Vector Space Model for Adaptation in Statistical Machine Translation

Boxing Chen, Roland Kuhn and George Foster
National Research Council Canada
first.last@nrc-cnrc.gc.ca

Abstract

This paper proposes a new approach to domain adaptation in statistical machine translation (SMT) based on a vector space model (VSM). The general idea is first to create a vector profile for the in-domain development ("dev") set. This profile might, for instance, be a vector with a dimensionality equal to the number of training subcorpora; each entry in the vector reflects the contribution of a particular subcorpus to all the phrase pairs that can be extracted from the dev set. Then, for each phrase pair extracted from the training data, we create a vector with features defined in the same way, and calculate its similarity score with the vector representing the dev set. Thus, we obtain a decoding feature whose value represents the phrase pair's closeness to the dev. This is a simple, computationally cheap form of instance weighting for phrase pairs. Experiments on large scale NIST evaluation data show improvements over strong baselines: +1.8 BLEU on Arabic to English and +1.4 BLEU on Chinese to English over a non-adapted baseline, and significant improvements in most circumstances over baselines with linear mixture model adaptation. An informal analysis suggests that VSM adaptation may help in making a good choice among words with the same meaning, on the basis of style and genre.

1 Introduction

The translation models of a statistical machine translation (SMT) system are trained on parallel data. Usage of language and therefore the best translation practice differs widely across genres, topics, and dialects, and even depends on a particular author's or publication's style; the word "domain" is often used to indicate a particular combination of all these factors. Unless there is a perfect match between the training data domain and the (test) domain in which the SMT system will be used, one can often get better performance by adapting the system to the test domain.

Domain adaptation is an active topic in the natural language processing (NLP) research community. Its application to SMT systems has recently received considerable attention. Approaches that have been tried for SMT model adaptation include mixture models, transductive learning, data selection, instance weighting, and phrase sense disambiguation, etc.

Research on mixture models has considered both linear and log-linear mixtures. Both were studied in (Foster and Kuhn, 2007), which concluded that the best approach was to combine sub-models of the same type (for instance, several different TMs or several different LMs) linearly, while combining models of different types (for instance, a mixture TM with a mixture LM) log-linearly. (Koehn and Schroeder, 2007), instead, opted for combining the sub-models directly in the SMT log-linear framework.

In transductive learning, an MT system trained on general domain data is used to translate in-domain monolingual data. The resulting bilingual sentence pairs are then used as additional training data (Ueffing et al., 2007; Chen et al., 2008; Schwenk, 2008; Bertoldi and Federico, 2009).

Data selection approaches (Zhao et al., 2004; Hildebrand et al., 2005; Lü et al., 2007; Moore and Lewis, 2010; Axelrod et al., 2011) search for bilingual sentence pairs that are similar to the in-domain "dev" data, then add them to the training data.

Instance weighting approaches (Matsoukas et al., 2009; Foster et al., 2010; Huang and Xiang, 2010; Phillips and Brown, 2011; Sennrich, 2012)

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 1285–1293,
Sofia, Bulgaria, August 4-9 2013.

typically use a rich feature set to decide on weights for the training data, at the sentence or phrase pair level. For example, a sentence from a subcorpus whose domain is far from that of the dev set would typically receive a low weight, but sentences in this subcorpus that appear to be of a general nature might receive higher weights.

The 2012 JHU workshop on Domain Adaptation for MT [1] proposed phrase sense disambiguation (PSD) for translation model adaptation. In this approach, the context of a phrase helps the system to find the appropriate translation.

In this paper, we propose a new instance weighting approach to domain adaptation based on a vector space model (VSM). As in (Foster et al., 2010), this approach works at the level of phrase pairs. However, the VSM approach is simpler and more straightforward. Instead of using word-based features and a computationally expensive training procedure, we capture the distributional properties of each phrase pair directly, representing it as a vector in a space which also contains a representation of the dev set. The similarity between a given phrase pair's vector and the dev set vector becomes a feature for the decoder. It rewards phrase pairs that are in some sense closer to those found in the dev set, and punishes the rest. In initial experiments, we tried three different similarity functions: Bhattacharyya coefficient, Jensen-Shannon divergency, and cosine measure. They all enabled VSM adaptation to beat the non-adaptive baseline, but Bhattacharyya similarity worked best, so we adopted it for the remaining experiments.

The vector space used by VSM adaptation can be defined in various ways. In the experiments described below, we chose a definition that measures the contribution (to counts of a given phrase pair, or to counts of all phrase pairs in the dev set) of each training subcorpus. Thus, the variant of VSM adaptation tested here bears a superficial resemblance to domain adaptation based on mixture models for TMs, as in (Foster and Kuhn, 2007), in that both approaches rely on information about the subcorpora from which the data originate. However, a key difference is that in this paper we explicitly capture each phrase pair's distribution across subcorpora, and compare it to the aggregated distribution of phrase pairs in the dev set. In mixture models, a phrase pair's distribution across subcorpora is captured only implicitly, by probabilities that reflect the prevalence of the pair within each subcorpus. Thus, VSM adaptation occurs at a much finer granularity than mixture model adaptation. More fundamentally, there is nothing about the VSM idea that obliges us to define the vector space in terms of subcorpora.

For instance, we could cluster the words in the source language into S clusters, and the words in the target language into T clusters. Then, treating the dev set and each phrase pair as a pair of bags of words (a source bag and a target bag) one could represent each as a vector of dimension S + T, with entries calculated from the counts associated with the S + T clusters (in a way similar to that described for phrase pairs below). The (dev, phrase pair) similarity would then be independent of the subcorpora. One can think of several other ways of defining the vector space that might yield even better results than those reported here. Thus, VSM adaptation is not limited to the variant of it that we tested in our experiments.

2 Vector space model adaptation

Vector space models (VSMs) have been widely applied in many information retrieval and natural language processing applications. For instance, to compute the sense similarity between terms, many researchers extract features for each term from its context in a corpus, define a VSM and then apply similarity functions (Hindle, 1990; Lund and Burgess, 1996; Lin, 1998; Turney, 2001).

In our experiments, we exploited the fact that the training data come from a set of subcorpora. For instance, the Chinese-English training data are made up of 14 subcorpora (see section 3 below). Suppose we have C subcorpora. The domain vector for a phrase-pair (f, e) is defined as

$$V(f,e) =< w_1(f,e), ...w_i(f,e), ..., w_C(f,e) >, \quad (1)$$

where $w_i(f,e)$ is a standard $tf \cdot idf$ weight, i.e.

$$w_i(f,e) = tf_i(f,e) \cdot idf(f,e). \quad (2)$$

To avoid a bias towards longer corpora, we normalize the raw joint count $c_i(f,e)$ in the corpus s_i by dividing by the maximum raw count of any phrase pair extracted in the corpus s_i. Let

[1] http://www.clsp.jhu.edu/workshops/archive/ws-12/groups/dasmt

$$tf_i(f,e) = \frac{c_i(f,e)}{max\{c_i(f_j,e_k), (f_j,e_k) \in s_i\}}. \quad (3)$$

The $idf(f,e)$ is the inverse document frequency: a measure of whether the phrase-pair (f,e) is common or rare across all subcorpora. We use the standard formula:

$$idf(f,e) = log\left(\frac{C}{df(f,e)} + \lambda\right), \quad (4)$$

where $df(f,e)$ is the number of subcorpora that (f,e) appears in, and λ is an empirically determined smoothing term.

For the in-domain dev set, we first run word alignment and phrases extracting in the usual way for the dev set, then sum the distribution of each phrase pair (f_j, e_k) extracted from the dev data across subcorpora to represent its domain information. The dev vector is thus

$$V(dev) =< w_1(dev), \ldots, w_C(dev) >, \quad (5)$$

where

$$w_i(dev) = \sum_{j=0}^{j=J} \sum_{k=0}^{k=K} c_{dev}(f_j,e_k)\, w_i(f_j,e_k) \quad (6)$$

J, K are the total numbers of source/target phrases extracted from the dev data respectively. $c_{dev}(f_j,e_k)$ is the joint count of phrase pair f_j, e_k found in the dev set.

The vector can also be built with other features of the phrase pair. For instance, we could replace the raw joint count $c_i(f,e)$ in Equation 3 with the raw marginal count of phrase pairs (f,e). Therefore, even within the variant of VSM adaptation we focus on in this paper, where the definition of the vector space is based on the existence of subcorpora, one could utilize other definitions of the vectors of the similarity function than those we utilized in our experiments.

2.1 Vector similarity functions

VSM uses the similarity score between the vector representing the in-domain dev set and the vector representing each phrase pair as a decoder feature. There are many similarity functions we could have employed for this purpose (Cha, 2007). We tested three commonly-used functions: the Bhattacharyya coefficient (BC) (Bhattacharyya, 1943; Kazama et al., 2010), the Jensen-Shannon divergence (JSD), and the cosine measure. According to (Cha, 2007), these belong to three different families of similarity functions: the Fidelity family, the Shannon's entropy family, and the inner Product family respectively. It was BC similarity that yielded the best performance, and that we ended up using in subsequent experiments.

To map the BC score onto a range from 0 to 1, we first normalize each weight in the vector by dividing it by the sum of the weights. Thus, we get the probability distribution of a phrase pair or the phrase pairs in the dev data across all subcorpora:

$$p_i(f,e) = \frac{w_i(f,e)}{\sum_{j=1}^{j=C} w_j(f,e)} \quad (7)$$

$$p_i(dev) = \frac{w_i(dev)}{\sum_{j=1}^{j=C} w_j(dev)} \quad (8)$$

To further improve the similarity score, we apply absolute discounting smoothing when calculating the probability distributions $p_i(f,e)$. We subtract a discounting value α from the non-zero $p_i(f,e)$, and equally allocate the remaining probability mass to the zero probabilities. We carry out the same smoothing for the probability distributions $p_i(dev)$. The smoothing constant α is determined empirically on held-out data.

The Bhattacharyya coefficient (BC) is defined as follows:

$$BC(dev;f,e) = \sum_{i=0}^{i=C} \sqrt{p_i(dev) \cdot p_i(f,e)} \quad (9)$$

The other two similarity functions we also tested are JSD and cosine (Cos). They are defined as follows:

$$JSD(dev;f,e) = \quad (10)$$
$$\frac{1}{2}[\sum_{i=1}^{i=C} p_i(dev) \log \frac{2p_i(dev)}{p_i(dev)+p_i(f,e)} +$$
$$\sum_{i=1}^{i=C} p_i(f,e) \log \frac{2p_i(f,e)}{p_i(dev)+p_i(f,e)}]$$

$$Cos(dev;f,e) = \frac{\sum_i p_i(dev) \cdot p_i(f,e)}{\sqrt{\sum_i p_i^2(dev)}\sqrt{\sum_i p_i^2(f,e)}} \quad (11)$$

corpus	# segs	# en tok	%	genres
fbis	250K	10.5M	3.7	nw
financial	90K	2.5M	0.9	fin
gale_bc	79K	1.3M	0.5	bc
gale_bn	75K	1.8M	0.6	bn ng
gale_nw	25K	696K	0.2	nw
gale_wl	24K	596K	0.2	wl
hkh	1.3M	39.5M	14.0	hans
hkl	400K	9.3M	3.3	legal
hkn	702K	16.6M	5.9	nw
isi	558K	18.0M	6.4	nw
lex&ne	1.3M	2.0M	0.7	lex
other_nw	146K	5.2M	1.8	nw
sinorama	282K	10.0M	3.5	nw
un	5.0M	164M	58.2	un
TOTAL	10.1M	283M	100.0	(all)
devtest				
tune	1,506	161K		nw wl
NIST06	1,664	189K		nw bng
NIST08	1,357	164K		nw wl

Table 1: NIST Chinese-English data. In the *genres* column: nw=newswire, bc=broadcast conversation, bn=broadcast news, wl=weblog, ng=newsgroup, un=UN proc., bng = bn & ng.

corpus	# segs	# en toks	%	gen
gale_bc	57K	1.6M	3.3	bc
gale_bn	45K	1.2M	2.5	bn
gale_ng	21K	491K	1.0	ng
gale_nw	17K	659K	1.4	nw
gale_wl	24K	590K	1.2	wl
isi	1,124K	34.7M	72.6	nw
other_nw	224K	8.7M	18.2	nw
TOTAL	1,512K	47.8M	100.0	(all)
devtest				
NIST06	1,664	202K		nwl
NIST08	1,360	205K		nwl
NIST09	1,313	187K		nwl

Table 2: NIST Arabic-English data. In the *gen* (genres) column: nw=newswire, bc=broadcast conversation, bn=broadcast news, ng=newsgroup, wl=weblog, nwl = nw & wl.

3 Experiments

3.1 Data setting

We carried out experiments in two different settings, both involving data from NIST Open MT 2012.[2] The first setting is based on data from the Chinese to English constrained track, comprising about 283 million English running words. We manually grouped the training data into 14 corpora according to genre and origin. Table 1 summarizes information about the training, development and test sets; we show the sizes of the training subcorpora in number of words as a percentage of all training data. Most training subcorpora consist of parallel sentence pairs. The *isi* and *lex&ne* corpora are exceptions: the former is extracted from comparable data, while the latter is a lexicon that includes many named entities. The development set (*tune*) was taken from the NIST 2005 evaluation set, augmented with some web-genre material reserved from other NIST corpora.

The second setting uses NIST 2012 Arabic to English data, but excludes the UN data. There are about 47.8 million English running words in these training data. We manually grouped the training data into 7 groups according to genre and origin. Table 2 summarizes information about the training, development and test sets. Note that for this language pair, the comparable isi data represent a large proportion of the training data: 72% of the English words. We use the evaluation sets from NIST 2006, 2008, and 2009 as our development set and two test sets, respectively.

3.2 System

Experiments were carried out with an in-house phrase-based system similar to Moses (Koehn et al., 2007). Each corpus was word-aligned using IBM2, HMM, and IBM4 models, and the phrase table was the union of phrase pairs extracted from these separate alignments, with a length limit of 7. The translation model (TM) was smoothed in both directions with KN smoothing (Chen et al., 2011). We use the hierarchical lexicalized reordering model (RM) (Galley and Manning, 2008), with a distortion limit of 7. Other features include lexical weighting in both directions, word count, a distance-based RM, a 4-gram LM trained on the target side of the parallel data, and a 6-gram English *Gigaword* LM. The system was tuned with batch lattice MIRA (Cherry and Foster, 2012).

3.3 Results

For the baseline, we simply concatenate all training data. We have also compared our approach to two widely used TM domain adaptation ap-

[2] http://www.nist.gov/itl/iad/mig/openmt12.cfm

proaches. One is the log-linear combination of TMs trained on each subcorpus (Koehn and Schroeder, 2007), with weights of each model tuned under minimal error rate training using MIRA. The other is a linear combination of TMs trained on each subcorpus, with the weights of each model learned with an EM algorithm to maximize the likelihood of joint empirical phrase pair counts for in-domain dev data. For details, refer to (Foster and Kuhn, 2007).

The value of λ and α (see Eq 4 and Section 2.1) are determined by the performance on the dev set of the Arabic-to-English system. For both Arabic-to-English and Chinese-to-English experiment, these values obtained on Arabic dev were used to obtain the results below: λ was set to 8, and α was set to 0.01. (Later, we ran an experiment on Chinese-to-English with λ and α tuned specifically for that language pair, but the performance for the Chinese-English system only improved by a tiny, insignificant amount).

Our metric is case-insensitive IBM BLEU (Papineni et al., 2002), which performs matching of n-grams up to n = 4; we report BLEU scores averaged across both test sets NIST06 and NIST08 for Chinese; NIST08 and NIST09 for Arabic. Following (Koehn, 2004), we use the bootstrap-resampling test to do significance testing. In tables 3 to 5, * and ** denote significant gains over the baseline at $p < 0.05$ and $p < 0.01$ levels, respectively.

We first compare the performance of different similarity functions: cosine (COS), Jensen-Shannon divergence (JSD) and Bhattacharyya coefficient (BC). The results are shown in Table 3. All three functions obtained improvements. Both COS and BC yield statistically significant improvements over the baseline, with BC performing better than COS by a further statistically significant margin. The Bhattacharyya coefficient is explicitly designed to measure the overlap between the probability distributions of two statistical samples or populations, which is precisely what we are trying to do here: we are trying to reward phrase pairs whose distribution is similar to that of the dev set. Thus, its superior performance in these experiments is not unexpected.

In the next set of experiments, we compared VSM adaptation using the BC similarity function with the baseline which concatenates all training data and with log-linear and linear TM mixtures

system	Chinese	Arabic
baseline	31.7	46.8
COS	32.3*	47.8**
JSD	32.1	47.1
BC	33.0**	48.4**

Table 3: Comparison of different similarity functions. * and ** denote significant gains over the baseline at $p < 0.05$ and $p < 0.01$ levels, respectively.

system	Chinese	Arabic
baseline	31.7	46.8
loglinear tm	28.4	44.5
linear tm	32.7**	47.5**
vsm, BC	**33.0****	**48.4****

Table 4: Results for variants of adaptation.

whose components are based on subcorpora. Table 4 shows that log-linear combination performs worse than the baseline: the tuning algorithm failed to optimize the log-linear combination even on dev set. For Chinese, the BLEU score of the dev set on the baseline system is 27.3, while on the log-linear combination system, it is 24.0; for Arabic, the BLEU score of the dev set on the baseline system is 46.8, while on the log-linear combination system, it is 45.4. We also tried adding the global model to the loglinear combination and it didn't improve over the baseline for either language pair. Linear mixture was significantly better than the baseline at the $p < 0.01$ level for both language pairs. Since our approach, VSM, performed better than the linear mixture for both pairs, it is of course also significantly better than the baseline at the $p < 0.01$ level.

This raises the question: is VSM performance significantly better than that of a linear mixture of TMs? The answer (not shown in the table) is that for Arabic to English, VSM performance is better than linear mixture at the $p < 0.01$ level. For Chinese to English, the argument for the superiority of VSM over linear mixture is less convincing: there is significance at the $p < 0.05$ for one of the two test sets (NIST06) but not for the other (NIST08). At any rate, these results establish that VSM adaptation is clearly superior to linear mixture TM adaptation, for one of the two language pairs.

In Table 4, the VSM results are based on the

system	Chinese	Arabic
baseline	31.7	46.8
linear tm	32.7**	47.5**
vsm, joint	33.0**	48.4**
vsm, src-marginal	32.2*	47.3*
vsm, tgt-marginal	32.6**	47.6**
vsm, src+tgt (2 feat.)	32.7**	48.2**
vsm, joint+src (2 feat.)	32.9**	48.4**
vsm, joint+tgt (2 feat.)	32.9**	48.4**
vsm, joint+src+tgt (3 feat.)	**33.1****	**48.6****

Table 5: Results for adaptation based on joint or maginal counts.

vector of the joint counts of the phrase pair. In the next experiment, we replace the joint counts with the source or target marginal counts. In Table 5, we first show the results based on source and target marginal counts, then the results of using feature sets drawn from three decoder VSM features: a joint count feature, a source marginal count feature, and a target marginal count feature. For instance, the last row shows the results when all three features are used (with their weights tuned by MIRA). It looks as though the source and target marginal counts contain useful information. The best performance is obtained by combining all three sources of information. The 3-feature version of VSM yields +1.8 BLEU over the baseline for Arabic to English, and +1.4 BLEU for Chinese to English.

When we compared two sets of results in Table 4, the joint count version of VSM and linear mixture of TMs, we found that for Arabic to English, VSM performance is better than linear mixture at the $p < 0.01$ level; the Chinese to English significance test was inconclusive (VSM found to be superior to linear mixture at $p < 0.05$ for NIST06 but not for NIST08). We now have somewhat better results for the 3-feature version of VSM shown in Table 5. How do these new results affect the VSM vs. linear mixture comparison? Naturally, the conclusions for Arabic don't change. For Chinese, 3-feature VSM is now superior to linear mixture at $p < 0.01$ on NIST06 test set, but 3-feature VSM still doesn't have a statistically significant edge over linear mixture on NIST08 test set. A fair summary would be that 3-feature VSM adaptation is decisively superior to linear mixture adaptation for Arabic to English, and highly competitive with linear mixture adaptation for Chinese to English.

Our last set of experiments examined the question: when added to a system that already has some form of linear mixture model adaptation, does VSM improve performance? In (Foster and Kuhn, 2007), two kinds of linear mixture were described: linear mixture of language models (LMs), and linear mixture of translation models (TMs). Some of the results reported above involved linear TM mixtures, but none of them involved linear LM mixtures. Table 6 shows the results of different combinations of VSM and mixture models. * and ** denote significant gains over the row $no\ vsm$ at $p < 0.05$ and $p < 0.01$ levels, respectively. This means that in the table, the baseline within each box containing three results is the topmost result in the box. For instance, with an initial Chinese system that employs linear mixture LM adaptation (lin-lm) and has a BLEU of 32.1, adding 1-feature VSM adaptation (+vsm, joint) improves performance to 33.1 (improvement significant at $p < 0.01$), while adding 3-feature VSM instead (+vsm, 3 feat.) improves performance to 33.2 (also significant at $p < 0.01$). For Arabic, including either form of VSM adaptation always improves performance with significance at $p < 0.01$, even over a system including both linear TM and linear LM adaptation. For Chinese, adding VSM still always yields an improvement, but the improvement is not significant if linear TM adaptation is already in the system. These results show that combining VSM adaptation and either or both kinds of linear mixture adaptation never hurts performance, and often improves it by a significant amount.

3.4 Informal Data Analysis

To get an intuition for how VSM adaptation improves BLEU scores, we compared outputs from the baseline and VSM-adapted system ("vsm, joint" in Table 5) on the Chinese test data. We focused on examples where the two systems had translated the same source-language (Chinese) phrase s differently, and where the target-language (English) translation of s chosen by the VSM-adapted system, t_V, had a higher Bhattacharyya score for similarity with the dev set than did the phrase that was chosen by the baseline system, t_B. Thus, we ignored differences in the two translations that might have been due to the secondary effects of VSM adaptation (such as a different tar-

		no-lin-adap	lin-lm	lin-tm	lin-lm+lin-tm
Chinese	no vsm	31.7	32.1	32.7	33.1
	+vsm, joint	33.0**	33.1**	33.0	33.3
	+vsm, 3 feat.	33.1**	33.2**	33.1	**33.4**
Arabic	no vsm	46.8	47.0	47.5	47.7
	+vsm, joint	48.4**	48.7**	48.6**	48.8**
	+vsm, 3 feat.	48.6**	48.8**	48.7**	**48.9****

Table 6: Results of combining VSM and linear mixture adaptation. "lin-lm" is linear language model adaptation, "lin-tm" is linear translation model adaptation. * and ** denote significant gains over the row "no vsm" at $p < 0.05$ and $p < 0.01$ levels, respectively.

get phrase being preferred by the language model in the VSM-adapted system from the one preferred in the baseline system because of a Bhattacharyya-mediated change in the phrase preceding it).

An interesting pattern soon emerged: the VSM-adapted system seems to be better than the baseline at choosing among synonyms in a way that is appropriate to the genre or style of a text. For instance, where the text to be translated is from an informal genre such as weblog, the VSM-adapted system will often pick an informal word where the baseline picks a formal word with the same or similar meaning, and vice versa where the text to be translated is from a more formal genre. To our surprise, we saw few examples where the VSM-adapted system did a better job than the baseline of choosing between two words with different meaning, but we saw many examples where the VSM-adapted system did a better job than the baseline of choosing between two words that both have the same meaning according to considerations of style and genre.

Two examples are shown in Table 7. In the first example, the first two lines show that VSM finds that the Chinese-English phrase pair (殴打, assaulted) has a Bhattacharyya (BC) similarity of 0.556163 to the dev set, while the phrase pair (殴打, beat) has a BC similarity of 0.780787 to the dev. In this situation, the VSM-adapted system thus prefers "beat" to "assaulted" as a translation for 殴打. The next four lines show the source sentence (SRC), the reference (REF), the baseline output (BSL), and the output of the VSM-adapted system. Note that the result of VSM adaptation is that the rather formal word "assaulted" is replaced by its informal near-synonym "beat" in the translation of an informal weblog text.

"apprehend" might be preferable to "arrest" in a legal text. However, it looks as though the VSM-adapted system has learned from the dev that among synonyms, those more characteristic of news stories than of legal texts should be chosen: it therefore picks "arrest" over its synonym "apprehend".

What follows is a partial list of pairs of phrases (all single words) from our system's outputs, where the baseline chose the first member of a pair and the VSM-adapted system chose the second member of the pair to translate the same Chinese phrase into English (because the second word yields a better BC score for the dev set we used). It will be seen that nearly all of the pairs involve synonyms or near-synonyms rather than words with radically different senses (one exception below is "center" vs "heart"). Instead, the differences between the two words tend to be related to genre or style: gunmen-mobsters, champion-star, updated-latest, caricatures-cartoons, spill-leakage, hiv-aids, inkling-clues, behaviour-actions, deceit-trick, brazen-shameless, aristocratic-noble, circumvent-avoid, attack-criticized, descent-born, hasten-quickly, precipice-cliff, center-heart, blessing-approval, imminent-approaching, stormed-rushed, etc.

4 Conclusions and future work

This paper proposed a new approach to domain adaptation in statistical machine translation, based on vector space models (VSMs). This approach measures the similarity between a vector representing a particular phrase pair in the phrase table and a vector representing the dev set, yielding a feature associated with that phrase pair that will be used by the decoder. The approach is simple, easy to implement, and computationally cheap. For the two language pairs we looked at, it provided a large performance improvement over a non-adaptive baseline, and also compared

1	phrase pairs	殴打↔ assaulted (0.556163) 殴打↔ beat (0.780787)
	SRC	...那些殴打村民的地皮流氓...
	REF	... those local ruffians and hooligans who beat up villagers ...
	BSL	... those who assaulted the villagers land hooligans ...
	VSM	... hooligans who beat the villagers ...
2	phrase pairs	缉拿↔ apprehend (0.286533) 缉拿↔ arrest (0.603342)
	SRC	... 缉拿凶手并且将之绳之以法。
	REF	... catch the killers and bring them to justice .
	BSL	... apprehend the perpetrators and bring them to justice .
	VSM	... arrest the perpetrators and bring them to justice .

Table 7: Examples show that VSM chooses translations according to considerations of style and genre.

favourably with linear mixture adaptation techniques.

Furthermore, VSM adaptation can be exploited in a number of different ways, which we have only begun to explore. In our experiments, we based the vector space on subcorpora defined by the nature of the training data. This was done purely out of convenience: there are many, many ways to define a vector space in this situation. An obvious and appealing one, which we intend to try in future, is a vector space based on a bag-of-words topic model. A feature derived from this topic-related vector space might complement some features derived from the subcorpora which we explored in the experiments above, and which seem to exploit information related to genre and style.

References

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *EMNLP 2011*.

Nicola Bertoldi and Marcello Federico. 2009. Domain adaptation for statistical machine translation with monolingual resources. In *Proceedings of the 4th Workshop on Statistical Machine Translation*, Athens, March. WMT.

A. Bhattacharyya. 1943. On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematical Society*, 35:99–109.

Sung-Hyuk Cha. 2007. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models ind Methods in Applied Sciences*, 1(4):300–307.

Boxing Chen, Min Zhang, Aiti Aw, and Haizhou Li. 2008. Exploiting n-best hypotheses for smt self-enhancement. In *ACL 2008*.

Boxing Chen, Roland Kuhn, George Foster, and Howard Johnson. 2011. Unpacking and transforming feature functions: New ways to smooth phrase tables. In *MT Summit 2011*.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *NAACL 2012*.

George Foster and Roland Kuhn. 2007. Mixture-model adaptation for SMT. In *Proceedings of the ACL Workshop on Statistical Machine Translation*, Prague, June. WMT.

George Foster, Cyril Goutte, and Roland Kuhn. 2010. Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Boston.

Michel Galley and C. D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *EMNLP 2008*, pages 848–856, Hawaii, October.

Almut Silja Hildebrand, Matthias Eck, Stephan Vogel, and Alex Waibel. 2005. Adaptation of the translation model for statistical machine translation based on information retrieval. In *Proceedings of the 10th EAMT Conference*, Budapest, May.

Donald Hindle. 1990. Noun classification from predicate.argument structures. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 268–275, Pittsburgh, PA, June. ACL.

Fei Huang and Bing Xiang. 2010. Feature-rich discriminative phrase rescoring for SMT. In *COLING 2010*.

Jun'ichi Kazama, Stijn De Saeger, Kow Kuroda, Masaki Murata, and Kentaro Torisawa. 2010. A

bayesian method for robust estimation of distributional similarities. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 247–256, Uppsala, Sweden, July. ACL.

Philipp Koehn and Josh Schroeder. 2007. Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 224–227, Prague, Czech Republic, June. Association for Computational Linguistics.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL 2007, Demonstration Session*.

P. Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Barcelona, Spain.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING/ACL-98*, pages 768–774, Montreal, Quebec, Canada.

Yajuan Lü, Jin Huang, and Qun Liu. 2007. Improving Statistical Machine Translation Performance by Training Data Selection and Optimization. In *Proceedings of the 2007 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Prague, Czech Republic.

K. Lund and C. Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods Instruments and Computers*, 28(2):203–208.

Spyros Matsoukas, Antti-Veikko I. Rosti, and Bing Zhang. 2009. Discriminative corpus weight estimation for machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Singapore.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *ACL 2010*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, Philadelphia, July. ACL.

Aaron B. Phillips and Ralf D. Brown. 2011. Training machine translation with a second-order taylor approximation of weighted translation instances. In *MT Summit 2011*.

Holger Schwenk. 2008. Investigations on large-scale lightly-supervised training for statistical machine translation. In *IWSLT 2008*.

Rico Sennrich. 2012. Perplexity minimization for translation model domain adaptation in statistical machine translation. In *EACL 2012*.

Peter Turney. 2001. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *Twelfth European Conference on Machine Learning*, page 491–502, Berlin, Germany.

Nicola Ueffing, Gholamreza Haffari, and Anoop Sarkar. 2007. Transductive learning for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, Prague, Czech Republic, June. ACL.

Bing Zhao, Matthias Eck, and Stephan Vogel. 2004. Language model adaptation for statistical machine translation with structured query models. In *Proceedings of the International Conference on Computational Linguistics (COLING) 2004*, Geneva, August.