A Comparative Study on Vocabulary Reduction for Phrase Table Smoothing

Yunsu Kim, Andreas Guta, Joern Wuebker*, Hermann Ney

{surname}@cs.rwth-aachen.de

ACL 2016 First Conference on Machine Translation - 12th August, 2016

Human Language Technology and Pattern Recognition
Lehrstuhl für Informatik 6
Computer Science Department
RWTH Aachen University, Germany

*Lilt, Inc.

Phrase Table Smoothing

Phrase translation probability: Sparsity problem

$$p(\tilde{f}|\tilde{e}) = \frac{N(\tilde{f}, \tilde{e})}{N(\tilde{e})}$$

- **Phrase vocabulary is huge!**
- **Bilingual training data is limited**

Smoothing methods

- **Word-based lexicon (a.k.a. IBM-1 lexical models) [Brown & Pietra^{+} 93]**
- **Good-Turing/Kneser-Ney smoothing [Foster & Kuhn^{+} 06]**

$\Rightarrow$ **Any others? Linguistically/mathematically motivated?**

Vocabulary Reduction

Reduce the vocabulary size: Word-to-label mapping

$$f \longmapsto c(f)$$

- $c =$ **word classes, part-of-speech tags, morphological stems, ...**
- **Denser distribution on a smaller vocabulary**
- **Widely used in various NLP tasks**

For phrases:

$$f_1\, f_2 \longmapsto c(f_1)\, c(f_2)$$

- **Robust to rare phrases**
- **Local context preserved**
- **Flexibility in choosing** c

Vocabulary Reduction: Key Questions for Phrase-based SMT

To maximize the phrase table smoothing performance...

1. **Which label vocabulary should we choose?**
   - **Size, structure, linguistic property, ...**
2. **How to apply a label mapping to phrase pairs?**
   - **Model forms**
3. **How much training data do we need?**

Word Classes from Brown Clustering

Word class: group of words with similar syntactic/semantic roles

- **Automatically clustered from training data**
- **Examples [Brown & deSouza^{+} 92]**
  - **Class 1: had hadn't hath would've could've should've must've might've**
  - **Class 2: head body hands eyes voice arm seat eye hair mouth**

Clustering parameters: adjust the vocabulary structure

- **Clustering iterations**
- **Initialization**
- **Number of classes**

$\Rightarrow$ **Easy to obtain various label vocabularies!**

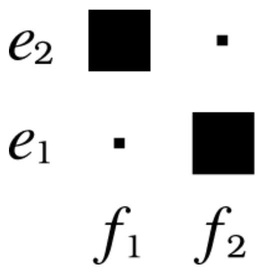
Smoothing Models

map-all: map every word in a phrase at once [Wuebker & Peitz+ 13]

$$f_1\,f_2 \longmapsto c(f_1)\,c(f_2) \qquad e_1\,e_2 \longmapsto c(e_1)\,c(e_2)$$

$$p(\tilde{f}|\tilde{e}) = p(c(\tilde{f})|c(\tilde{e}))$$

map-each: map each word in a phrase at a time (this work)

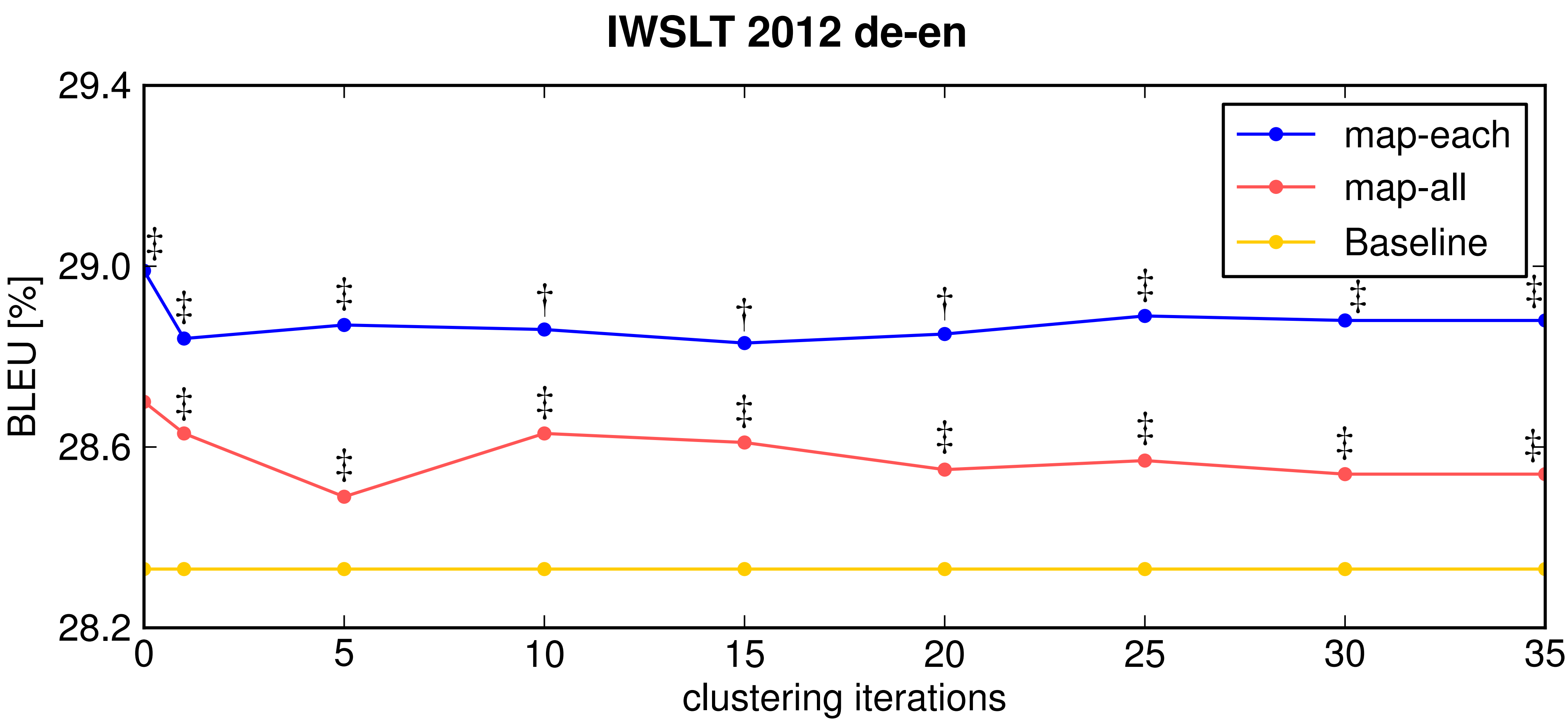


$$f_1\,f_2 \longmapsto c\,(f_1)\;\; f_2 \qquad e_1\,e_2 \longmapsto e_1\;\; c(e_2)$$

$$f_1\,f_2 \longmapsto f_1\;\; c\,(f_2) \qquad e_1\,e_2 \longmapsto c(e_1)\;\; e_2$$

$$p(\tilde{f}|\tilde{e}) = \sum_j w_j \cdot p(c^{(j)}(\tilde{f})|c^{(a_j)}(\tilde{e}))$$

Comparison of Clustering Iterations



Statistical significance: ‡ = 95%, † = 90%
Number of classes = 100

Comparison of Initializations

	Initialization	BLEU [%]
Baseline		28.3
+ map-each	random	28.9‡
	top-frequent	29.0‡
	same-countsum	28.8‡
	same-#words	28.9‡
	count-bins	29.0‡

- **random: randomly assign words to classes**
- **top-frequent: top-frequent words have their own classes, while all other words are in the last class**
- **same-countsum: each class has almost the same sum of word unigram counts**
- **same-#words: each class has almost the same number of words**
- **count-bins: each class represents a bin of the total count range**

Comparison of Label Vocabulary Size

	#vocab (source)	BLEU [%]
Baseline		28.3
+ map-each (word class)	100	29.0‡
	200	28.9†
	500	28.7
	1000	28.7
	10000	28.7
+ map-each (POS)	52	28.9†
+ map-each (lemma)	26744	28.8

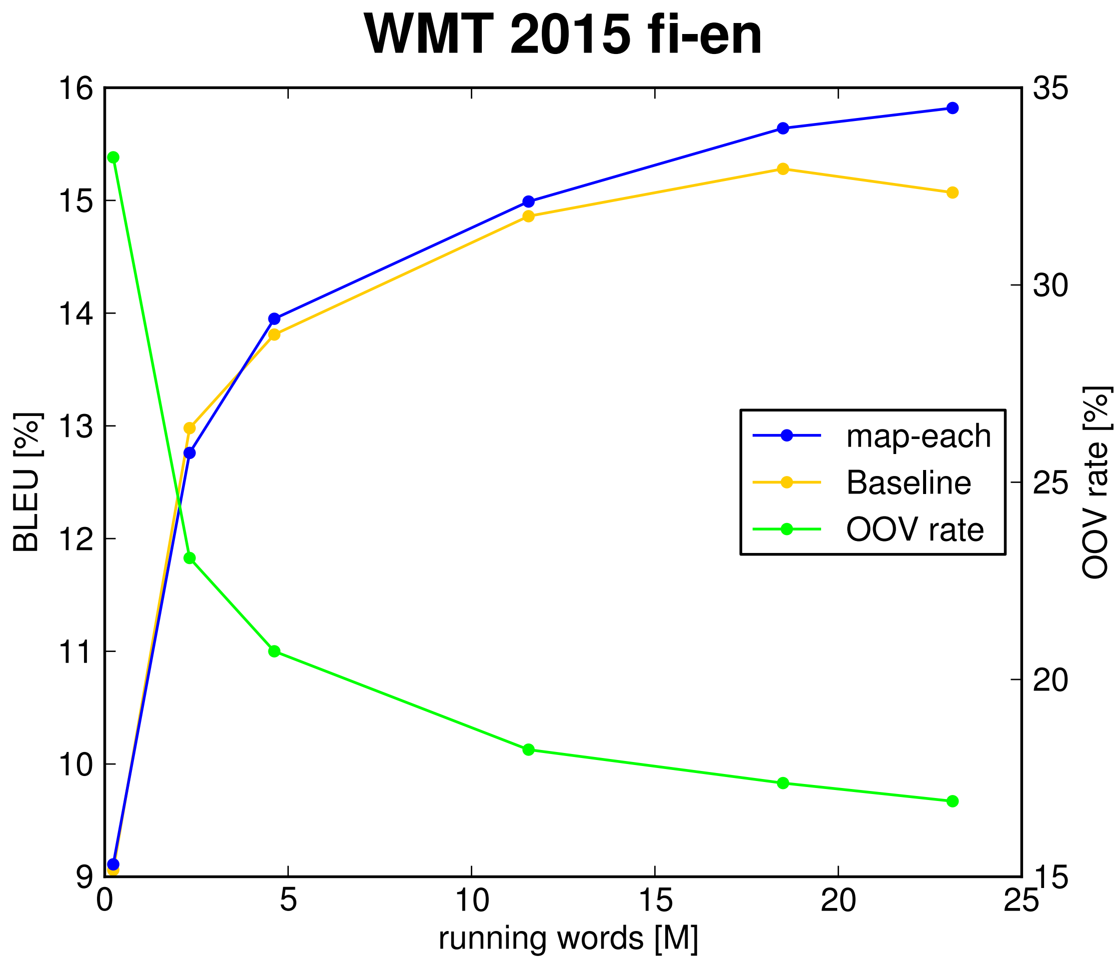
- **Little difference with respect to the vocabulary size**

Comparison of Smoothing Models

	IWSLT 2012 de-en	WMT 2015 fi-en	WMT 2014 en-de	WMT 2015 en-cs
	BLEU [%]	BLEU [%]	BLEU [%]	BLEU [%]
Baseline	28.3	15.1	14.6	15.3
+ map-all	28.6‡	15.3‡	14.8‡	15.4‡
+ map-each	29.0‡	15.8‡	15.1‡	15.8‡

- map-each outperforms map-all consistently

Comparison of Training Data Size



- **Bigger improvement for larger training data**
- **More OOV words for smaller training data: not handled by the smoothing**

Conclusion

Vocabulary reduction for phrase table smoothing

1. yields up to +0.7% BLEU
2. is almost equally effective with any word-label mapping (e.g. randomized labels)
   - Emphasizes the sparsity of the standard phrase translation model
   - Linguistic explanation?
3. performs better when mapping one word in a phrase at a time (map-each)
4. more suitable for large-scale translation tasks

Related work

- Similar comparative experiments on neural machine translation systems [Sennrich & Haddow 16]

Thank you for your attention

Yunsu Kim

kim@cs.rwth-aachen.de

http://www.hltpr.rwth-aachen.de/

References

[Brown & deSouza^{+} 92] P.F. Brown, P.V. deSouza, R.L. Mercer, V.J.D. Pietra, J.C. Lai: Class-based n-gram Models of Natural Language. *Computational Linguistics*, Vol. 18, No. 4, pp. 467–479, December 1992.

[Brown & Pietra^{+} 93] P.F. Brown, V.J.D. Pietra, S.A.D. Pietra, R.L. Mercer: The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, Vol. 19, No. 2, pp. 263–311, June 1993.

[Foster & Kuhn^{+} 06] G. Foster, R. Kuhn, H. Johnson: Phrasetable Smoothing for Statistical Machine Translation. In *2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pp. 53–61, Sydney, Austrailia, July 2006.

[Sennrich & Haddow 16] R. Sennrich, B. Haddow: Linguistic Input Features Improve Neural Machine Translation. In *Proceedings of ACL 2016 1st Conference on Machine Translation (WMT 2016)*, pp. 83–91, Berlin, Germany, August 2016.

[Wuebker & Peitz^{+} 13] J. Wuebker, S. Peitz, F. Rietig, H. Ney: Improving Statistical Machine Translation with Word Class Models. In *Proceedings of 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pp. 1377–1381, Seattle, USA, October 2013.

Appendix: Position-dependent Weights for map-each

- **(inverse) unigram of the replaced word**

$$\frac{1}{w_j} = \frac{N(f_j)}{\sum_{f'} N(f')}$$

- **(inverse) source phrase replacement probability**

$$\frac{1}{w_j} = \frac{N(f_{b_k} \dots f_j \dots f_{j_k})}{\sum_{f'} N(f_{b_k} \dots f' \dots f_{j_k})}$$

- **factorizing likelihood**

$$w_j = N(c^{(j)}(\tilde{f}))$$