N-best Reranking by Multitask Learning

Kevin Duh Katsuhito Sudoh Hajime Tsukada Hideki Isozaki Masaaki Nagata
NTT Communication Science Laboratories
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0237, Japan
{kevinduh,sudoh,tsukada,isozaki}@cslab.kecl.ntt.co.jp
nagata.masaaki@lab.ntt.co.jp

Abstract

We propose a new framework for N-best reranking on sparse feature sets. The idea is to reformulate the reranking problem as a Multitask Learning problem, where each N-best list corresponds to a distinct task.

This is motivated by the observation that N-best lists often show significant differences in feature distributions. Training a single reranker directly on this heterogenous data can be difficult.

Our proposed meta-algorithm solves this challenge by using multitask learning (such as ℓ_1/ℓ_2 regularization) to discover common feature representations across N-best lists. This meta-algorithm is simple to implement, and its modular approach allows one to plug-in different learning algorithms from existing literature. As a proof of concept, we show statistically significant improvements on a machine translation system involving millions of features.

1 Introduction

Many natural language processing applications, such as machine translation (MT), parsing, and language modeling, benefit from the N-best reranking framework (Shen et al., 2004; Collins and Koo, 2005; Roark et al., 2007). The advantage of N-best reranking is that it abstracts away the complexities of first-pass decoding, allowing the researcher to try new features and learning algorithms with fast experimental turnover.

In the N-best reranking scenario, the training data consists of sets of hypotheses (i.e. N-best lists) generated by a first-pass system, along with their labels. Given a new N-best list, the goal is to rerank it such that the best hypothesis appears near the top of the list. Existing research have focused on training a *single* reranker directly on the entire data. This approach is reasonable if the data is homogenous, but it fails when features vary significantly across different N-best lists. In particular, when one employs *sparse* feature sets, one seldom finds features that are simultaneously active on multiple N-best lists.

In this case, we believe it is more advantageous to view the N-best reranking problem as a *multitask learning* problem, where each N-best list corresponds to a distinct task. Multitask learning, a subfield of machine learning, focuses on how to effectively train on a set of different but related datasets (tasks). Our heterogenous N-best list data fits nicely with this assumption.

The contribution of this work is three-fold:

1. We introduce the idea of viewing N-best reranking as a multitask learning problem. This view is particularly apt to any general reranking problem with sparse feature sets.
2. We propose a simple meta-algorithm that first discovers common feature representations across N-bests (via multitask learning) before training a conventional reranker. Thus it is easily applicable to existing systems.
3. We demonstrate that our proposed method outperforms the conventional reranking approach on a English-Japanese biomedical machine translation task involving millions of features.

The paper is organized as follows: Section 2 describes the feature sparsity problem and Section 3 presents our multitask solution. The effectiveness of our proposed approach is validated by experiments demonstrated in Section 4. Finally, Sections 5 and 6 discuss related work and conclusions.

2 The Problem of Sparse Feature Sets

For concreteness, we will describe N-best reranking in terms of machine translation (MT), though our approach is agnostic to the application. In MT reranking, the goal is to translate a foreign language sentence f into an English sentence e by picking from a set of likely translations. A standard approach is to use a linear model:

$$\hat{e} = \arg\max_{e \in N(f)} \mathbf{w}^T \cdot \mathbf{h}(e,f) \tag{1}$$

where $\mathbf{h}(e,f)$ is a D-dimensional feature vector, $\mathbf{w}$ is the weight vector to be trained, and $N(f)$ is the set of likely translations of f, i.e. the N-best list. The feature $\mathbf{h}(e,f)$ can be any quantity defined in terms of the sentence pair, such as translation model and language model probabilities.

Here we are interested in situations where the feature definitions can be quite sparse. A common methodology in reranking is to first design *feature templates* based on linguistic intuition and domain knowledge. Then, numerous features are instantiated based on the training data seen. For example, the work of (Watanabe et al., 2007) defines feature templates based on bilingual word alignments, which lead to extraction of heavily-lexicalized features of the form:

$$h(e,f) = \begin{cases} 1 & \text{if foreign word "Monsieur"} \\ & \text{and English word "Mr."} \\ & \text{co-occur in } e,f \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

One can imagine that such features are sparse because it may only fire for input sentences that contain the word "Monsieur". For all other input sentences, it is an useless, inactive feature.

Another common feature involves word ngram templates, for example:

$$h(e,f) = \begin{cases} 1 & \text{if English trigram} \\ & \text{"Mr. Smith said" occurs in } e \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

In this case, all possible trigrams seen in the N-best list are extracted as features. One can see that this kind of feature can be very sensitive to the first-pass decoder: if the decoder has loose reordering constraints, then we may extract exponentially many nonsense ngram features such as "Smith said Mr." and "said Smith Mr.". Granted, the reranker training algorithm may learn that these nonsense ngrams are indicative of poor hypotheses, but it is unlikely that the exact same nonsense ngrams will appear given a different test sentence.

In summary, the following issues compound to create extremely sparse feature sets:

1. Feature templates are heavily-lexicalized, which causes the number of features to grow unbounded as the the amount of data increases.

2. The input (f) has high variability (e.g. large vocabulary size), so that features for different inputs are rarely shared.

3. The N-best list output also exhibits high variability (e.g. many different word reorderings). Larger N may improve reranking performance, but may also increase feature sparsity.

When the number of features is too large, even popular reranking algorithms such as SVM (Shen et al., 2004) and MIRA (Watanabe et al., 2007; Chiang et al., 2009) may fail. Our goal here is to address this situation.

3 Proposed Reranking Framework

In the following, we first give an intuitive comparison between single vs. multiple task learning (Section 3.1), before presenting the general meta-algorithm (Section 3.2) and particular instantiations (Section 3.3).

3.1 Single vs. Multiple Tasks

Given a set of I input sentences $\{f^i\}$, the training data for reranking consists of a set of I N-best lists $\{(\mathbf{H}^i, \mathbf{y}^i)\}_{i=1,\ldots,I}$, where $\mathbf{H}^i$ are features and $\mathbf{y}^i$ are labels.

To clarify the notation:[1] for an input sentence f^i, there is a N-best list $N(f^i)$. For a N-best list $N(f^i)$, there are N feature vectors corresponding to the N hypotheses, each with dimension D. The collection of feature vectors for $N(f^i)$ is represented by $\mathbf{H}^i$, which can be seen as a $D \times N$ matrix. Finally, the N-dimensional vector of labels $\mathbf{y}^i$ indicates the translation quality of each hypothesis in $N(f^i)$. The purpose of the reranker training algorithm is to find good parameters from $\{(\mathbf{H}^i, \mathbf{y}^i)\}$.

[1]Generally we use bold font $\mathbf{h}$ to represent a vector, bold-capital font $\mathbf{H}$ to represent a matrix. Script h and $h(\cdot)$ may be scalar, function, or sentence (depends on context).

The conventional method of training a single reranker (single task formulation) involves optimizing a generic objective such as:

$$\arg\min_{\mathbf{w}} \sum_{i=1}^{I} L(\mathbf{w}, \mathbf{H}^i, \mathbf{y}^i) + \lambda\Omega(\mathbf{w}) \quad (4)$$

where $\mathbf{w} \in \mathbb{R}^D$ is the reranker trained on all lists, and $L(\cdot)$ is some loss function. $\Omega(\mathbf{w})$ is an optional regularizer, whose effect is traded-off by the constant λ. For example, the SVM reranker for MT (Shen et al., 2004) defines $L(\cdot)$ to be some function of sentence-level BLEU score, and $\Omega(\mathbf{w})$ to be the large margin regularizer.[2]

On the other hand, multitask learning involves solving for multiple weights, $\mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^I$, one for each N-best list. One class of multitask learning algorithms, Joint Regularization, solves the following objective:

$$\arg\min_{\mathbf{w}^1,..,\mathbf{w}^I} \sum_{i=1}^{I} L(\mathbf{w}^i, \mathbf{H}^i, \mathbf{y}^i) + \lambda\Omega(\mathbf{w}^1, .., \mathbf{w}^I) \quad (5)$$

The loss decomposes by task but the joint regularizer $\Omega(\mathbf{w}^1, .., \mathbf{w}^I)$ couples together the different weight parameters. The key is to note that multiple weights allow the algorithm to fit the heterogenous data better, compared to a single weight vector. Yet these weights are still tied together so that some information can be shared across N-best lists (tasks).

One instantiation of Eq. 5 is ℓ_1/ℓ_2 regularization: $\Omega(\mathbf{w}^1, .., \mathbf{w}^I) \triangleq ||\mathbf{W}||_{1,2}$, where $\mathbf{W} = [\mathbf{w}^1|\mathbf{w}^2|\dots|\mathbf{w}^I]^T$ is a I-by-D matrix of stacked weight vectors. The norm is computed by first taking the 2-norm on columns of $\mathbf{W}$, then taking a 1-norm on the resulting D-length vector. This encourages the optimizer to choose a small subset of features that are useful across all tasks.

For example, suppose two different sets of weight vectors $\mathbf{W_a}$ and $\mathbf{W_b}$ for a 2 lists, 4 features reranking problem. The ℓ_1/ℓ_2 norm for $\mathbf{W_a}$ is 14; the ℓ_1/ℓ_2 norm for $\mathbf{W_b}$ is 12. If both have the same loss $L(\cdot)$ in Eq. 5, the multitask optimizer would prefer $\mathbf{W_b}$ since more features are shared:

$$\mathbf{W_a}: \begin{array}{c} \left[\begin{array}{cccc} 4 & 0 & 0 & 3 \\ 0 & 4 & 3 & 0 \end{array}\right] \\ \begin{array}{cccc} 4 & 4 & 3 & 3 \rightarrow 14 \end{array} \end{array} \quad \mathbf{W_b}: \begin{array}{c} \left[\begin{array}{cccc} 4 & 3 & 0 & 0 \\ 0 & 4 & 3 & 0 \end{array}\right] \\ \begin{array}{cccc} 4 & 5 & 3 & 0 \rightarrow 12 \end{array} \end{array}$$

[2]In MT, evaluation metrics like BLEU do not exactly decompose across sentences, so for some training algorithms this loss is an approximation.

3.2 Proposed Meta-algorithm

We are now ready to present our general reranking meta-algorithm (see Algorithm 1), termed Reranking by Multitask Learning (RML).

Algorithm 1 Reranking by Multitask Learning

```
Input: N-best data $\{(\mathbf{H}^i, \mathbf{y}^i)\}_{i=1,\dots,I}$
Output: Common feature representation $h_c(e,f)$
    and weight vector $\mathbf{w}_c$
1: [optional] RandomHashing($\{\mathbf{H}^i\}$)
2: $\mathbf{W}$ = MultitaskLearn($\{(\mathbf{H}^i, \mathbf{y}^i)\}$)
3: $h_c$ = ExtractCommonFeature($\mathbf{W}$)
4: $\{\mathbf{H}_c^i\}$ = RemapFeature($\{\mathbf{H}^i\}, h_c$)
5: $\mathbf{w}_c$ = ConventionalReranker($\{(\mathbf{H}_c^i, \mathbf{y}^i)\}$)
```

The first step, random hashing, is optional. Random hashing is an effective trick for reducing the dimension of sparse feature sets without suffering losses in fidelity (Weinberger et al., 2009; Ganchev and Dredze, 2008). It works by collapsing random subsets of features. This step can be performed to speed-up multitask learning later. In some cases, the original feature dimension may be so large that hashed representations may be necessary.

The next two steps are key. A multitask learning algorithm is run on the N-best lists, and a common feature space shared by all lists is extracted. For example, if one uses the multitask objective of Eq. 5, the result of step 2 is a set of weights $\mathbf{W}$. ExtractCommonFeature($\mathbf{W}$) then returns the feature id's (either from original or hashed representation) that receive nonzero weight in any of $\mathbf{W}$.[3] The new features $h_c(e,f)$ are expected to have lower dimension than the original features $h(e,f)$. Section 3.3 describes in detail different multitask methods that can be plugged-in to this step.

The final two steps involve a conventional reranker. In step 4, we remap the N-best list data according to the new feature representations $h_c(e,f)$. In step 5, we train a conventional reranker on this common representation, which by now should have overcome sparsity issues. Using a conventional reranker at the end allows us to exploit existing rerankers designed for specific NLP applications. In a sense, our meta-algorithm simply involves a change of representation for the conventional reranking scenario, where the

[3]For example in $\mathbf{W_b}$, features 1-3 have nonzero weights and are extracted. Feature 4 is discarded.

new representation is found by multitask methods which are well-suited to heterogenous data.

3.3 Multitask Objective Functions

Here, we describe various multitask methods that can be plugged in Step 2 of Algorithm 1. Our goal is to demonstrate that a wide range of existing methods from the multitask learning literature can be brought to our problem. We categorize multitask methods into two major approaches:

1. Joint Regularization: Eq. 5 is an example of joint regularization, with ℓ_1/ℓ_2 norm being a particular regularizer. The idea is to use the regularizer to ensure that the learned functions of related tasks are close to each other. The popular ℓ_1/ℓ_2 objective can be optimized by various methods, such as boosting (Obozinski et al., 2009) and convex programming (Argyriou et al., 2008). Yet another regularizer is the ℓ_1/ℓ_∞ norm (Quattoni et al., 2009), which replaces the 2-norm with a max.

One could also define a regularizer to ensure that each task-specific $\mathbf{w}^i$ is close to some average parameter, e.g. $\sum_i ||\mathbf{w}^i - \mathbf{w}^{avg}||_2$. If we interpret $\mathbf{w}^{avg}$ as a prior, we begin to see links to **Hierarchical Bayesian** methods for multitask learning (Finkel and Manning, 2009; Daume, 2009).

2. Shared Subspace: This approach assumes that there is an underlying feature subspace that is common to all tasks. Early works on multitask learning implement this by neural networks, where different tasks have different output layers but share the same hidden layer (Caruana, 1997).

Another method is to write the weight vector as two parts $\mathbf{w} = [\mathbf{u}; \mathbf{v}]$ and let the task-specific function be $\mathbf{u}^T \cdot \mathbf{h}(e, f) + \mathbf{v}^T \cdot \Theta \cdot \mathbf{h}(e, f)$ (Ando and Zhang, 2005). Θ is a $D' \times D$ matrix that maps the original features to a subspace common to all tasks. The new feature representation is computed by the projection $\mathbf{h}_c(e, f) \triangleq \Theta \cdot \mathbf{h}(e, f)$.

Multitask learning is a vast field and relates to areas like collaborative filtering (Yu and Tresp, 2005) and domain adaptation. Most methods assume some common representation and is thus applicable to our framework. The reader is urged to refer to citations in, e.g. (Argyriou et al., 2008) for a survey.

4 Experiments and Results

As a proof of concept, we perform experiments on a MT system with millions of features. We use a hierarchical phrase-based system (Chiang, 2007) to generate N-best lists (N=100). Sparse features used in reranking are extracted according to (Watanabe et al., 2007). Specifically, the majority are lexical features involving joint occurrences of words within the N-best lists and source sentences.

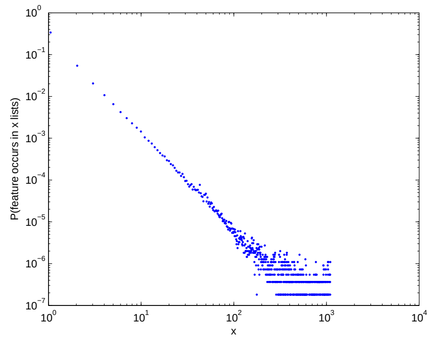

Figure 1: This log-log plot shows that there are many rare features and few common features. The probability that a feature occurs in x number of N-best lists behaves according to the power-law $x^{-\alpha}$, where $\alpha = 2.28$.

It is worth noting that the fact that the first pass system is a hierarchical system is not essential to the feature extraction step; similar features can be extracted with other systems as first-pass, e.g. a phrase-based system. That said, the extent of the feature sparsity problem may depend on the performance of the first-pass system.

We experiment with medical domain MT, where large numbers of technical vocabulary cause sparsity challenges. Our corpora consists of English abstracts from PubMed[4] with their Japanese translations. The first-pass system is built on hierarchical phrases extracted from 17k sentence pairs and target (Japanese) language models trained on 800k medical-domain sentences. For our reranking experiments, we used 500 lists as the training set[5], 500 lists as held-out, and another 500 for test.

4.1 Data Characteristics

We present some statistics to illustrate the feature sparsity problem: From 500 N-best lists, we extracted a total of 2.4 million distinct features. By type, 75% of these features occur in *only one* N-best list in the dataset. Less than 3% of features

[4]A database of the U.S. National Library of Medicine.

[5]In MT, training data for reranking is sometimes referred to as “dev set” to distinguish from the data used in first-pass. Also, while the 17k bitext may seem small compared to other MT work, we note that 1st pass translation quality (around 28 BLEU) is high enough to evaluate reranking methods.

occur in ten or more lists. The distribution of feature occurrence is clearly Zipfian, as seen in the power-law plot in Figure 1.

We can also observe the *feature growth rate* (Table 1). This is the number of new features introduced when an additional N-best list is seen. It is important to note that on average, 2599 new features are added everytime a new N-best list is seen. This is as much as $2599/4188 = 62\%$ of the active features. Imagine an online training algorithm (e.g. MIRA or perceptron) on this kind of data: whenever a loss occurs and we update the weight vector, less than half of the weight vector update applies to data we have seen thus far. Herein lies the potential for overfitting.

From observing the feature grow rate, one may hypothesize that adding large numbers of N-best lists to the training set (500 in the experiments here) may not necessarily improve results. While adding data potentially improves the estimation process, it also increases the feature space dramatically. Thus we see the need for a feature extraction procedure.

(Watanabe et al., 2007) also reports the possibility of overfitting in their dataset (Arabic-English newswire translation), especially when domain differences are present. Here we observe this tendency already on the same domain, which is likely due to the highly-specialized vocabulary and the complex sentence structures common in research paper abstracts.

4.2 MT Results

Our goal is to compare different feature representations in reranking: The **baseline** reranker uses the original sparse feature representation. This is compared to feature representations discovered by three different multitask learning methods:

- Joint Regularization (Obozinski et al., 2009)
- Shared Subspace (Ando and Zhang, 2005)
- Unsupervised Multitask Feature Selection (Abernethy et al., 2007).[6]

We use existing implementations of the above methods.[7] The conventional reranker (Step 5, Algorithm 1) used in all cases is SVM^{rank}.[8] Our initial experiments show that the SVM baseline performance is comparable to MIRA training, so we use SVM throughout. The labels for the SVM are derived as in (Shen et al., 2004), where top 10% of hypotheses by smoothed sentence-BLEU is ranked before the bottom 90%. All multitask learning methods work on hashed features of dimension 4000 (Step 1, Algorithm 1). This speeds up the training process.

All hyperparameters of the multitask method are tuned on the held-out set. In particular, the most important is the number of common features to extract, which we pick from $\{250, 500, 1000\}$.

Table 2 shows the results by BLEU (Papineni et al., 2002) and PER. The Oracle results are obtained by choosing the best hypothesis per N-best list by sentence-level BLEU, which achieved 36.9 BLEU in both Train and Test. A summary of our observations is:

1. The baseline (All sparse features) overfits. It achieves the oracle BLEU score on the train set (36.9) but performs poorly on the test (28.6).

2. Similar overfitting occurs when traditional ℓ_1 regularization is used to select features on

Nbest id	**#NewFt**	**#SoFar**	**#Active**
1	3900	3900	3900
2	7535	11435	7913
3	6078	17513	7087
4	3868	21381	4747
5	1896	23277	2645
6	3542	26819	4747
....			
100	2440	289118	4299
101	1639	290757	2390
102	3468	294225	4755
103	2350	296575	3824
Average	2599	–	4188

Table 1: Feature growth rate: For N-best list i in the table, we have (#NewFt = number of new features introduced since N-best $i - 1$) ; (#SoFar = Total number of features defined so far); and (#Active = number of active features for N-best i). E.g., we extracted 7535 new features from N-best 2; combined with the 3900 from N-best 1, the total features so far is 11435.

[6]This is not a standard multitask algorithm since most multitask algorithms are supervised. We include it to see if unsupervised or semi-supervised multitask algorithms is promising. Intuitively, the method tries to select subsets of features that are correlated across multiple tasks using random sampling (MCMC). Features that co-occur in different tasks form a high probability path.

[7]Available at http://multitask.cs.berkeley.edu

[8]Available at http://svmlight.joachims.org

the sparse feature representation[9]. ℓ_1 regularization is a good method of handling sparse features for classification problems, but in reranking the lack of tying between lists makes this regularizer inappropriate. A small set of around 1200 features are chosen: they perform well independently on each task in the training data, but there is little sharing with the test data.

3. All three multitask methods obtained features that outperformed the baseline. The BLEU scores are 28.8, 28.9, 29.1 for Unsupervised Feature Selection, Joint Regularization, and Shared Subspace, respectively, which all outperform the 28.6 baseline. All improvements are statistically significant by bootstrap sampling test (1000 samples, $p < 0.05$) (Zhang et al., 2004).

4. Shared Subspace performed the best. We conjecture this is because its feature projection can create new feature combinations that is more expressive than the feature selection used by the two other methods.

5. PER results are qualitatively similar to BLEU results.

6. As a further analysis, we are interested in seeing whether multitask learning extracts novel features, especially those that have low frequency. Thus, we tried an additional feature representation (feature threshold) which only keeps features that occur in more than x N-bests, and concatenate these high-frequency features to the multitask features. The feature threshold alone achieves nice BLEU results (29.0 for $x > 10$), but the combination outperforms it by statistically significant margins (29.3-29.6). This implies that multitask learning is extracting features that complement well with high frequency features.

For the multitask features, improvements of 0.2 to 1.0 BLEU are modest but *consistent*. Figure 2 shows the BLEU of bootstrap samples obtained as part of the statistical significance test. We see that **multitask** almost never underperform **baseline** in any random sampling of the data. This implies that the proposed meta-algorithm is very stable, i.e. it is not a method that sometimes improves and sometimes degrades.

[9]Optimized by the Vowpal Wabbit toolkit: http://hunch.net/vw/

Finally, a potential question to ask is: what kinds of features are being selected by the multitask learning algorithms? We found that that two kinds of features are usually selected: one is general features that are not lexicalized, such as "count of phrases", "count of deletions/insertions", "number of punctuation marks". The other kind is lexicalized features, such as those in Equations 2 and 3, but involving functions words (like the Japanese characters "wa", "ga", "ni", "de") or special characters (such as numeral symbol and punctuation). These are features that can be expected to be widely applicable, and it is promising that multitask learning is able to recover these from the millions of potential features. [10]

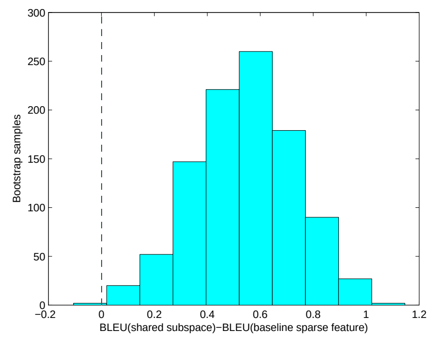

Figure 2: BLEU difference of 1000 bootstrap samples. 95% confidence interval is $[.15, .90]$ The proposed approach therefore seems to be a stable method.

5 Related Work in NLP

Previous reranking work in NLP can be classified into two different research focuses:

1. Engineering better features: In MT, (Och and others, 2004) investigates features extracted from a wide variety of syntactic representations, such as parse tree probability on the outputs. Although their results show that the proposed syntactic features gave little improvements, they point to some potential reasons, such as domain mismatch for the parser and overfitting by the reranking

[10]Note: In order to do this analysis, we needed to run Joint Regularization on the original feature representation, since the hashed representations are less interpretable. This turns out to be computationally prohibitive in the time being so we only ran on a smaller data set of 50 lists. Recently new optimization methods that are orders of magnitude faster have been developed (Liu et al., 2009), which makes larger-scale experiments possible.

Feature Representation	#Feature	Train BLEU	Test BLEU	Test PER
(baselines)				
First pass	20	29.5	28.5	38.3
All sparse features (Main baseline)	2.4M	36.9	28.6	38.2
All sparse features w/ ℓ_1 regularization	1200	36.5	28.5	38.6
Random hash representation	4000	33.0	28.5	38.2
(multitask learning)				
Unsupervised FeatureSelect	500	32.0	**28.8**	**37.7**
Joint Regularization	250	31.8	**28.9**	**37.5**
Shared Subspace	1000	32.9	**29.1**	**37.3**
(combination w/ high-frequency features)				
(a) Feature threshold $x > 100$	3k	31.7	27.9	38.2
(b) Feature threshold $x > 10$	60k	35.8	29.0	37.9
Unsupervised FeatureSelect + (b)	60.5k	36.2	**29.3**	**37.6**
Joint Regularization + (b)	60.25k	36.1	**29.4**	**37.5**
Shared Subspace + (b)	61k	36.2	**29.6**	**37.3**
Oracle (best possible)	–	36.9	36.9	33.1

Table 2: Results for different feature sets, with corresponding feature size and train/test BLEU/PER. All multitask features give statistically significant improvements over the baselines (boldfaced), e.g. Shared Subspace: 29.1 BLEU vs Baseline: 28.6 BLEU. Combinations of multitask features with high frequency features also give significant improvements over the high frequency features alone.

method. Recent work by (Chiang et al., 2009) describes new features for hierarchical phrase-based MT, while (Collins and Koo, 2005) describes features for parsing. Evaluation campaigns like WMT (Callison-Burch et al., 2009) and IWSLT (Paul, 2009) also contains a wealth of information for feature engineering in various MT tasks.

2. Designing better training algorithms: N-best reranking can be seen as a subproblem of structured prediction, so many general structured prediction algorithms (c.f. (Bakir et al., 2007)) can be applied. In fact, some structured prediction algorithms, such as the MIRA algorithm used in dependency parsing (McDonald et al., 2005) and MT (Watanabe et al., 2007) uses iterative sets of N-best lists in its training process. Other training algorithms include perceptron-style algorithms (Liang et al., 2006), MaxEnt (Charniak and Johnson, 2005), and boosting variants (Kudo et al., 2005).

The division into two research focuses is convenient, but may be suboptimal if the training algorithm and features do not match well together. Our work can be seen as re-connecting the two focuses, where the training algorithm is explicitly used to help discover better features.

Multitask learning is currently an active subfield within machine learning. There has already been some applications in NLP: For example, (Collobert and Weston, 2008) uses a deep neural network architecture for multitask learning on part-of-speech tagging, chunking, semantic role labeling, etc. They showed that jointly learning these related tasks lead to overall improvements. (Deselaers et al., 2009) applies similar methods for machine transliteration. In information extraction, learning different relation types can be naturally cast as a multitask problem (Jiang, 2009; Carlson et al., 2009). Our work can be seen as following the same philosophy, but applied to N-best lists.

In other areas, (Reichart et al., 2008) introduced an active learning strategy for annotating multitask linguistic data. (Blitzer et al., 2006) applies the multitask algorithm of (Ando and Zhang, 2005) to domain adaptation problems in NLP. We expect that more novel applications of multitask learning will appear in NLP as the techniques become scalable and standard.

6 Discussion and Conclusion

N-best reranking is a beneficial framework for experimenting with large feature sets, but unfortunately feature sparsity leads to overfitting. We addressed this by re-casting N-best lists as multitask learning data. Our MT experiments show consistent statistically significant improvements.

From the Bayesian view, multitask formulation of N-best lists is actually very natural: Each N-best is generated by a different data-generating distribution since the input sentences are different, i.e. $p(e|f^1) \neq p(e|f^2)$. Yet these N-bests are related since the general $p(e|f)$ distribution depends on the same first-pass models.

The multitask learning perspective opens up interesting new possibilities for future work, e.g.:

- Different ways to partition data into tasks, e.g. clustering lists by document structure, or hierarchical clustering of data
- Multitask learning on lattices or N-best lists with larger N. It is possible that a larger hypothesis space may improve the estimation of task-specific weights.
- Comparing multitask learning to sparse online learning of batch data, e.g. (Tsuruoka et al., 2009).
- Modifying the multitask objective to incorporate application-specific loss/decoding, such as Minimum Bayes Risk (Kumar and Byrne, 2004)
- Using multitask learning to aid large-scale feature engineering and visualization.

Acknowledgments

We have received numerous helpful comments throughout the course of this work. In particular, we would like to thank Albert Au Yeung, Jun Suzuki, Shinji Watanabe, and the three anonymous reviewers for their valuable suggestions.

References

Jacob Abernethy, Peter Bartlett, and Alexander Rakhlin. 2007. Multitask learning with expert advice. In *COLT*.

Rie Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR*.

Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. 2008. Convex multitask feature learning. *Machine Learning*, 73(3).

G. Bakir, T. Hofmann, B. Scholkopf, A. Smola, B. Taskar, and S. V. N. Vishwanathan, editors. 2007. *Predicting structured data*. MIT Press.

J. Blitzer, R. McDonald, and F. Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. Findings of the 2009 workshop on statistical machine translation. In *WMT*.

Andrew Carlson, Justin Betteridge, Estevam Hruschka, and Tom Mitchell. 2009. Coupling semi-supervised learning of categories and relations. In *NAACL Workshop on Semi-supervised learning for NLP (SSLNLP)*.

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

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL*.

David Chiang, Wei Wang, and Kevin Knight. 2009. 11,001 new features for statistical machine translation. In *NAACL*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2).

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural langauge parsing. *Computational Linguistics*, 31(1).

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML*.

Hal Daume. 2009. Bayesian multitask learning with latent hierarchies. In *UAI*.

Thomas Deselaers, Sasa Hasan, Oliver Bender, and Hermann Ney. 2009. A deep learning approach to machine transliteration. In *WMT*.

Jenny Rose Finkel and Chris Manning. 2009. Hierarchical Bayesian domain adaptation. In *NAACL-HLT*.

Kuzman Ganchev and Mark Dredze. 2008. Small statistical models by random feature mixing. In *ACL-2008 Workshop on Mobile Language Processing*.

Jing Jiang. 2009. Multitask transfer learning for weakly-supervised relation extraction. In *ACL*.

Taku Kudo, Jun Suzuki, and Hideki Isozaki. 2005. Boosting-based parse reranking with subtree features. In *ACL*.

Shankar Kumar and William Byrne. 2004. Minimum bayes-risk decoding for statistical machine translation. In *HLT-NAACL*.

P. Liang, A. Bouchard-Cote, D. Klein, and B. Taskar. 2006. An end-to-end discriminative approach to machine translation. In *ACL*.

J. Liu, S. Ji, and J. Ye. 2009. Multi-task feature learning via efficient l2,1-norm minimization. In *UAI*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large margin training of dependency parsers. In *ACL*.

Guillaume Obozinski, Ben Taskar, and Michael Jordan. 2009. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*.

F.J. Och et al. 2004. A smorgasbord of features for statistical machine translation. In *HLT/NAACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *ACL*.

Michael Paul. 2009. Overview of the iwslt 2009 evaluation campaign. In *IWSLT*.

Ariadna Quattoni, Xavier Carreras, Michael Collins, and Trevor Darrell. 2009. An efficient projection for L1-Linfinity regularization. In *ICML*.

Roi Reichart, Katrin Tomanek, Udo Hahn, and Ari Rappoport. 2008. Multi-task active learning for linguistic annotations. In *ACL*.

Brian Roark, Murat Saraclar, and Michael Collins. 2007. Discriminative n-gram language modeling. *Computer Speech and Language*, 21(2).

Libin Shen, Anoop Sarkar, and Franz Och. 2004. Discriminative reranking for machine translation. In *HLT-NAACL*.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. 2009. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *ACL-IJCNLP*.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *EMNLP-CoNLL*.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *ICML*.

Kai Yu and Volker Tresp. 2005. Learning to learn and collaborative filtering. In *NIPS-2005 Workshop on Inductive Transfer*.

Ying Zhang, Stephan Vogel, and Alex Waibel. 2004. Interpreting BLEU/NIST scores: How much improvement do we need to have a better system? In *LREC*.