and unseen test set. (No dictionary features)*

Slot type	Original test	Unseen test
fromloc.city_name	97.40	93.27
toloc.city_name	96.98	80.66
OVERALL	92.12	86.26

Table 1 shows the F1 scores on the original test set and the unseen test set. The degradation due to unseen entities is quite significant, particularly for the `toloc.city_name` slot (more than 15% absolute). Note that in these experiments, no dictionary feature is used. The undertraining problem is solely due to the strong current word features. Further investigation suggests its connection with some distributional properties of the ATIS dataset.

As the histogram in Figure 1 demonstrates, the city name slot has an extremely short tail distribution in the data. Almost all of the names appear more than 5 times, and the majority of the names appear more than 20 times! Such a distribution tends to significantly reduce the impact of contextual features for detecting city names, leading to the degradation shown in Table 1 when unseen names are encountered at test time.

With the targeted dropout technique we described, this problem can be largely resolved. In Figure 2, we vary the dropout probability, and plot the F1 scores of the two targeted slots on the two test sets. It is clear from the plot that the proposed algorithm can improve the robustness against unseen entities, particularly for the `toloc.city_name` slot (from lower 80% to around 94%). Although for both types of slots, it appears that the gains come at the cost of some small degradation of the performance on the original test set, the improvement on unseen entities tends to be more significant. Considering the tens of thousands of city names outside of the small test set, the results on the unseen test set are arguably more informative.

Meanwhile, the random dropout, in which all features are removed with the same probability, does not appear to be particularly useful for detecting unseen entities. While there are some gains on `toloc.city_name` as we reach higher dropout probabilities, the magnitude of the improvement is much smaller (Figure 3).

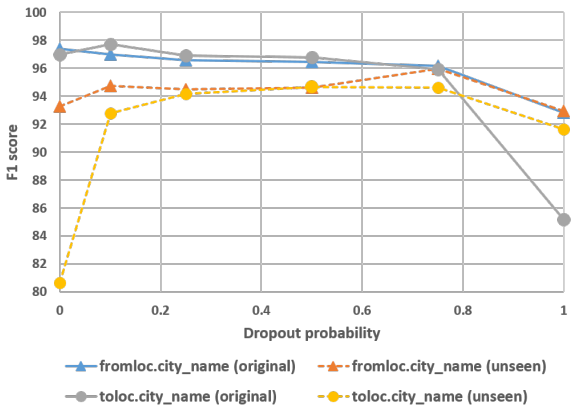

Figure 2: *F1 scores of unseen city names with* **targeted** *feature dropout. Solid line: original test set; Dashed line: unseen test set; Circle: toloc.city_name; Triangle: fromloc.city_name*

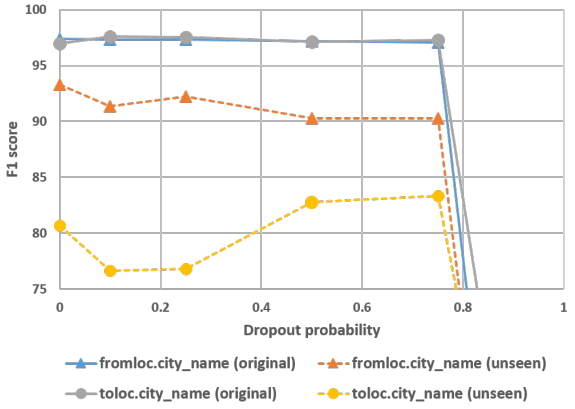

Figure 3: *F1 scores of unseen city names with* **random** *feature dropout. Note that with dropout probability 1, all features will be dropped.*

4.2. On Weather Domain Task

The performance degradation that we demonstrated in Table 1 was mainly due to the frequent repetition of the city name entities in the training data. In most of the other datasets that we deal with, the tail distribution of entities is usually much longer. In these cases, the main risk of undertraining stems from the use of entity dictionaries. In this set of experiments, we investigate how dictionary features can impact unseen entity tagging, and also how the targeted feature dropout can alleviate such a problem.

We have a larger dataset on weather domain, intended for handling spoken language weather related inquiries. It consists of approximately 20K training queries, and about 2.5K queries each for tuning and testing respectively.

We still focus on one location related slot called `absolute_location`, which could be any location the user may ask a weather related question about. It is also the most frequent entity type in the data, covering over 50% of all the entities. On this dataset, the `absolute_location` slot has

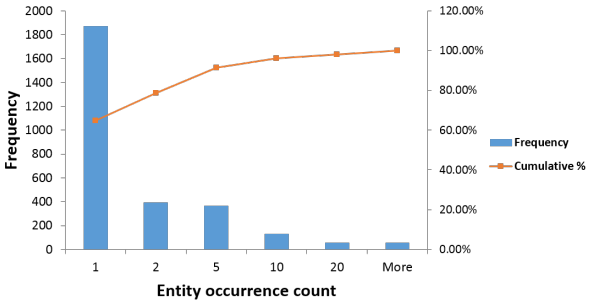

Figure 4: *Histogram of the absolute_location slot on the weather data*

a much richer representation. As we can see from Figure 4, the distribution of the occurrence counts looks much more natural – over 60% of the entities are singletons, and only a small number of entities appear more than 20 times. Under such conditions, the performance loss due to unseen entities is no longer significant, although it does exists. More importantly, the proposed technique can still bring accuracy improvement (Figure 5). While dropping features still tends to degrade the performance on the original test set (solid line), the tradeoff is in fact quite favorable with small dropout probabilities (e.g. $p < 0.4$).

The addition of entity dictionaries can further improve the overall tagging accuracy for `absolute_location`. To thoroughly measure the impact of adding entity dictionaries, the following set of experiments are conducted in a controlled fashion.

- We simulate entity dictionaries with different coverage levels of our dataset. Note that the weight for the dictionary feature is affected by its coverage on the *training* set. Only with a sufficiently high weight, we can benefit from higher coverage on the test set.
- In addition to the original test set and the unseen test set, we create a third test set in which the location entities are all in the dictionary, but have not been observed on the training data. In other words, only dictionary features and contextual features will be active for these entities. we call it the *in-dictionary* test set.

In essence, the two additional test sets correspond to two completely opposite operating conditions: For the unseen test set, dictionary features are of no use, but for the in-dictionary test set, they are exploited to the maximum. Therefore, having these two test sets essentially allows us to quantify this tradeoff between contextual features and dictionary features. In general, improving the in-dictionary test set tends to degrade the unseen test set. However, with the proposed technique, such a tradeoff can be improved.

We vary the dictionary coverage on the original dataset from 0% to 100%. At each coverage level, the F1 scores of `absolute_location`, with and without the proposed dropout technique are demonstrated in Table 2. Note that the dropout probability here is fixed at 0.25 for all conditions – from previous experiments, 0.25 tends to be a safe and effective rate of dropping features.

The tradeoff between the performance on the unseen and in-dictionary test sets is quite obvious. Without dropout ($p = 0$), as the coverage level increases, the F1 score on the in-dictionary test set improves from 94.4 to 99.5. Meanwhile, the degradation on the unseen test set is substantial – at 100% coverage, we completely lose the ability to detect unseen entities. Such

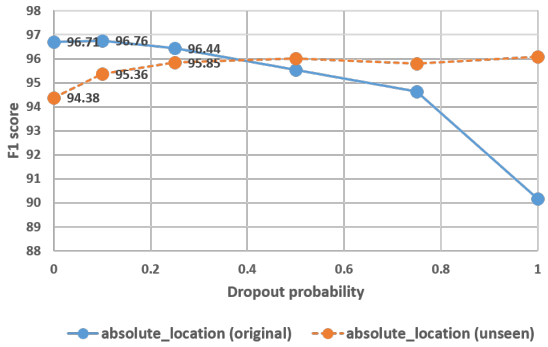

Figure 5: *F1 scores of absolute_location with the proposed targeted feature dropout (No dictionary features).*

Table 2: *Slot F1 scores of absolute_location on the original, unseen, in-dictionary test sets, with and without the proposed targeted feature dropout. Dropout probability fixed at 0.25. Dictionary coverage on the original dataset is varied.*

Coverage	Original		Unseen		In-dictionary	
	p=0	p=.25	p=0	p=.25	p=0	p=.25
0%	96.7	96.4	94.4	95.9	94.4	95.9
25%	96.7	96.9	92.9	95.9	97.3	97.1
50%	96.5	96.8	89.4	95.0	97.7	97.2
75%	97.2	97.0	82.6	94.9	98.4	97.1
100%	98.8	97.2	1.3	94.9	99.5	97.6

a tradeoff sometimes may not be desirable, especially if we do not expect the dictionary to cover all the entities we can possibly observe. To maximize the benefit of the entity dictionary, sufficient coverage on the training set is required, but as soon as we increase the coverage, we start hurting the performance on unseen entities.

The main takeaway from Table 2 is that, regardless of the dictionary coverage, even in the absence of dictionaries, the proposed dropout technique can always lead to a better operating point. On the unseen test set, $p = .25$ (with dropout) appears to consistently outperform $p = 0$ (without dropout), by margins which are usually much more significant than the negligible degradation we see on the in-dictionary and the original test sets. In other words, the benefit of using large entity dictionaries can be mostly preserved, while the risk of hurting unseen entities is essentially eliminated.

5. Conclusions

In this paper, we focused on the undertraining problem of contextual features in slot entity detection. We described a low-overhead extension to the feature dropout algorithm to mitigate the swamping effect of strong current word features and dictionary features. We presented extensive experimental results to illustrate the undertraining problem and the performance degradation on unseen entities, and demonstrated how the proposed technique can improve the robustness against them, without sacrificing the ability to detect seen entities.

6. References

[1] Lafferty, J. and McCallum, A. and Pereira, F., "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data", ICML, 2001.

[2] Sutton, C. and Sindelar, M. and McCallum, A., "Feature bagging: Preventing weight undertraining in structured discriminative learning", in CIIR Technical Report IR-402, University of Masschusetts, 2005.

[3] Hinton, G. and Srivastava, N. and Krizhevsky, A. and Sutskever, I. and Salakhutdinov, R., "Improving Neural Networks by Preventing Co-adaptation of Feature Detectors", arXiv, 2012.

Chris J.C. Burges and Bernhard Scholkopf.

[4] Burges, C. and Scholkopf, B., "Improving the accuracy and speed of support vector machines", NIPS, 1997.

[5] Bryll, R. and Gutierrez-Osuna, R. and Quek, F., "Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets", Pattern recognition, 36(6):12911302, 2003.

Bishop:95, Rifai:11, Wager:13, Maaten:13, Wang:13

[6] Bishop, C., "Training with noise is equivalent to Tikhonov regularization", Neural computation, 7(1):108116.

[7] Rifai, S. and Glorot, X. and Bengio, Y. and and Vincent, P., "Adding noise to the input of a model trained with a regularized objective", arXiv, 2011.

[8] Wager, S. and Wang, S. and Liang, P., "Dropout training as adaptive regularization", arXiv, 2013.

[9] Van der Maaten, L. and Chen, M. and Tyree, S. and Weinberger, K., "Learning with marginalized corrupted features", ICML, 2013.

[10] Wang, S. and Wang, M. and Wager, S. and Liang, P. and Manning, C., "Feature Noising for Log-linear Structured Prediction", EMNLP, 2013.

[11] Freund, Y. and Schapire, R., "Large margin classification using the perceptron algorithm", Machine Learning, 37(3):277296, 1999.