Multi-Layer Ensembling Techniques for Multilingual Intent Classification

Charles Costello, Ruixi Lin, Vishwas Mruthyunjaya, Bettina Bolla, Charles Jankowski

CloudMinds Technology Inc Santa Clara, CA, USA

{charlie.costello, ruixi.lin, vish.m, bettina.bolla, charles.jankowski}@cloudminds.com

Abstract

In this paper we determine how multi-layer ensembling improves performance on multilingual intent classification. We develop a novel multi-layer ensembling approach that ensembles both different model initializations and different model architectures. We also introduce a new banking domain dataset and compare results against the standard ATIS dataset and the Chinese SMP2017 dataset to determine ensembling performance in multilingual and multi-domain contexts. We run ensemble experiments across all three datasets, and conclude that ensembling provides significant performance increases, and that multi-layer ensembling is a no-risk way to improve performance on intent classification. We also find that a diverse ensemble of simple models can reach perform comparable to much more sophisticated state-of-the-art models. Our best F_1 scores on ATIS, Banking, and SMP are 97.54%, 91.79%, and 93.55% respectively, which compare well with the state-of-the-art on ATIS and best submission to the SMP2017 competition. The total ensembling performance increases we achieve are 0.23%, 1.96%, and 4.04% F_1 respectively.

1 Introduction

In this work we determine how beneficial multi-layer ensembling is for intent classification in multilingual and multi-domain contexts. Our hypothesis is that multi-layer ensembles can provide significant performance increases compared to their constituent models. Additionally, we test whether ensembles of simple models can compete with more complex models, and how well ensembled models generalize across different languages and domains.

Our contributions in this work are threefold:

- We present an exhaustive performance comparison of different CNN and RNN multi-layer ensembles across three different datasets on the intent classification task. We run experiments on 12 individual models and 4083 separate ensembles.
- We develop a novel multi-layer ensembling approach. As opposed to focusing on either ensembling
 different random initializations or different model architectures, as most ensembling approaches
 do, we combine both methods in order to fully explore the possible performance improvements of
 ensembling.
- We make a novel comparison of ensemble results across multiple domains and languages. This
 multilingual and mulit-domain analysis allows us to discuss the generalizability of ensembling techniques, as well as their potential performance gains in different contexts.

Our paper is outlined in the following manner. In section 2 we discuss other work related to our own. In section 3 we describe and compare the three datasets. In section 4 we discuss the technical background for our work, including the intent classification task, the individual CNN and RNN models, and the multi-layer ensembling approach. In section 5 we present a detailed description of our experiments and an analysis of the results. Finally, in section 6 we draw general conclusions from our work.

Place licence statement here for the camera-ready version.

2 Related Work

As an essential component of NLP systems, intent classification has been thoroughly researched. Although approaches such as SVMs (Sheikh et al., 2016), neural bag-of-words (Purohit et al., 2015), and knowledge guided patterns (Bhargava et al., 2013) have proven successful, deep learning has been shown to be a more effective algorithm to solve classification tasks in natural language processing (Mikolov et al., 2010; Shi et al., 2016).

The two major deep learning variants used for intent classification are Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). Although CNNs are heavily used in computer vision tasks, they have also been adapted and proven successful on NLP tasks (Collobert and Weston, 2008; Shi et al., 2016). RNN approaches have also worked very well for intent classification (Ravuri and Stolcke, 2015; Socher et al., 2012). In popular cases, intent classification resides alongside slot filling as a jointly modeled NLP task using deep learning algorithms (Bhardwaj and Rudnicky, 2017; Hashemi et al., 2016). For instance, the CNN based triangular CRF architecture exploits the dependency between intent and slots (Xu and Sarikaya, 2013), whilst the attention-based RNN model includes attention vectors similar to those used in attention-based encoder-decoder models (Liu and Lane, 2016). Another such joint classification model is an RNN architecture that uses GRU representations to identify intents and slots (Zhang and Wang, 2016).

Although CNNs and RNNs have shown impressive performance on many NLP tasks, individual models are still susceptible to overfitting and local optima. To combat this, many researchers have explored ensembling multiple models to obtain higher quality inference (Rokach, 2010). These methods have been used widely in NLP research (Jin et al., 2017), and Purohit et al. (2015) used single-layer ensembling in their work on intent classification.

3 Data

3.1 ATIS

The ATIS (Airline Travel Information System) dataset was collected by DARPA in the early 90's (Price, 1990; Tur et al., 2010). It is one of the most frequently used dataset for SLU (spoken language understanding) research. It was based on spoken queries regarding flight related information. An example utterance is: "I'd like to fly from Boston to San Francisco." The dataset exists in both spoken and text form; we use the text form here.

The utterances are represented by semantic frames, where each sentence has a single or multiple intents and slots filled with phrases. Labels are encoded with IOB (Inside Outside Beginning) representation (Sang et al., 1999). There are 17 single intents and 8 multiple intent combinations. The distribution is biased as the most frequent intent, Flight, represents about 70% of the traffic. The ATIS train set contains 4,978 sentences. The test set contains 893 sentences.

3.2 Banking

For a particular banking-related application, we wanted to get domain specific data. Since most banking data is proprietary, we had to collect our own. Our proprietary banking dataset was collected as an initial in-house pilot study of 360 usable written utterances and increased to a total of 5,358 usable (clean) utterances by utilizing Mechanical Turk. An example utterance is: "I would like to open a checking account." Data collection was based on 12 prompts representing different slot/intent combinations.

The utterances were tokenized and annotated following the IOB representation used in the ATIS dataset, with single or multiple intents. There are 9 single intents, including the Null Intent, and 10 dual intent combinations. The distribution is somewhat skewed towards the Find_Out intent, but is much more evenly distributed than ATIS. Our banking train set contains 4,286 sentences. The test set contains 1,072 sentences.

3.3 SMP

The SMP2017ECDT (SMP) dataset is a Chinese domain classification dataset consisting of textual queries recorded from human-computer dialogues (Zhang et al., 2017). We divide the datasets into a

train and a test set of 3,069 and 667 samples respectively. The first step for many QA systems is to categorize a user intent into a specific domain, and we consider these domains as the first level user intents, for instance, "结婚了吗" (Are you married) is labeled as Chit-chat, which implies that the user intent is related to chit-chatting. The domains cover 31 categories, including Chit-chat and 30 task-oriented categories. The SMP dataset is skewed towards Chit-chat with around 20% of data in it, and the rest of the 30 categories are more evenly distributed. Since the Chinese data given is not tokenized, we used the Jieba tokenizer¹ to tokenize the sentences.

4 Technical Background

4.1 Task

Intent classification is the task of correctly labeling a natural language utterance from a predetermined set of intents. We can treat it as a multiclass classification task, and train a discriminative machine learning model to output a predicted classification for a given utterance. In cases where an utterance has multiple intents, we concatenate the intents and treat the result as a single, distinct intent. This allows us to treat the task as single-label, even when multiple intents are used. We use CNN and RNN models to perform this statistical inference. These models are trained on labeled utterances using backpropagation and gradient descent. At inference time, our models compute the probability distribution over all intent classes for a given utterance conditioned on the trained weights. This probability for each intent label is calculated by taking the softmax of the model outputs, and the predicted class is then simply chosen by taking the argmax of the distribution.

4.2 CNNs

Convolutional Neural Networks (CNNs) are widely used for sentence classification tasks (LeCun and Bengio, 1995). CNNs have proven to be effective for NLP tasks like sentence classification (Kim, 2014). Besides a baseline of a simple CNN with four interleaved layers of convolution and pooling, we developed three CNN variants, including a character-level model (dos Santos and Zadrozny, 2014), CharCNN, an attention based (Neumann and Vu, 2017) bidirectional model (Vu, 2016), ABiCNN, and a combination of both variants which is a character-level attentive bidirectional model, ACharBiCNN. We pad samples of variable lengths to the maximal sample length of all samples in a dataset in order to do mini-batch training. There are 50 and 100 convolution filters for the first and second convolution respectively.

The character approach (dos Santos and Zadrozny, 2014) learns a word representation from the characters that make up a word in a character embedding layer. The output of the character embedding layer will be a mini batch of sentences embedded with the combined vector, which is followed by the normal CNN convolution and pooling layers.

We also apply a bidirectional CNN structure (Vu, 2016) and an attention mechanism (Neumann and Vu, 2017). Bidirectional NNs have shown better performances than unidirectional NNs in language modeling and machine translation (Vu, 2016; Bahdanau et al., 2015). Attention based neural models have been seen successfully used in pattern recognition (Olshausen et al., 1993), NLP (Yin et al., 2016), and computer vision tasks (Sun and Fisher, 2003). Our attentive bidirectional CNN model is built as follows. A bidirectional CNN takes inputs from a positive and a negative time direction, so that the output gets information from both past and future states. A combined hidden layer output can thus be obtained by a weighted sum of the individual hidden layer outputs, activated by a non-linear activation function. Then attention mechanism is applied, which helps attend to important parts of a feature map by learning weights of each part. The attention based bidirectional hidden layer output is then input to a fully-connected layer, followed by a softmax layer. Combining the character embedding approach and the attention mechanism, we obtained the character-level attentive bidirectional CNN model.

¹https://github.com/fxsjy/jieba

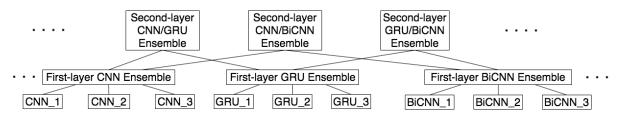


Figure 1: Multi-layer Ensemble Approach

4.3 RNNs

RNNs are deep neural networks designed to process sequential data. They have proven to be very effective for many natural language processing tasks, including language modeling (Mikolov et al., 2010) and text generation (Sutskever et al., 2011). RNNs are distinguished from traditional deep neural networks by the use of a cell architecture that is sequentially unrolled for a specified number of timesteps. This allows the network to communicate through multiple timesteps, effectively remembering past information. However, it was noted by Bengio et al. (1994) that RNNs suffer from the vanishing gradient problem when the number of timesteps exceed a small number, prohibiting the network from remembering long-term information. To correct this problem, many RNN architectures were suggested that would allow for longer-term memory. Two of the most popular, which we use in this work, are the Long Short-Term Memory (LSTM) cell (Hochreiter and Schmidhuber, 1997), and the Gated Recurrent Unit (GRU) cell (Cho et al., 2014).

When developing RNN models for intent classification, we looked at many architectural considerations, including using a LSTM or GRU cell architecture, using character embeddings as described in Section 4.2 (dos Santos and Zadrozny, 2014), and using a bidirectional architecture (Schuster et al., 1997). We then created a model for each combination, resulting in 8 total models. Each model has 512 hidden units and a word embeddings size of 300. Models with character embeddings have a character embedding size of 300. Word and character embeddings were randomly initialized and trained along with the model's other weights, without the use of any pretrained vectors. We trained for 100 epochs with a batch size of 32 and a dropout rate of 0.5.

4.4 Multi-layer Ensembling

We ran experiments on numerous multi-layer ensembles of our CNN and RNN models, as well as the individual models themselves, on all three datasets. We evaluated the ensembles and individual models on each test set, and used the unweighted F_1 score as our metric. When ensembling models, we use a majority vote with confidence approach (Jin et al., 2017). For each utterance, we first see if a majority of the models agree upon the prediction. If so, we choose that prediction. If not, we compare the confidence of each model's prediction as measured by the softmax score, and take the highest score. We chose majority vote with confidence as our ensembling method because it is a very simple and straightforward technique, and the goal of our work is to show how even simple ensembling can improve performance greatly.

Figure 1 shows our multi-layer ensemble. We can see that is comprised of distinct first-layer and second-layer ensembles that are arranged in a hierarchical structure. The first layer ensembles different random initializations of each model architecture (e.g. CNN_1, CNN_2, and CNN_3). The second layer then ensembles the ensembled version of each model architectures (e.g. CNN ensemble and GRU ensemble). This allows the final ensembles to learn from both different random initializations and different model architectures.

More concretely, to obtain our first-layer ensembles, we first train each model three times with the same hyperparameters but different random initializations for the weights. We then use majority vote with confidence to find the ensemble predictions, and finally calculate the F_1 score. In Tables 1 and 3, we show the three individual model F_1 scores (columns "1", "2", "3"), as well as their ensemble F_1 (column "En"). To obtain the second-layer ensembles, we generate all possible combinations of the first-layer ensembles that are of size two or more. As we have 12 first-layer ensembles, this gives us second-

Model	1	2	3	En	Mode	1	1	2	3	En
CNN	95.41	95.52	95.63	95.97	CNN		87.22	87.59	88.90	89.27
CharCNN	91.94	92.50	93.39	94.40	CharC	CNN	87.59	87.69	88.53	89.18
ABiCNN	96.64	96.75	97.09	97.20	ABiC	NN	88.99	89.37	89.83	90.11
ACharBiCNN	93.73	93.84	94.29	95.07	ACha	rBiCNN	88.25	88.71	89.09	90.11
(a) ATIS				(b) Banking						
		Mode	1	1	2	3	En			
		CNN		85.46	85.61	86.06	87.26			
		CharC	CNN	83.36	83.81	84.11	86.51			
ABiCNN		86.36	86.51	86.96	88.16					
		ACha	rBiCNN	84.41	85.01	85.61	89.06			

Table 1: CNN First-layer Ensembles F_1 (%)

(c) SMP

Model	ATIS	Banking	SMP	Avg Inc
CNN/CharCNN	95.97	89.46	87.71	1.14
CNN/ABiCNN	96.70	90.49	89.81	2.10
CNN/ACharBiCNN	95.97	90.11	89.51	1.48
CharCNN/ABiCNN	96.19	90.58	88.76	1.70

(a) Interesting Two-model Ensembles

Dataset	$ F_1 $	Ensemble
ATIS	97.20	ABiCNN
		CNN,
Banking	90.86	ABiCNN,
		ACharBiCNN
		CNN,
SMP	89.96	CharCNN,
		ABiCNN

(b) Best Ensembles

Table 2: CNN Second-layer Ensembles F_1 (%)

layer ensembles with $\{2,3\dots12\}$ number of first-layer ensembles. This yields $2^{12}-12-1=4083$ distinct second-layer ensembles. We again use majority vote with confidence to evaluate the second-layer ensembles. Tables 2 and 4 compare F_1 scores of different second-layer ensembles.

5 Experiments and Results

5.1 CNN Ensembles

First-layer Ensembles: The results of experiments on ensembling single CNN models are listed for ATIS, Banking, and SMP in Table 1. The first thing that draws our attention is that first-layer ensembling improves the performances on all individual models across all three datasets. Despite slight performance differences at each of the three times of training, the F_1 scores of ensembles are higher than all of individual models. Comparing the ensemble F_1 with the average F_1 of all three individual models, there is a lowest 0.37% improvement (ABiCNN on ATIS with individual scores: 96.64%, 96.75%, 97.09%, ensemble score: 97.20%) and a highest 4.05% improvement (ACharBiCNN on SMP with individual scores: 84.41%, 85.01%, 85.61%, ensemble score: 89.06%). This shows that first-layer ensembles help to exploit the benefits of different random initializations, mitigating the risk of converging to local optima.

Second-layer Ensembles: In addition, with the first-layer ensemble results as input, the second-layer ensemble shows some interesting results when it comes to combining models of different structures. Out of the 11 second-layer ensembles of different combinations of the CNN models, we focused on comparing the ensemble of character embedding based single-model ensembles with the ensemble of simple CNNs, the ensemble of attention and bidirectional based single-model ensembles with the ensemble of simple CNNs, and the ensemble of character embedding based single-model ensembles with the ensemble of attention and bidirectional based ones. We did an exhaustive comparison across all 2-ensembles,

Model	1	2	3	En	Model	1	2	3	En
GRU	96.98	97.20	97.31	97.31	GRU	87.69	87.78	88.53	88.15
CharGRU	95.07	95.30	95.41	95.63	CharGRU	88.15	88.62	89.55	90.02
BiGRU	96.30	96.53	97.20	96.86	BiGRU	86.94	87.69	88.15	88.15
CharBiGRU	94.18	94.29	94.40	94.62	CharBiGRU	86.66	87.41	89.18	88.99
LSTM	96.19	96.53	96.86	96.98	LSTM	87.97	88.06	88.43	88.99
CharLSTM	95.07	95.97	96.86	96.53	CharLSTM	88.90	89.18	89.74	90.67
BiLSTM	96.75	96.75	96.86	97.09	BiLSTM	87.69	88.99	89.18	89.65
CharBiLSTM	93.95	94.51	94.74	94.74	CharBiLSTM	88.53	88.90	89.09	89.93
(a) ATIS				(b) Ro	nkina	I	1		

ATIS				(b) Bankin		
Model	1	2	3	En		
GRU	83.06	83.96	85.61	84.71		
CharGRU	86.81	88.31	89.36	89.96		
BiGRU	84.11	85.31	86.21	86.66		
CharBiGRU	85.16	87.11	87.41	88.46		
LSTM	84.41	85.76	85.91	87.26		
CharLSTM	88.01	88.31	89.51	91.30		
BiLSTM	83.81	84.86	86.06	86.36		
CharBiLSTM	87.56	88.16	88.76	89.51		
(c) SMP						

Table 3: RNN First-layer Ensembles F_1 (%)

3-ensembles, and the all-ensemble, and found that the second-layer ensembles outperformed their component first-layer ensembles. We presented the most interesting 2-ensemble results in Table 2a. The average increase is calculated by averaging the increase for all three datasets, where the increase for each dataset is again given by the average of percentage increase of the ensemble (e.g. CNN/CharCNN) compared to its component single-model ensembles (e.g. CNN and CharCNN).

We analyzed the idiosyncrasies of the best models across datasets. The best F_1 on ATIS, Banking, and SMP (Table 2b) are 97.20% with a minimal first-layer ABiCNN ensemble, 90.86% with a minimal second-layer CNN/ABiCNN/ACharBiCNN ensemble, and 89.96% with a minimal second-layer CNN/CharCNN/ABiCNN ensemble. It seems that the specific ensemble constituents differ from dataset to dataset, but there is a dominant first-layer ensemble component ABiCNN appearing in all best ensembles across all three datasets, and it results from the fact that the attention structure and bidirectionality generally contribute additional contextual information to learning, regardless of the language the data is in. From another aspect, despite of idiosyncratic components in the best models, a combination of different model structures, like combining simple structures with character embeddings, attention, and bidirectionality, will further help reduce False Positive rates as they compensate for one another on a probabilistic basis.

5.2 RNN Ensembles

We performed a series of experiments on RNN first-layer and second-layer ensembles on all three datasets. We obtained first-layer ensembles by collecting F_1 scores of each RNN model's three trained versions and their ensembles for ATIS, Banking, and SMP (Table 3). We then performed second-layer ensemble experiments on all of the RNN models, and recored the best ensemble for each dataset in Table 4a.

First-layer Ensembles: We can derive many interesting points from these results. Firstly, each model benefits greatly from ensembling. More specifically, every first-layer ensemble performs better than at least two of its individual versions, and the vast majority outperform all three. The average F_1 gain for first-layer RNN ensembles is 0.66% (ATIS), 1.50% (Banking), and 2.66% (SMP).

We can also see how model complexity affects performance across different datasets. For exam-

Dataset	F_1	Ensemble		
		GRU, LSTM,		
ATIS	97.42	BiGRU, BiLSTM,		
		CharGRU, CharBiLSTM		
Banking	91.32	CharGRU, CharLSTM		
SMP	91.60	BiGRU, CharGRU,		
SMP	91.00	CharLSTM, CharBiGRU		
(a) Best RNN Ensembles				

F_1	Ensemble
	GRU, BiGRU,
97.54	BiLSTM, CNN
	CharCNN, ABiCNN
	ACharBiCNN
91.79	CharGRU, CharLSTM,
	CharBiGRU, CharBiLSTM
	CNN, CharCNN
	ABiCNN, ACharBiCNN
	CharLSTM, CharBiGRU,
93.55	CharBiLSTM, CharCNN,
	ABiCNN, ACharBiCNN
	97.54

(b) Best CNN/RNN Ensembles

Table 4: Second-layer Ensembles F_1 (%)

ple on ATIS, the simple GRU model is the highest scoring first-layer ensemble with 97.31%. Adding complexity, either through bidirectionality or character embedding generally lowers performance, with CharBiGRU and CharBiLSTM getting the lowest scores of 94.62% and 94.74% respectively. This is in contrast to Banking, where character embedding always helps performance (e.g. 88.15% (GRU) to 90.02% (CharGRU)), and bidirectionality giving mixed results. SMP performance also contrasts with ATIS, as GRU performs worst, and character embedding helps greatly (e.g. 87.26% (LSTM) to 91.30% (CharLSTM)).

It is particularly interesting that Banking and SMP follow similar performance trends, despite the stark lexical and syntactic differences between English and Chinese, as well as ATIS and Banking having such different performance characteristics despite being the same language and similar size. This points to dataset/model coherence being based on subtle data characteristics. It is not obvious by inspecting ATIS and Banking data why character embedding is beneficial for Banking but detrimental for ATIS, though Banking's greater semantic diversity may be a contributing factor.

A related consideration is the relationship between bidirectionality and character embedding. In all three datasets, CharBiGRU and CharBiLSTM show significantly lower performance than their corresponding unidirectional versions, CharGRU and CharLSTM. This indicates that character embeddings and bidirectionality are somehow incompatible in an RNN architecture. This may suggest that the temporal information that is gained by reading input in both directions is lost or confused by embedding input as both words and characters.

Another interesting result is how performance ranges between single model ensembles vary between different datasets. While Banking and ATIS only have a 2.52% range (88.15% GRU and BiGRU to 90.67% CharLSTM) and 3.12% range (94.62% CharBiGRU to 97.74% GRU) respectively, SMP has a 6.59% range (84.71% GRU to 91.30% CharLSTM), more than twice either of the others. However, that range mostly captures the difference between character embedded models and non-character embedded models. Consequently, while character embedding may help for English datasets like Banking, it is critical for Chinese datasets. This makes intuitive sense, as Chinese characters contain rich semantic information.

Second-layer Ensembles: Finally, we can draw some results from the RNN second-layer ensemble experiments (Table 4a). We can see that each best second-layer ensemble outperforms all of its single-model ensembles, showing that second-layer ensembles further increase the performance of first-layer ensembles. Each best second-layer ensemble also contains at least one GRU and at least one LSTM, again pointing to the importance of diversity in ensembling.

Model	ATIS	Banking	SMP
Without Char	97.54	91.04	91.45
With Char	97.54	91.79	93.55
Gain	0.00	0.75	2.10

Model	ATIS	Banking	SMP
First-layer Gain	0.54	1.03	1.91
Second-layer Gain	0.94	1.24	2.91
Total Gain	0.23	1.96	4.04

(a) Character embeddings gains

(b) Ensembling layer gains

Table 5: F_1 (%) gains from character embedding and ensembling layers

5.3 CNN and RNN Ensembles

Second-layer Ensembles: Table 4b shows the best second-layer ensembles resulting from experiments with all 12 first-layer ensembles. On ATIS, multiple second-layer ensembles obtain the highest F_1 of 97.54%. We show one of these in table 4b, though other ensembles also obtain the same performance, including {GRU, BiLSTM, ABiCNN}. The first thing to note here is that these ensembles generally contain six or more first-layer ensembles, indicating that there is some critical mass of models needed to achieve maximum performance. Also, these ensembles have an even or very close to even split between RNN and CNN models, as well as at least one character embedded and one bidirectional model. This extends the theme discussed earlier, that greater performance is achieved through a diverse ensemble. We can see especially strong proof of this in SMP's best second-layer ensemble performance of 93.55%. This is a nearly two percent increase from the best RNN second-layer ensemble of 91.60%, and a more than three percent increase from the best CNN second-layer ensemble of 89.96%. This large performance increase over two robust second-layer ensembles clearly shows that diversity of model architecture is critical for ensemble success.

Table 5a shows the performance difference between the best second-layer ensembles without and with character embedding. Again we can see that character embedding is critical for the Chinese dataset, and much less so for the English datasets. Additionally, we can see that on ATIS, second-layer ensembles both with and without character embedding attain the best performance. This points to second-layer ensembling being effectively only beneficial; diverse second-layer ensembles can raise performance without any real risk of degrading performance by adding specific models, even if those models do poorly by themselves.

We can also compare performance between datasets. ATIS performs 5.75% better than Banking, and 3.99% better than SMP, while SMP performs 1.76% better than Banking. Indeed, throughout both first-layer and second-layer ensembles, the same models and ensembles generally perform better on ATIS than Banking and SMP. This indicates that some datasets are generally easier for models to learn than others. The difficulty of SMP can at least in part be explained by having a much more diverse set of intents than either Banking or ATIS. Also, both Banking and SMP have a greater lexical and stylistic range than ATIS, certainly making training more difficulty.

Finally, Table 5b shows the average first-layer and average second-layer performance increases, as well as the total F_1 ensemble gain for each dataset. The average ensemble performance increases were calculated by averaging each ensemble's average difference between its performance and the performance of each of its constituent models, and the total gain was calculated as the difference between the best individual model and the best second-layer ensemble. We can see here that ensembling is much more important for some models and datasets than others, though clearly both first-layer and second-layer ensembling is more important for SMP than Banking or ATIS. The total gains represent the potential total performance increase that someone may gain through utilizing multi-layer ensembling. We see that ensembling helps less for the easier ATIS dataset, but that total increases of around 2% and 4% on Banking and SMP are significant. Furthermore, these total gains start from the best possible individual models; total gains from other individual models can be significantly greater. Together, these results show that ensembling can be a powerful technique for creating better intent classification systems.

Model	F_1 (%)	Development Time
Recursive NN (Guo et al., 2014)	95.40	High
Bidirectional GRU with Context Window (Zhang and Wang, 2016)	97.53	High
Attention Encoder-Decoder NN (Liu and Lane, 2016)	97.98	High
Our Best Ensemble	97.54	Low

(a) ATIS

Model	F_1 (%)	Development Time
Self-Inhibiting Residual CNN (Lu, 2017)	92.88	High
LSTM with database keyword extraction (Tang et al., 2017)	93.91	High
Our Best Ensemble	93.55	Low

(b) SMP

Table 6: Comparison to Previous Approaches

5.4 Comparison to Previous Approaches

Tables 6a and 6b show how our best ensembles perform compared to other approaches. We can see that our best ensembles perform quite well compared to other approaches on both ATIS and SMP. Many of these models are considerably more sophisticated than our own. For example, Liu and Lane (2016) use an encoder-decoder architecture, which combines two separate neural networks, and Tang et al. (2017) use a separate data-based keyword extraction algorithm along with a deep learning model. Our models, on the other hand, are all standard models that can be developed quickly without needing to fine-tune the architecture. The result that our ensembles of simpler models can perform in the same range as these more sophisticated models shows that ensembling is a viable strategy for achieving very high performance on a dataset without intensive model engineering. This simplicity is also advantageous from an industry viewpoint. Since our models don't involve time-consuming development, they can more easily be deployed to a production environment.

6 Conclusion

Running intent classification experiments on many variants of CNN and RNN models, as well as numerous first-layer and second-layer ensembles of those models, presented two main conclusions. First, both first-layer and second-layer ensembling can boost performance greatly, and greater diversity ensembles almost always leads to better results. Indeed, a large, diverse ensemble of straightforward models can be competitive with much more sophisticated state-of-the-art models. Consequently, researchers and engineers hoping to improve performance on intent classification should not hesitate to use multi-layer ensembling in their work.

The second main conclusion of our work concerns the idiosyncratic nature of datasets. Different models and ensembles perform quite differently on even fairly similar datasets. Although some architectures do show general performance trends across tasks and datasets, such as RNNs generally outperforming CNNs on natural language processing tasks, different datasets are idiosyncratic enough that it is extremely difficult to know *a priori* which specific model will work best for a given dataset. Network architecture, cell type, character embedding, bidirectionality, and attention can all affect datasets very differently. This idiosyncrasy strengthens the case for using ensemble methods, as they can help to distribute the insights of specific models across different datasets. We also found that diverse ensembles provide a non-negative performance gain, so that multi-layer ensembling won't hurt performance and often greatly helps it. In this way, multi-layer ensembling is a no-risk and low-effort way to improve model performance on intent classification.

These conclusions support our hypothesis that multi-layer ensembling can provide significant performance increases compared to their constituent models. In this work we also saw that multi-layer ensembling of simple models can compete with more complex models, and that ensembling provides a no-risk method to improve performance on intent classification across different languages and domains.

References

- [Bahdanau et al.2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.
- [Bengio et al.1994] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.
- [Bhardwaj and Rudnicky2017] Arjun Bhardwaj and Alexander Rudnicky. 2017. User intent classification using memory networks: A comparative analysis for a limited data scenario. *CoRR*.
- [Bhargava et al.2013] Aditya Bhargava, Asli Celikyilmaz, Dilek Hakkani-Tür, and Ruhi Sarikaya. 2013. Easy contextual intent prediction and slot detection. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pages 8337–8341.
- [Cho et al.2014] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *Association for Computational Linguistics*, pages 103–111.
- [Collobert and Weston2008] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. *Proceedings of the 25th international conference on Machine learning*, pages 160–167.
- [dos Santos and Zadrozny2014] Cicero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. *In Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, page 1818–1826.
- [Guo et al.2014] Daniel Guo, Gokhan Tur, Wen tau Yih, and Geoffrey Zweig. 2014. Joint semantic utterance classification and slot filling with recursive neural networks. 2014 IEEE Spoken Language Technology Workshop (SLT), pages 554–559.
- [Hashemi et al.2016] Homa Hashemi, Amir Asiaee, and Reiner Kraft. 2016. Query intent detection using convolutional neural networks. *International Conference on Web Search and Data Mining, Workshop on Query Understanding*.
- [Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- [Jin et al.2017] Lifeng Jin, Michael White, Evan Jaffe, Laura Zimmerman, and Douglas Danforth. 2017. Combining cnns and pattern matching for question interpretation in a virtual patient dialogue system. *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 11–21.
- [Kim2014] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *EMNLP*, pages 1746–1751.
- [LeCun and Bengio1995] Yann LeCun and Yoshua Bengio. 1995. Convolutional networks for images, speech, and time-series. *The handbook of brain theory and neural networks*, 3361(10).
- [Liu and Lane2016] Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *Interspeech 2016*, abs/1609.01454.
- [Lu2017] Chenyu Lu. 2017. Team report of deepbrain: A self-inhibiting residual convolution block based approach for text classification. *SMP-ECDT*.
- [Mikolov et al.2010] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. *Proceedings of the 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010*, 2:1045–1048.
- [Neumann and Vu2017] Michael Neumann and Ngoc Thang Vu. 2017. Attentive convolutional neural network based speech emotion recognition: A study on the impact of input features, signal length, and acted speech.
- [Olshausen et al.1993] Bruno Olshausen, Charles Anderson, and David Van Essenla. 1993. A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information. *The Journal of Neuroscience*, 13(11):4700–4719.
- [Price1990] Patti Price. 1990. Evaluation of spoken language systems: The atis domain. *Proceedings of the Workshop on Speech and Natural Language*, pages 91–95.

- [Purohit et al.2015] Hemant Purohit, Guozhu Dong, Valerie Shalin, Krishnaprasad Thirunarayan, and Amit Sheth. 2015. Intent classification of short-text on social media. *SocialCom 2015: 8th IEEE International Conference on Social Computing and Networking*, pages 222–228.
- [Ravuri and Stolcke2015] Suman Ravuri and Andreas Stolcke. 2015. Recurrent neural network and lstm models for lexical utterance classification. *ISCA International Speech Communication Association*, pages 135–139.
- [Rokach2010] Lior Rokach. 2010. Ensemble-based classifiers. Artificial Intelligence Review, 33:1–39.
- [Sang et al.1999] Erik Sang, Kim Tjong, and Jorn Veenstra. 1999. Representing text chunks. *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics*, pages 173–179.
- [Schuster et al.1997] Mike Schuster, Kuldip K. Paliwal, and A. General. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*.
- [Sheikh et al.2016] Imran Sheikh, Irina Illina, Dominique Fohr, and Georges Linares. 2016. Learning word importance with the neural bag-of-words model. *Proceedings of ACL 2016*.
- [Shi et al.2016] Yangyang Shi, Kaisheng Yao, Le Tian, and Daxin Jiang. 2016. Deep lstm based feature mapping for query classification. *HLT-NAACL*.
- [Socher et al.2012] Richard Socher, Brody Huval, Christopher Manning, and Andrew Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 1201–1211.
- [Sun and Fisher2003] Yaoru Sun and Robert Fisher. 2003. Object-based visual attention for computer vision. *Artificial Intelligence*, 146(2003):77–123.
- [Sutskever et al.2011] Ilya Sutskever, James Martens, and Geoffrey Hinton. 2011. Generating text with recurrent neural networks. *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.
- [Tang et al.2017] Jiecong Tang, Yongshi Liang, jiangyue Yan, Yanghui Li, Dawei Ling, Zhen Zeng, Zefeng Du, and Peijie Huang. 2017. Team report of scau sigsds on the closed task of intent classification for the evaluation of chinese human computer dialogue technology. *SMP-ECDT*.
- [Tur et al.2010] Gokhan Tur, Dilek Hakkani-Tür, and Larry Heck. 2010. What is left to be understood in atis? 2010 IEEE Spoken Language Technology Workshop, pages 19–24.
- [Vu2016] Ngoc Thang Vu. 2016. Sequential convolutional neural networks for slot filling in spoken language understanding. *CoRR*, arXiv:1606.07783v1.
- [Xu and Sarikaya2013] Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular crf for joint intent detection and slot filling. *Automatic Speech Recognition and Understanding (ASRU)*, 2013 IEEE Workshop on, pages 78–83.
- [Yin et al.2016] Wenpeng Yin, Hinrich Schutze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs.
- [Zhang and Wang2016] Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2993–2999.
- [Zhang et al.2017] Weinan Zhang, Zhigang Chen, Wanxiang Che, Guoping Hu, and Ting Liu. 2017. The first evaluation of chinese human-computer dialogue technology. *CoRR*, abs/1709.10217.