Local System Voting Feature for Machine Translation System Combination

Markus Freitag, Jan-Thorsten Peter, Stephan Peitz, Minwei Feng and Hermann Ney
Human Language Technology and Pattern Recognition Group
Computer Science Department
RWTH Aachen University
D-52056 Aachen, Germany
`<surname>@cs.rwth-aachen.de`

Abstract

In this paper, we enhance the traditional confusion network system combination approach with an additional model trained by a neural network. This work is motivated by the fact that the commonly used binary system voting models only assign each input system a global weight which is responsible for the global impact of each input system on all translations. This prevents individual systems with low system weights from having influence on the system combination output, although in some situations this could be helpful. Further, words which have only been seen by one or few systems rarely have a chance of being present in the combined output. We train a local system voting model by a neural network which is based on the words themselves and the combinatorial occurrences of the different system outputs. This gives system combination the option to prefer other systems at different word positions even for the same sentence.

1 Introduction

Adding more linguistic informed models (e.g. language model or translation model) additionally to the standard models into system combination seems to yield no or only small improvements. The reason is that all these models should have already been applied during the decoding process of the individual systems (which serve as input hypotheses for system combination) and hence already fired before system combination. To improve system combination with additional models, we need to define a model which can not be applied by an individual system.

In state-of-the-art confusion network system combination the following models are usually applied:

System voting (globalVote) models For each word the voting model for system i ($1 \leq i \leq I$) is 1 iff the word is from system i, otherwise 0.

Binary primary system model (primary) A model that marks the primary hypothesis.

Language model 3-gram language model (LM) trained on the input hypotheses.

Word penalty Counts the number of words.

To gain improvements with additional models, it is better to define models which are not used by an individual system. A simple model which can not be applied by any individual system is the binary system voting model (globalVote). This model is the most important one during system combination decoding as it determines the impact of each individual system. Each system i is assigned one globalVote model which fires if the word is generated by system i. Nevertheless, this simple model is independent of the actual words and the score is only based on the global preferences of the individual systems. This disadvantage prevents system combination from producing words which have only been seen by systems with low system weights (low globalVote model weights). To give systems and words with low weights a chance to affect the final output, we define a new local system voting model (localVote) which makes decisions based on the current word options and not only on a general weight. The local system voting model allows system combination to prefer different system outputs at different word positions even for the same sentence.

Motivated by the success of neural networks in language modelling (Bengio et al., 2006, Schwenk and Gauvain, 2002) and translation modelling (Son et al., 2012), we choose feedforward neural networks to train the novel model. Instead of calculating the probabilities in a discrete space, the neural network projects the words into a continuous space. This projection gives us the option to assign probability also to input sequences which were not observed in the training data. In system combination each training sentence has to be translated by all individual system engines which is time consuming. Due to this we have a small amount of training data and thus it is very likely that many input sequences of a test set have not be seen during training.

The remainder of this paper is structured as follows: in Section 2, we discuss some related work. In Section 3, the novel local system voting model is described. In Section 4, experimental results are presented which are analyzed in Section 5. The paper is concluded in Section 6.

2 Related Work

In confusion network decoding, pairwise alignments between all system outputs are generated. From the calculated alignment information, a confusion network is built from which the system combination output is determined using majority voting and additional models. The hypothesis alignment algorithm is a crucial part of building the confusion network and many alternatives have been proposed in the literature:

(Bangalore et al., 2001) use a multiple string alignment (MSA) algorithm to identify the unit of consensus and applied a posterior language model to extract the consensus translations. In contrast to the following approaches, MSA is unable to capture word reorderings.

(Matusov et al., 2006) produce pairwise word alignments with the statistical alignment algorithm toolkit GIZA++ that explicitly models word reordering. The context of a whole document of translations rather than a single sentence is taken into account to produce the alignments.

(Sim et al., 2007) construct a consensus network by using TER (Snover et al., 2006) alignments. Minimum bayes risk decoding is applied to obtain a primary hypothesis to which all other hypotheses are aligned.

(Rosti et al., 2007) extend the TER alignment approach and introduce an incremental TER alignment which aligns one system at a time to all previously aligned hypotheses.

(Karakos et al., 2008) use the inversion transduction grammar (ITG) formalism (Wu, 1997) and treat the alignment problem as a problem of bilingual parsing to generate the pairwise alignments.

(He et al., 2008) propose an indirect hidden markov model (IHMM) alignment approach to address the synonym matching and word ordering issues in hypothesis alignment.

(Heafield and Lavie, 2010) use the METEOR toolkit to calculate pairwise alignments between the hypotheses.

All confusion network system combination approaches only use the global system voting models. Regarding to this chapter, there has been similar effort in the area of speech recognition:

(Hillard et al., 2007) Similar work has been presented for system combination of speech recognitions systems: the authors train a classifier to learn which system should be selected for each output word. The learning target for each slot is the set of systems which match the reference word, or the null class if no systems match the reference word. Their novel approach outperforms the ROVER baseline by up to 14.5% relatively on an evaluation set.

3 Novel Local System Voting Model

In the following subsections we introduce a novel local system voting model (localVote) trained by a neural network. The purpose of this model is to prefer one particular path in the confusion network and therefore all local word decisions between two nodes leading to this particular path. More precisely, we want the neural network to learn an oracle path extracted from the confusion network graph which leads to the lowest error score. In Subsection 3.1, we describe a polynomial approximation algorithm to extract the best sentence level BLEU (sBLEU) path in a confusion network. Taking this path as reference path, we define the model in Subsection 3.2 followed by its integration in the linear model combination in Subsection 3.3.

3.1 Finding sBLEU-optimal Hypotheses

In this section, we describe a polynomial approximation algorithm to extract the best sBLEU hypothesis from a confusion network. (Leusch et al., 2008) showed that this problem is generally NP-hard for the popular BLEU (Papineni et al., 2002) metric. Nevertheless, we need some paths which serve as “reference paths“.

Using BLEU as metric to extract the best possible path is problematic as in the original BLEU definition there is no smoothing for the geometric mean. This has the disadvantage that the BLEU score becomes zero already if the four-gram precision is zero, which can happen obviously very often with short or difficult translations. To allow for sentence-wise evaluation, we use the sBLEU metric (Lin and Och, 2004), which is basically BLEU where all n-gram counts are initialized with 1 instead of 0. The brevity penalty is calculated only on the current hypothesis and reference sentence.

We use the advantage that confusion networks can be sorted topologically. We walk the confusion network from the start node to the end node, keeping track of all n-grams seen so far. At each node we keep a k-best list containing the partial hypotheses with the most n-gram matches leading to this node and recombine only partial hypotheses containing the same translation. As the search space can become exponentially large, we only keep k possible options at each node. This pruning can lead to search errors and hence yield non-optimal results. If needed for hypotheses with the same n-gram counts, we prefer hypotheses with a higher translation score based on the original models. For the final node we add the brevity penalty to all possible translations.

As we are only interested in arc decisions which match a reference word, we simplify the confusion network before applying the algorithm. If all arcs between two adjacent nodes are not present in the reference, we remove all of them and add a single arc labeled with "UNK". This reduces the vocabulary size and still gives us the same best sBLEU scores as before. In Figure 1, a confusion network of four input hypotheses is given. As the words *black*, *red*, *orange*, and *green* are all not present in the reference, all of them are mapped to one single "UNK" arc (cf. Figure 2). The best sBLEU path is *the UNK car*.

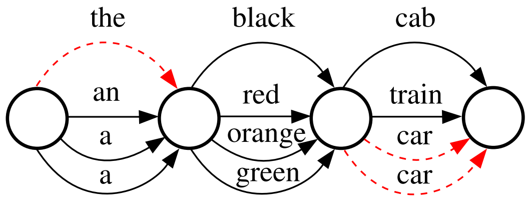

Figure 1: System A: *the black cab* ; System B: *an red train* ; System C: *a orange car* ; System D: *a green car* ; Reference: *the blue car* .

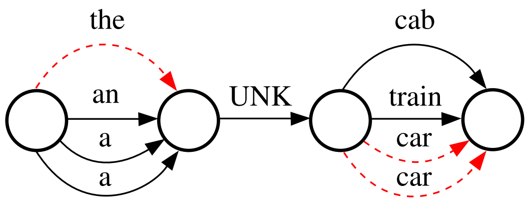

Figure 2: As the words *black*, *red*, *orange*, and *green* in Figure 1 are all not present in the reference (*the blue car*), they are mapped to one single "UNK" arc.

3.2 localVote Model Training

The purpose of the new localVote model is to prefer the best sBLEU path and therefore to learn the word decisions between all adjacent nodes which lead to this particular path. During the extraction of the best sBLEU hypotheses from the confusion network, we keep track of all arc decisions. This gives us the possibility to generate local training examples based only on the I arcs between two nodes. For the confusion network illustrated in Figure 2, we generate two training examples for the neural network training. Based on the arcs *the*, *an*, *a* and *a* we learn the output *the*. Based on the arcs *cab*, *train*, *car* and *car* we learn the output *car*.

In all upcoming system setups, we use the open source toolkit NPLM (Vaswani et al., 2013) for training and testing the neural network models. We use the standard setup as described in the paper and use the neural network with one projection layer and one hidden layer. For more details we refer the reader to the original paper of the NPLM toolkit. The inputs to the neural network are the I words produced by the I different individual systems. The outputs are the posterior probabilities of all words of the vocabulary. The input uses the so-called 1-of-n coding, i.e. the i-th word of the vocabulary is coded by setting the i-th element of the vector to 1 and all the other elements to 0.

For a system combination of I individual systems, a training example consists of $I+1$ words. The first I words (input of the neural network) are representing the words of the individual systems, the last position (output of the neural network) serves as slot for the decision we want to learn (extracted from the best sBLEU path). We do not add the "UNK" arcs to the neural network training as they do not help to increase the sBLEU score. Figure 3 shows the neural network training example for the last words of Figure 2. The output of each

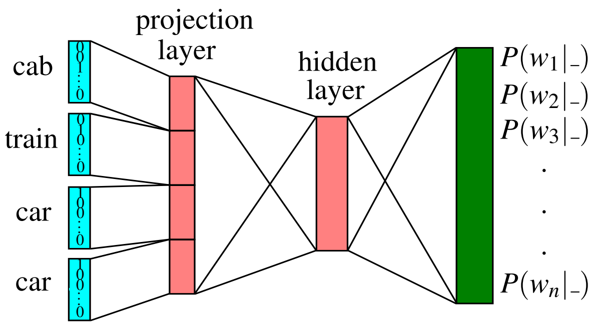

Figure 3: Unigram neural network training example: System *A* produces *cab*, System *B* *train*, System *C* *car*, System *D* *car*, reference is *car*. 1-of-*n* encoding was applied to map words to a suitable neural network input.

Table 1: Training examples from Figure 2.

input layer				
Sys A	Sys B	Sys C	Sys D	ref
the	an	a	a	the
cab	train	car	car	car

individual system provides one input word. In Table 1 the two training examples for Figure 2 are illustrated.

As a neural network training example only consists of the I words between two adjacent nodes, we are able to produce several training examples for each sentences. For a system combination of I systems and a development set of S sentences with an average sentence length of L, we can generate up to $I*S*L$ neural network training examples.

Further, we can expand the model to use arbitrary history size, if we take the predecessor words into account. Instead of just using the local word decision of a system, we add additionally the predecessors of the individual systems into the training data. In Figure 4, we e.g. use the bigram *red train* instead of the unigram *train* for system *B* into the training data. In Table 2 all bigram training examples of Figure 2 can be seen.

3.3 localVote model Integration

Having a trained localVote model, we then add it as an additional model into the confusion network. We calculate for each arc the probability of the word in the trained neural network. E.g. for Figure 1, we extract the probabilities for all arcs by the strings illustrated in Table 3. Finally, we add the scores as a new model and assign it a weight which is trained additionally to the standard model weights with MERT.

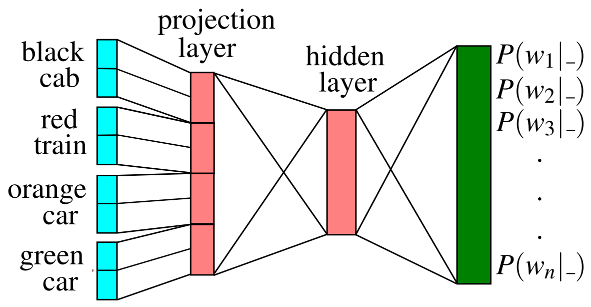

Figure 4: Bigram neural network training example: System *A* produces *black cab*, System *B* *red train*, System *C* *orange car*, System D *green car*, reference is *car*.

Table 2: Training examples (bigram) from Fig. 2.

input layer				
Sys *A*	Sys *B*	Sys *C*	Sys *D*	ref
<s>the	<s>an	<s>a	<s>a	the
black cab	red train	orange car	green car	car

Table 3: Calculating the probability for all possible output words from Figure 1. The output layer is the current generated word.

input layer				
Sys A	Sys B	Sys C	Sys D	arc word
the	an	a	a	the
the	an	a	a	an
the	an	a	a	a
black	red	orange	green	black
black	red	orange	green	red
black	red	orange	green	orange
black	red	orange	green	green
cab	train	car	car	cab
cab	train	car	car	train
cab	train	car	car	car

3.4 Word Classes

The neural network training sets are relatively small as all sentences have to be translated by all individual system engines. This results in many unseen words in the test sets. To overcome this problem, we use word classes (Och, 1999) instead of words which were trained (10 iterations) on the target part of the bilingual training corpus in some experiments. We use the trained word classes on both input layer and output layer.

4 Experiments

All experiments have been conducted with the open source system combination toolkit Jane (Freitag et al., 2014). For training and scoring neural networks, we use the open source toolkit NPLM (Vaswani et al., 2013). NPLM is a toolkit for training and using feedforward neural language models. Variations in neural network architecture have been tested. We tried various hidden layer sizes as well as projection layer sizes. We achieved similar results for all setups and decided to stick to 1 hidden layer whose size is 200, a learning rate of 0.08 and let the training run 20 epochs in all experiments.

Translation quality is measured in lowercase with BLEU (Papineni et al., 2002) and TER (Snover et al., 2006) whereas the performance of each setup is the best score on the tune set across five different MERT runs. The system combination weights of the linear model are optimized with MERT on 200-best lists with (TER-BLEU)/2 as optimization criterion. For all language pairs we use three different test sets. In the following the test set for extracting the training examples for the neural network training is labeled as *tune (NN)*. The test set *tune (MERT)* indicates the tune set for MERT and *test* indicates the blind test set.

The individual systems are different extensions of phrase-based or hierarchical phrase-based systems. The systems are built on the same amount of preprocessed training data and differ mostly in the models which are used to score the translation options. Further, some systems are syntactical augmented based on syntax trees on either source or target side.

4.1 BOLT Chinese→English

For Chinese→English, we use the current BOLT data set (corpus statistics are given in Table 4). The test sets consist of text drawn from "discussion forums" in Mandarin Chinese. We use nine individual systems to perform the system combination experiments. The lambda weights are optimized on a tune set of 985 sentences (tune (MERT)). We train the proposed localVote model on 15,323,897 training examples extracted from the 1844 sentences tune (NN) set.

As a first step we have to determine the k-best pruning threshold for extracting the sBLEU optimal path from the current confusion networks (cf. Section 3.1). In Figure 5 the (TER-BLEU)/2 results of the sBLEU optimal hypotheses extracted with different k-best sizes are given. Although, the BLEU score improves by setting k to a higher value, the computational time increases. To find a tradeoff between running time and performance, we set the k-best size to 1200 in the following experiments.

Table 4: Corpus statistics Chinese→English.

	Chinese	English
Sentences	13M	
Running words	255M	279M
Vocabulary	370K	833K
Tune sentences	1844 (NN), 985 (MERT)	
Test sentences	1124	

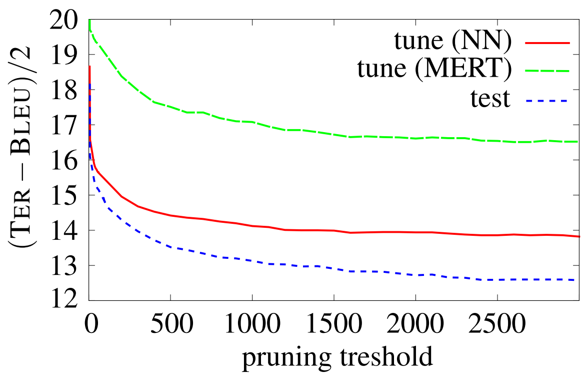

Figure 5: (TER − BLEU)/2 scores for different k-best pruning thresholds on the BOLT Chinese→English data set.

Experimental results are given in Table 5. The *baseline* is a system combination run without any localVote model of nine individual systems using the standard models as described in (Freitag et al., 2014). The *oracle* score is calculated on the hypothesis of the sBLEU best path extracted with $k = 1200$. We train the neural network on 15,323,897 training examples generated from the 1844 tune (NN) sentences. By training a neural network based on unigram decisions (*unigram NN*), we gain small improvements of -0.6 points in TER. As we have only few sentences of training data, many words have not been seen during neural network training. To overcome this problem, we train 1500 word classes on the target part of the bilingual data. Learning the localVote model on word classes (*unigram wcNN*) gain improvement of +0.7 points in BLEU and -0.6 points in

Table 5: Results for the BOLT Chinese→English translation task. The localVote models of the systems +*unigram NN* and +*unigram wcNN* are trained based on one word per system. The localVote models of the systems +*bigram NN* and +*bigram wcNN* are trained based on two words per system. For systems labeled with *wcNN*, the neural network is trained on word classes. Significance is marked with † for 95% confidence and ‡ for 99% confidence, and is measured with the bootstrap resampling method as described in (Koehn, 2004).

system	tune		test	
	BLEU	TER	BLEU	TER
baseline	17.9	61.5	18.3	60.9
+unigram NN	18.1	61.2	18.3	60.3†
+unigram wcNN	18.4	61.5	19.0‡	60.3†
+bigram NN	18.1	61.3	18.6†	60.3†
+bigram wcNN	18.1	61.2	18.7†	59.9‡
oracle	28.6	62.3	31.1	57.2

TER. By taking a bigram history into the training of the neural network, we reach only small further improvement. Compared to the *baseline*, the system combination +*bigram NN* outperforms the *baseline* by +0.3 points in BLEU and -0.6 points in TER. By using word classes (+*bigram wcNN*) we gain improvement of +0.4 points in BLEU and -1.0 points in TER.

All results are reached with a word class size of 1500. In Figure 6 the $(\text{TER} - \text{BLEU})/2$ scores on tune(MERT) of system combinations including one unigram localVote model trained with different word class sizes are illustrated. Independent of the word class size, system combination including a localVote model always performs better compared to the baseline. The best performance is reached by a word class size of 1500. One reason for the loss of performance when using no word classes is the size of the neural network tune set. Within a size of 1844 sentences, many words of the test set have never been seen during neural network training. The test set has a vocabulary size of 6106 within 2487 words (40.73%) are not present in the training set (tune (NN)) of the neural network. For the MERT tune set 2556 words (40.91%) are not present in the neural network training set. Word classes tackle this problem and it is much more likely that each word class has been seen during the training procedure of the neural network.

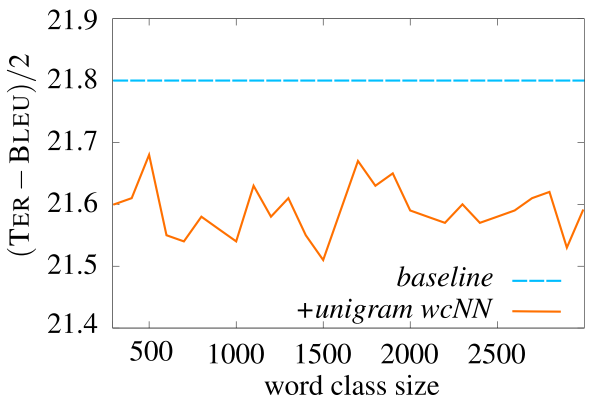

Figure 6: $(\text{TER} - \text{BLEU})/2$ scores for different word class sizes on the BOLT Chinese→English tune (MERT) set.

4.2 BOLT Arabic→English

For Arabic→English, we use the current BOLT data set (corpus statistics are given in Table 6). The test sets consist of text drawn from ”discussion forums” in Egyptian Arabic. We train the neural network on 6,591,158 training examples extracted from the 1510 sentences tune (NN) dev set. The model weights are optimized on a 1080 sentences tune set. All results are system combinations of five individual systems. The test set has a vocabulary size of 3491 within 1510 words (43.25%) are not present in the training set (tune (NN)) of the neural network. For the MERT tune set 1549 words (43.24%) are not part of the neural network training set.

We run the same experiment pipeline as for Chinese→English and first determine the k-best threshold for getting the oracle paths in the confusion networks. As the Arabic→English system combination is only based on 5 individual systems, the confusion networks are much smaller. We set the pruning threshold to 1000 ($k = 1000$) which is a good tradeoff between running time and performance. Figure 7 shows the $(\text{TER} - \text{BLEU})/2$ scores for different k-best pruning thresholds. Increasing k to a higher value then 1000 improves the $(\text{TER} - \text{BLEU})/2$ only slightly.

Experimental results are given in Table 7. The *baseline* is a system combination run without any localVote model of five individual systems using the standard models as described in (Freitag et al., 2014). The *oracle* score represents the score

Table 6: Corpus statistics BOLT Arabic→English.

	Arabic	English
Sentences	8M	
Running words	189M	186M
Vocabulary	608K	519K
Tune sentences	1510 (NN), 1080 (MERT)	
Test sentences	1137	

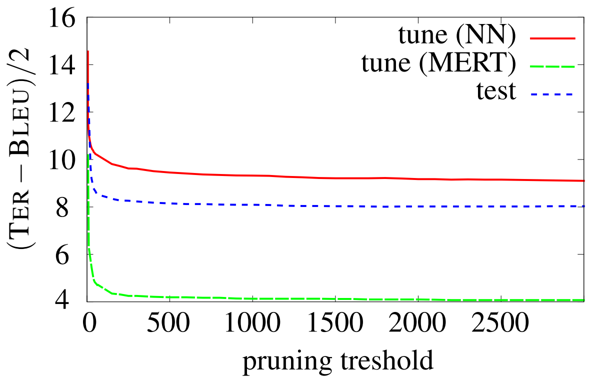

Figure 7: (TER − BLEU)/2 scores for different k-best pruning thresholds on the BOLT Arabic→English tune (MERT) set.

of the sBLEU best path extracted with $k = 1000$. Training a localVote model based on the best sBLEU path (*+unigram NN*) gives us improvement of +0.9 points in BLEU compared to the *baseline*. Adding bigram context to the neural network training (*+bigram NN*) yields improvement of +0.8 points in BLEU compared to the *baseline* system combination. By training word classes on the bilingual part of the training data, we gain additional improvements. When using word classes and a history size of two, *+bigram wcNN* yields the best performance with +1.1 points in BLEU compared to the *baseline*.

All results are conducted with a word class size of 1000. The tune set performance of different unigram localVote models trained on different word class sizes are illustrated in Figure 8. The results are fluctuating and we set the word class size to 1000 in all Arabic→English experiments.

5 Analysis

In this section we compare the final translations of the Chinese→English system combination *+bigram wcNN* with the *baseline*. The word occurrence distributions for both setups are illustrated in Table 8. This table shows how many input systems produce a certain word and finally if it is part of the system combination output. As the original idea of system combination is based on majority voting, it should be more likely that a word which is produced by more input systems is in the final system combination output than a word which is only produced by few input systems. E.g. 11008 words have been produced by all 9 individual systems from which all of them are in both the system combination *baseline* and the advanced system *+bigram wcNN*. If a word is only produced by 8 individual systems, a ninth system does not produce this word. 98,9% of the words produced by only 8 different individual systems are in the final

Table 7: Results for the BOLT Arabic→English translation task. The localVote models of the systems *+unigram NN* and *+unigram wcNN* are trained by a neural network based on one word per system. The localVote models of the systems *+bigram NN* and *+bigram wcNN* are trained by a neural network based on two words per system. For systems labeled with *wcNN*, the neural network is trained on word classes for both input and output layer. Significance is marked with ‡ for 99% confidence and is measured with the bootstrap resampling method as described in (Koehn, 2004).

system	**tune**		**test**	
	BLEU	TER	BLEU	TER
baseline	30.1	51.2	27.6	55.8
+unigram NN	31.4	51.2	28.5‡	56.0
+unigram wcNN	31.1	51.1	28.3‡	55.7
+bigram NN	31.3	51.1	28.4‡	55.8
+bigram wcNN	31.4	51.2	28.7‡	56.0
oracle	38.1	46.3	34.8	50.9

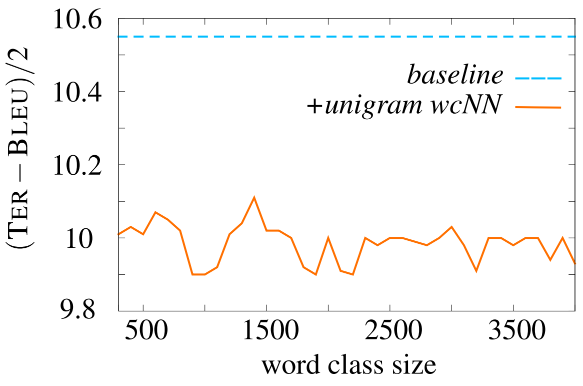

Figure 8: (TER − BLEU)/2 tune set scores for different word class sizes on the BOLT Arabic→English task.

Table 8: Word occurrence distribution for the Chinese→English setup. First column indicates in how many systems a word appears. E.g. 120/14072 (0.9%) indicates that 14072 words only appear in one individual input system from which 120 (0.9%) are present in the baseline system combination hypothesis.

#	*baseline*	*+bigram wcNN*
1	120/14072 (0.9%)	214/14072 (1.5%)
2	592/ 6129 (9.7%)	764/ 6129 (12.5%)
3	1141/ 4159 (27.4%)	1319/ 4159 (31.7%)
4	1573/ 3241 (48.5%)	1669/ 3241 (51.5%)
5	2051/ 2881 (71.2%)	1993/ 2881 (69.2%)
6	2381/ 2744 (86.8%)	2332/ 2744 (85.0%)
7	2817/ 2965 (95.0%)	2820/ 2965 (95.1%)
8	3818/ 3860 (98.9%)	3815/ 3860 (98.8%)
9	11008/11008(100.0%)	11008/11008(100.0%)

baseline system combination output. The missing words result mostly from alignment errors produced by the pairwise alignment algorithm when aligning the single systems together.

We observe the problem that the globalVote models prevent words, which have only been produced by few systems, to be present in the system combination output. In Table 8, you can see that words which are only produced by 1-4 individual systems are more likely to be present in the final output when including the novel localVote model. As e.g. in the baseline 592 of the 6129 words which have only been produced by two individual system are in the output, the advanced *+bigram wcNN* setup contains additional 172 words. These statistics demonstrate the functionality of the novel localVote model which does not only improve the translation quality in terms of BLEU, but also tackles the problem of the dominating globalVote models.

The Arabic→English word occurrence distribution is illustrated in Table 9. A similar scenario as for the Chinese→English translation task can be observed. The words which only occur in few individual systems have a much higher chance to be in the final output when using the novel local voting system model. It is also visible that the neural network model prevents some words of being in the combined output even if the word have been produced by 4 of 5 systems. The novel local system voting model gives system combination the option to select words which have only be generated by few individual systems.

Table 9: Word occurrence distribution for the Arabic→English setup. First column indicates in how many systems a word appears. E.g. 214/5791 (3.7%) indicates that 5791 words only appear in one individual input system from which 214 (3.7%) are present in the baseline system combination hypothesis.

#	*baseline*	*+bigram wcNN*
1	214/ 5791 (3.7%)	285/ 5791 (4.9%)
2	1225/ 3200 (38.3%)	1243/ 3200 (38.8%)
3	2162/ 2719 (79.5%)	2297/ 2719 (84.5%)
4	3148/ 3207 (98.2%)	3119/ 3207 (97.3%)
5	14602/14602(100.0%)	14602/14602(100.0%)

6 Conclusion

In this work we proposed a novel local system voting model (localVote) which has been trained by a feedforward neural network. In contrast to the traditional globalVote model, the presented localVote model takes the word contents and their combinatorial occurrences into account and does not only promote global preferences for some individual systems. This advantage gives confusion network decoding the option to prefer other systems at different positions even in the same sentence. As all words are projected to a continuous space, the neural network gives also unseen word sequences a useful probability. Due to the relatively small neural network training set, we used word classes in some experiments to tackle the data sparsity problem.

Experiments have been conducted with high quality input systems for the BOLT Chinese→English and Arabic→English translation tasks. Training an additional model by a neural network with word classes yields translation improvement from up to +0.9 points in BLEU and -0.5 points in TER. We also took word context into account and added the predecessors of the individual systems to the neural network training which yield additional small improvement. We analyzed the translation results and the functionality of the localVote model. The occurrence distribution shows that words which have been produced by only few input systems are more likely to be part of the system combination output when using the proposed model.

Acknowledgement

This material is partially based upon work supported by the DARPA BOLT project under Contract No. HR0011-12-C-0015. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA. Further, this paper has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n^{o} 645452 (QT21).

References

Srinivas Bangalore, German Bordel, and Giuseppe Riccardi. 2001. Computing consensus translation from multiple machine translation systems. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 351–354, Madonna di Campiglio, Italy, December.

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer.

Markus Freitag, Matthias Huck, and Hermann Ney. 2014. Jane: Open source machine translation system combination. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 29–32, Gothenburg, Sweden, April. Association for Computational Linguistics.

Xiaodong He, Mei Yang, Jianfeng Gao, Patrick Nguyen, and Robert Moore. 2008. Indirect-HMM-based Hypothesis Alignment for Combining Outputs from Machine Translation Systems. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 98–107, Honolulu, HI, USA, October.

Kenneth Heafield and Alon Lavie. 2010. Combining Machine Translation Output with Open Source: The Carnegie Mellon Multi-Engine Machine Translation Scheme. *The Prague Bulletin of Mathematical Linguistics*, 93:27–36.

Dustin Hillard, Björn Hoffmeister, Mari Ostendorf, Ralf Schlüter, and Hermann Ney. 2007. i rover: improving system combination with classification. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 65–68, Rochester, NY, USA, April. Association for Computational Linguistics.

Damianos Karakos, Jason Eisner, Sanjeev Khudanpur, and Markus Dreyer. 2008. Machine translation system combination using ITG-based alignments. In *46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies (ACL): Short Papers*, pages 81–84, Columbus, OH, USA, June.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 388–395, Barcelona, Spain, July.

Gregor Leusch, Evgeny Matusov, and Hermann Ney. 2008. Complexity of finding the bleu-optimal hypothesis in a confusion network. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 839–847, Honolulu, HI, USA, October.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *The 42nd Annual Meeting on Association for Computational Linguistics (ACL)*, page 605, Barcelona, Spain, July.

Evgeny Matusov, Nicola Ueffing, and Hermann Ney. 2006. Computing Consensus Translation from Multiple Machine Translation Systems Using Enhanced Hypotheses Alignment. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 33–40, Trento, Italy, April.

Franz Josef Och. 1999. An Efficient Method for Determining Bilingual Word Classes. In *Ninth Conference on European Chapter of the Association for Computational Linguistics (EACL)*, pages 71–76, Bergen, Norway, June.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *40th Annual Meeting on Association for Computational Linguistics (ACL)*, pages 311–318, Philadelphia, PA, USA, July.

Antti-Veikko Rosti, Necip Fazil Ayan, Bing Xiang, Spyridon Matsoukas, Richard Schwartz, and Bonnie Dorr. 2007. Combining outputs from multiple machine translation systems. In *The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 228–235, Rochester, NY, USA, April.

Holger Schwenk and Jean-Luc Gauvain. 2002. Connectionist language modeling for large vocabulary continuous speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages I–765, Orlando, FL, USA, May.

Khe Chai Sim, William J. Byrne, Mark J.F. Gales, Hichem Sahbi, and Phil C. Woodland. 2007. Consensus Network Decoding for Statistical Machine

Translation System Combination. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 105–108, Honolulu, HI, USA, April.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciula, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Association for Machine Translation in the Americas (AMTA)*, pages 223–231, Cambridge, MA, USA, August.

Le Hai Son, Alexandre Allauzen, and François Yvon. 2012. Continuous space translation models with neural networks. In *The 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*, pages 39–48, Montreal, Canada, June.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1387–1392, Seattle, WA, USA, October.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational linguistics*, 23(3):377–403.